19941106 (this version 19950101)
Bo Nordin (bosse@cb.uu.se)
Centre for Image Analysis
018-183470

Brief PHIGS Reference

Introduction

This document is essentially a listing of the two files *phigsextrn.c* (functions) and *phigs.h* (datatypes, enumerations, and constants) but also an introduction to PHIGS. There are many more PHIGS functions available than those listed here. I have removed all functions dealing with archives, gse's, application data, and Store objects, as well as many gdp functions. I have also removed a few cases of C++ specialisations. This document consists of two parts: the first is a list of PHIGS functions, ordered after function, the second is a list of PHIGS datatypes, in alphabetical order.

In order to find out how the arguments to a particular function should be given, look it up in the datatype section. Some Pints are not really integers, but rather enumerations; such constants can be found in the section PHIGS constants after PHIGS datatypes. I have identified most of them (I think) in the function prototypes; I have called them __xxx in the comments.

NOTE: there are two versions of the document: one full version and one version without the function and datatypes description.

PHIGS Overview

This section is a very brief version of *PHIGS Programming Manual* by Tom Gaskins. What I'm describing here is PHIGS+ with some SUN extensions. PHIGS+ features are generally distinguished by either of the words plus or extended.

How to draw with PHIGS

PHIGS is a tool for *visualising models*. You describe your model and display it on a workstation (a workstation in PHIGS is *not* identical to a SUN workstation, for instance; it is a window on a workstation rather). There are four steps in this:

1. define the model in the PHIGS database, the *CSS (central structure storage)*
2. create a PHIGS workstation to display the model
3. link the model's description to the workstation
4. tell the workstation to display the model.

This is done by the following PHIGS program (OK, it's only a skeleton of a program):

```
popen_phigs()   /* open phigs, parameter are "fixed" */
popen_struct()
...
/* create a structure with attributes and primitives, i.e. 1. above */
...
pclose_struct()
```

```
popen_ws();  /* 2: open a workstation */
ppost_struct(); /* 3. post the model to the workstation */
pupd_ws(); /* 4. update the workstation */
.....
punpost_struct();
pclose_ws();
pclose_phigs();
```

Of course, there's a lot more to drawing in PHIGS but these are the basic steps.

PHIGS datatypes and functions

The file phigs.h contain all PHIGS' datatypes and phigsextrn.h all its functions. In PHIGS, the names of datatypes all begin with a capital "P" (Ppoint3, for instance) while function names begin with the lower-case letter "p" (ppolyline, for instance).

In addition to datatypes and functions, PHIGS also defines mnemonic names for many constants. These identifiers are all upper-case letters, as PIN_PICK, for instance.

Structures and traversal

Structures are opened with popen_struct which also gives each structure a unique name (which you select). You close a structure with pclose_struct. Only one structure may be opened at any time. Once you have created a structure it must be associated with a workstation and displayed. You associate it with a workstation with ppost_struct. It tells PHIGS that a structure is to be displayed on a particular workstation. It's not displayed until the workstation is updated, though. Either you update the workstation explicitly by calling pupd_ws (or predraw_all_structs) or you direct the workstation to automatically update with every change by calling pset_disp_upd_st.

In fact, what you are doing when adding primitives or attributes (*elements*) to an open structure is creating a small program. When PHIGS displays the structure, it steps through the elements of the structure. If the element is an attribute PHIGS sets that attribute and if it is a primitive PHIGS draws it (using the current attributes). This process is called *traversal* and is what happens when a workstation is updated. Elements may be other things than attributes and primitives, as we shall see.

One important element type is the invocation of another structure. This is called executing another structure and is done with pexec_struct.

Primitives

PHIGS can draw the following primitives:

Line primitives:
  polylines (ppolyline, ppolyline3, ppolyline_set3_data),
  B-spline curve (pnuni_bsp_curv)
Area primitives:
  fill area (pfill_area, fill_area3, pfill_area_set, pfill_area_set3_data,
            ptri_strip3_data, pquad_mesh3_data, pnuni_bsp_surf)
Text primitives:

general: (ptext, ptext3)
annotation (panno_text_rel, panno_text_rel3)

Other primitives:

polymarker (ppolymarker, ppolymarker3)
cell array (pcell_array, pcell_array3, cell_array3_plus)
generalized drawing primitive (pgdp, pgdp3)

The difference between text and annotation text is that annotation text is always displayed in the plane of the screen and thus remain legible regardless of how the model is rotated or scaled.

Colour

Colour is probably the most important attribute and is therefore treated separately. I here assume that PHIGS runs a colour workstation. In PHIGS you specify a colour with a *colour type* and a *colour value*. These are passed to PHIGS in a Pgcolr structure:

```
typedef struct { Pint type;  union { Pint ind; struct { Pfloat x, y, z } general; } val; } Pgcolr;
```

The type field is for the colour type and the val union is the value. In this document I will assume that type is PMODEL_RGB or PINDIRECT. The difference between the two is in how a colour is defined. If type is PMODEL_RGB the val union member general contains the red (x), green (y), and blue (z) components of the colour. If type is PINDIRECT the union member ind is an index in a colour table. You set entries in the colour table with pset_colr_rep. As all indexed attributes, colour indices, too, are workstation dependent. In the lists of attribute setting functions below the suffix _ind indicates a colour index (pset_line_colr thus takes a full colour definition as argument while pset_line_colr_ind takes a colour index).

It is important to understand the difference between direct and indirect colour: the same type of indirection is used for bundled attributes and for views.

Bundled attributes

In PHIGS, attributes can be set either individually or in groups (or *bundles*). Bundling provides a way to control attributes. There are separate bundles for each workstation for line, interior (area), edge, text, and marker attributes. You select bundles with bundle index structure elements. There is a separate element for each bundle. You define bundles with one function and select (i.e. create an element in a structure) them with another

Many of the bundles have a shortform for setting them in addition to the full form. Line attributes for instance have the two forms pset_line_rep_plus (full form with all attributes) and pset_line_rep (shortform with only some attributes). The defined bundle is then selected with pset_line_ind.

Bundles contain most of the attributes relevant to their corresponding primitive type. But you can select which of these attributes are taken from the bundle and which are taken from the currently defined values in the traversal list. You make this selection with aspect source flags (aspect is just another name for attribute). Use the function pset_indiv_asf for this.

Bundles are workstation-specific and there exist a number of predefined bundles on each workstation.

Line attributes

Lines have the following attributes:

linetype	pset_linetype
linewidth	pset_linewidth,
linecolour	pset_line_colr, pset_line_colr_ind
line shading method:	pset_line_shad_meth
curve appr. criteria:	pset_curve_approx (b-spline curves only)

The bundled form is selected with pset_line_ind and define with pset_line_rep or pset_line_rep_plus.

Area attributes

Area have the following attributes (this is only a selection):

edge colour:	pset_edge_colr, pset_edge_colr_ind
edge flag:	pset_edge_flag (controls whether edges are displayed or not)
edgetype:	pset_edgetype
edgewidth:	pset_edgewidth
face culling mode:	pset_face_cull_mode
interior colour:	pset_int_colr, pset_int_colr_ind
interior style:	pset_int_style
pattern ref point:	pset_pat_ref_point
pattern reference:	pset_pat_ref_point_vecs
pattern size:	pset_pat_size
pattern:	pset_pat_rep, pset_pat_rep_plus (workstation dependent)

There are two bundles of area primitives. These are, with selection and definition functions:

interior:	pset_int_int	pset_int_rep, pset_int_rep_plus.
edge:	pset_edge_int	pset_edge_rep, pset_edge_rep_plus

The edge bundle contains the first four area attributes while the interior bundle contains colour and style plus many more.

Marker attributes

Marker has the following attributes:

type:	pset_marker_type
size:	pset_marker_size
colour:	pset_marker_colr, pset_marker_colr_ind

The bundled form is selected with pset_marker_ind and defined with pset_marker_rep or pset_marker_rep_plus.

Text attributes

Text has the following attributes:

font:	pset_text_font
character height:	pset_char_ht
character spacing:	pset_char_space
character expansion:	pset_char_expan
character up vector:	pset_char_up_vec
path:	pset_text_path
alignment:	pset_text_align
precision:	pset_text_prec
colour:	pset_text_colr, pset_text_colr_ind

The bundled form is selected with pset_text_ind and defined with pset_text_rep or pset_text_rep_plus.

Annotation text has essentially the same functionality. One additional attribute is annotation style (pset_anno_style).

Modelling

The modelling stage is the creation of graphical objects in their own coordinate system and then transforming them to the world coordinate system. The local transform is the work-horse of the modelling stage. You use to position, scale, and rotate primitives into place. It can be regarded as just another attribute. You can create a local transform with pset_local_tran(3). There are help functions to create transformation matrices:

protate_x, protate_y, protate_z, pscale(3), ptranslate(3), protate,
pbuild_tran_matrix(3)

Matrices can be concatenated with pcompose_matrix(3).

Building hierarchies is an important part of modelling. This is done by invoking other structures (pexec_struct). If you build a street with houses, for instance, you would start by building a window struct, a door struct, and a chimney struct in their own coordinate system, then build a house in its coordinate system. In the street struct you would then set a local transformation matrix before each invocation of house. In the house struct you would select a different local transformation for the chimney, each window, and the door. The two local transformations will be autmatically concatenated.

Viewing

Once a model is assembled, we'd like to see it from various directions. If modelling is the transformation of the model (objects are moved), viewing is the transformation of the viewer (you, the viewer, moves). PHIGS uses the model described in Foley, chapter 6, for viewing. The view reference coordinate system defines the *view orientation* and the choice of projection, the view volume and the projection to a viewport define the *view mapping*.

You evaluate the view orientation matrix with peval_view_ori_matrix(3) and the view

mapping matrix with peval_view_map_matrix(3*)*. You can then define your view with pset_view_rep3.
This last function defines a view representation index much like colour indices or bundle indices. You select the view index for a view with pset_view_ind. You cannot change view index 0 (this is also the default).

CSS, Structure editing, and inheritance rules

One important feature of PHIGS is the possibility to edit structures after they have been created. But first a little about inheritance rules: how do executed structures inherit attributes, transformations, etc.

Executed structures inherit the attributes, transformations, names (see Input), aspect source flags, bundle indices, etc. from their ancestors. Changes by children do not affect ancestors, i.e. attributes, etc. are restored.

When you call popen_struct the element pointer is set to point after the last element. PHIGS has a number of functions for resetting the pointer, deleting elements, replacing elements (by changing the edit mode). In particular, you can insert a special kind of element (label) which you can use to position the pointer. In SUN PHIGS you can also move the element pointer to a pick element (see Input). Because elements that we want to remove often are made pickable this last function largely removes the need for labels.

Input

PHIGS input isn't as sophisticated as it could be. Gaskins suggests that we should use X input whenever possible. But here I will concentrate on PHIGS' input functions.

There are six types of logical input devices in PHIGS:
locator *(loc)*, *stroke*, *pick*, *choice*, valuator *(val)*, and *string*
All devices may be used in three different modes:
*reques*t, *sample*, or *event*.
You set the mode with pset_xxx_mode where xxx is one of the italicised items above. Unless you want to change default parameters for a device you don't have to initialise it explicitly.

Request mode functions are preq_xxx, sample mode function are psample_xxx. where xxx is one of the six modes.

In event mode we use pawait_event in the event loop. One of the output parameters from this function tells us what device triggered the event. You use this information to select one of the functions pget_xxx.

For locator input, PHIGS automatically converts device coordinate points to world coordinate points.

Pick devices are very useful but also hard to use. This is because of the hierarchical model that PHIGS use and also because of the (I must admit) overly complicated approach PHIGS takes. In order for a object (primitive, structure, or whatever) to be pickable it must be given a name and a pick id. A name is set with padd_names_set and a pick id with pset_pick_id. Both these are modal attributes so if we want everything to be pickable we just set a name first in the main structure. This will then be inherited by all substructures. Now everything would be fine if it weren't for the pick filter. This is filter which determines which names are pickable: in other words, we must include the name (assuming we use the simple model outlined above) in the pick filter with pset_pick_filter. Any picked primitive is identified by the pick path: this path contains the structure id, pick id, and element number, for the picked primitive and all its ancestors. A variable in the returned pick path tells us the depth of the hierarchy.

If events occurred where views overlap, PHIGS returns an event in the view with the highest priority. By default this is view 0 but you can change that with pset_view_tran_in_pri.

Other functions

PHIGS has many inquiry functions. You can ask PHIGS about the state of PHIGS, workstation capabilities, elements in structures, bundles, input functionality, and more...

Other functions include hidden line and surface removal and the possibility to control in detail the rendering pipeline.

Relation between SPHIGS and PHIGS

This section contains a list of some SPHIGS functions (those used in Foley et al, chapter 7) and their counterparts in PHIGS. The list is in alphabetical order with the SPHIGS name as sort key. Together with chapter 7 in Foley et al and the on-line manual answerbook, this list can be used as a reference manual.

SPH_closeStructure	pclose_struct
SPH_composeMatrix	pcompose_matrix, pcompose_matrix3
SPH_deleteElement	pdel_elem
SPH_evaluateViewOrientationMatrix	peval_view_ori_matrix3
SPH_evaluarteViewMappingMatrix	peval_view_map_matrix3
SPH_executeStructure	pexec_struct
SPH_fillArea	pfill_area, pfill_area3
SPH_label	plabel
SPH_moveElementPointerToLabel	pset_elem_ptr_label
SPH_offsetElementPointer	poffset_elem_ptr
SPH_openStructure	popen_struct
SPH_pickCorrelate	----------
SPH_polyhedron	pset_of_fill_area3, and more...
SPH_polyLine	ppolyline, ppolyline3
SPH_polyMarker	polymarker, ppolymarker3
SPH_postRoot	ppost_struct
SPH_regenerateScreen	pupd_ws, predraw_all_structs
SPH_rotateX	protate_x, protate
SPH_rotateY	protate_y, protate
SPH_rotateZ	protate_z
SPH_scale	pscale, pscale3
SPH_setEdgeColor	pset_edge_colr, pset_edge_colr_ind
SPH_setEdgeFlag	pset_edge_flag

SPH_setEdgeStyle	pset_edgetype
SPH_setEdgeWidthScaleFactor	pset_edgewidth
SPH_setElementPointer	pset_elem_ptr
SPH_setImplicitRegenerationMode	pset_disp_upd_st
SPH_setInteriorColor	pset_int_colr, pset_int_colr_ind
SPH_setLineColor	pset_line_colr, pset_line_colr_ind
SPH_setLineStyle	pset_linetype
SPH_setLineWidthScaleFactor	pset_linewidth
SPH_setLocalTransformation	pset_local_tran, pset_local_tran3
SPH_setMarkerColor	pset_,marker_colr, pset_marker_colr_ind
SPH_setMarkerSizeScaleFactor	pset_marker_size
SPH_setMarkerStyle	pset_marker_type
SPH_setPickIdentifier	pset_pick_id
SPH_setRenderingMode	
SPH_setTextColor	pset_text_colr, pset_text_colr_ind
SPH_setTextFont	pset_text_font
SPH_setViewRepresentation	pset_view_rep3
SPH_text	ptext, ptext3, panno_text_rel, panno_text_rel3
SPH_translate	ptranslate, ptranslate3
SPH_unpostRoot	punpost_struct

An example program

This example program draws a polyline. The purpose of the program is to show what a very simple PHIGS program must contain in terms of opening PHIGS, opening workstations, etc.

```
/* Copyright 1992 O'Reilly and Associates, Inc. Permission to use, copy,
 * and modify this program is hereby granted, as long as this copyright
 * notice appears in each copy of the program source code.
 */

#include <phigs/phigs..h>
#define WORKSTATION_ID      (Pint) 1
#define STRUCTURE_ID                    (Pint) 1

main() {
 Ppoint_list3     point_list;

 static Ppoint3 points() = {{.2,.2,0},{.4,.7,0},{.6,.3,0},{.8,.8,0}};

 point_list.num_points = 4;
 point_list.points = points;

 popen_phigs( PDEF_ERR_FILE, PDEF_MEM_SIZE );
 popen_struct( STRUCTURE_ID );
 pset_linetype( PLINE_DASH );
 ppolyline3( &point_list );
 pclose_struct();

 popen_ws( WORKSTATION_ID, NULL, phigs_ws_type_x_tool );
 ppost_struct( WORKSTATION_ID, STRUCTURE_ID, (Pfloat) 1.0 );
 pupd_ws( WORKSTATION_ID, PFLAG_PERFORM );

 sleep( 10 );   /* let's show it for 10 seconds */
 pclose_ws( WORKSTATION_ID );
 pclose_phigs();
}
```

PHIGS functions

Opening, closing PHIGS workstations

This sectiom contains control functions

```
void popen_phigs( /* OPEN PHIGS, MUST BE CALLED BEFORE CALLING OTHER PHIGS FCNS */
 char   *error_file,  /* error file name (use PDEF_ERR_FILE) */
 Pint    memory  ); /* not used (use PDEF_MEM_SIZE) */

void popen_xphigs( /* OPEN PHIGS, A MORE FLEXIBLE POPEN_PHIGS */
 char             *error_file,        /* error file name (use PDEF_ERR_FILE) */
 size_t           memory,
 unsigned longxinfo_mask,
 Pxphigs_info    *xinfo );

void pclose_phigs(); /* CLOSE PHIGS, AFTER CLOSING ALL STRUCTURES */

void pemergency_close_phigs(); /* CLOSE PHIGS EVEN IF A STRUCTURE IS OPEN */

void popen_ws( /* OPEN A WORKSTATION AND CLEAR IT */
 Pint ws_id,                      /* workstation identifier (your choice) */
 void *conn_id,                   /* connection identifier (use (void*)NULL, for instance) */
 Pint ws_type );                  /* workstation type (use phigs_ws_type_x_tool) */

void pclose_ws( /* CLOSE A WORKSTATION */
 Pint ws_id );                    /* workstation identifier */

void pmessage( /* SEND A MESSAGE TO A WORKSTATION xxxxxxxxxxxxxxxxxxxxx */
 Pint    ws,                      /* workstation identifier          */
 char   *msg );                   /* message string                  */

void predraw_all_structs( /* FORCED REDISPLAY OF A WORKSTATION */
 Pint    ws,                      /* workstation identifier          */
 Pctrl_flag       control_flag );   /* controls the redraw of the structures: PFLAG_COND/PFLAG_ALWAYS */

void pupd_ws( /* UPDATE A WORKSTATION TO THE CURRENT CONTENTS OF CSS */
 Pint    ws,                      /* workstation identifier          */
 Pregen_flag    regen_flag ); /* when to do the regeneration: PFLAG_PERFORM/PFLAG_POSTPONE */;

void pset_disp_upd_st( /* SET DISPLAY UPDATE STATE */
 Pint     ws,                     /* workstation identifier          */
 Pdefer_mode def_mode,      /* deferral mode: PDEFER_xx where xx is ASAP, BNIG, BNIL, ASTI, or WAIT */
 Pmod_mode mod_mode );   /* modification mode: PMODE_xx where xx is NIVE, UWOR, or UQUM */

void pinq_sys_st( /* INQUIRE THE SYSTEM STATE  */
 Psys_st           *sys_state );      /* OUT the system state: PHOP or PHCL */

void pinq_ws_st( /* INQUIRE THE WORKSTATION STATE */
 Pws_st            *ws_state );      /* OUT workstation state: WSOP or WSCL */

void pinq_ws_conn_type( /* INQUIRE THE SPECIFIC TYPE OF A WORKSTATION */
 Pint    ws,                      /* workstation identifier          */
 Pstorestore,                     /* handle to Store object          */
 Pint    *error_ind,              /* OUT error indicator                         */
 void    **conn_id,               /* OUT connection identifier   */
 Pint    *ws_type );              /* OUT workstation type                        */

void pinq_open_wss( /* INQUIRE LIST OF ALL WORKSTATIONS */
 Pint    length,                  /* length of application list      */
 Pint    start,                   /* starting position               */
 Pint    *error_ind,              /* OUT error indicator             */
 Pint_list *idlist,               /* OUT list of ws ids              */
 Pint    *total_length );         /* OUT length of list in PHIGS   */
```

```
void pinq_disp_upd_st( /* INQUIRE THE DISPLAY UPDATE STATE */
 Pint             ws,            /* workstation identifier */
 Pint            *error_ind,     /* OUT error indicator */
 Pdefer_mode *def_mode,     /* OUT deferral mode */
 Pmod_mode  *mod_mode,  /* OUT modification mode     */
 Pdisp_surf_empty *disp_empty, /* OUT display surface empty */
 Pvisual_st       *state );       /* OUT state of visual representation */

caddr_t phigs_ws_type_set( /* SET A WORKSTATION TYPE xxxxxxxxxxxxxx */
 Pint            wst,
 ... );

caddr_t phigs_ws_type_get( /* GET A LIST OF WORKSTATION ATTRIBUTES xxxxxxxxxxxx*/
 Pint    wst,
 char   *attr,
 ... );

Pint phigs_ws_type_create( /* CREATE A NEW WORKSTATION TYPE */
 Pint    base_wst,
 ... );

void phigs_ws_type_destroy( /* DELETE A WORKSTATION TYPE xxxxxxxxxxxx*/
 Pint    wst );

void pinq_list_avail_ws_types( /* INQUIRE AVAILABLE WORKSTATION TYPES */
 Pint    length,                 /* length of application list */
 Pint    start,                  /* starting position */
 Pint    *error_ind,             /* OUT error indicator          */
 Pint_list        *types,        /* OUT list of ws types         */
 Pint    *length_list );         /* OUT length of list in PHIGS */

void pinq_num_disp_pris( /* INQUIRE THE NUMBER OF AVAILABLE DISPLAY PRIORITIES */
 Pint    wst,                    /* workstation type             */
 Pint    *error_ind,             /* OUT error indicator          */
 Pint    *num_pri );             /* OUT number of display priorities          */

void pinq_ws_cat( /* INQUIRE THE TYPE CATEGORY OF A WORKSTATION */
 Pint    type,                   /* workstation type             */
 Pint    *error_ind,             /* OUT error indicator          */
 Pws_cat *category );            /* OUT workstation category  */

void pinq_ws_class( /* INQUIRE THE CLASS OF A WORKSTATION */
 Pint    type,                   /* workstation type             */
 Pint    *error_ind,             /* OUT error indicator          */
 Pws_class *class );             /* OUT workstation class: PCLASS_xx where xx is one of VEC, RASTER, or OTHER */

void pinq_dyns_ws_attrs( /* INQUIRE WHICH WORKSTATION LIST CHANGES CAN BE QUICKLY UPDATED */
 Pint    type,                   /* workstation type             */
 Pint    *error_ind,             /* OUT error indicator          */
 Pdyns_ws_attrs );               *attr                           /* OUT attributes dynamics     */

void pinq_dyns_structs( /* INQUIRE WHETHER CERTAIN CHANGES TO STRUCTURES CAN BE QUICKLY UPDATED */
 Pint            type,           /* workstation type             */
 Pint            *error_ind,     /* OUT error indicator          */
 Pdyns_structs  *dynamics );     /* OUT structure dynamics       */

void pinq_dyns_ws_attrs_plus( /* INQUIRE WHETHER CHANGES TO LIGHT, DEPTH CUE,
                                 AND COLOUR MAPPING CAN BE QUICKLY UPDATED */
 Pint            ws_type,        /* IN: workstation type */
 Pint            *err_ind,       /* OUT: error indicator */
 Pdyns_ws_attrs_plus *attr );    /* OUT: dynamic mod attributes */

void pinq_ws_st_table( /* INQUIRE WORKSTATION TABLE LENGTHS */
 Pint            type,           /* workstation type             */
 Pint            *error_ind,     /* OUT error indicator          */
 Pws_st_tables *lengths );       /* OUT lengths of workstation tables          */
```

```
void pinq_ws_st_table_plus( /* INQUIRE EXTENDED (PHIGS+) WORKSTATION TABLE LENGTHS */
 Pint              type,          /* workstation type              */
 Pint              *error_ind,    /* OUT error indicator           */
 Pws_tables_plus *lengths );      /* OUT lengths of workstation tables            */

void pinq_disp_space_size3( /* INQUIRE THE 3D DEVICE COORDINATES LIMITS */
 Pint              type,          /* workstation type              */
 Pint              *error_ind,    /* OUT error indicator           */
 Pdisp_space_size3 *size );       /* OUT display size              */

void pinq_disp_space_size( /* INQUIRE THE 2D DEVICE COORDINATES LIMITS */
 Pint              type,          /* workstation type              */
 Pint              *error_ind,    /* OUT error indicator           */
 Pdisp_space_size *size );        /* OUT display size              */

void pinq_def_disp_upd_st( /* INQUIRE THE DEFAULT DISPLAY UPDATE STATE */
  Pint             type,          /* workstation type              */
  Pint             *error_ind,    /* OUT error indicator           */
  Pdefer_mode *def_mode,   /* OUT deferral mode */
  Pmod_mode *mod_mode ); /* OUT modification mode */ );

void pinq_phigs_facs( /* INQUIRE PHIGS DESCRIPTION TABLE  */
 Pint    length,                  /* length of application list    */
 Pint    start,                   /* starting position             */
 Pint    *error_ind,              /* OUT error indicator           */
 Pint    *open_ws,                /* OUT max. num. simultaneously open ws     */
 Pint    *open_ar,                /* OUT max. num. simultaneously open archive files */
 Pint    *num_names,              /* OUT number of available names for name sets */
 Pint_list *char_sets,            /* OUT list of character sets    */
 Pint *length_list,               /* OUT length of list in PHIGS */
 Pint    *norm_filt,              /* OUT maximum length of normal filter list for ISS */
 Pint    *inv_filt );             /* OUT maximum length of inverted filter list for ISS */
```

Structures and structure editing

This sectiom contains functions for structure handling and editing

```
void popen_struct( /* OPEN A PHIGS STRUCTURE, CREATING IT IF IT DOES NOT EXIST */
 Pint    struct_id );             /* structure identifier          */

void pclose_struct(); /* CLOSE A PHIGS STRUCTURE */

void pexec_struct( /* EXECUTE A PHIGS STRUCTURE, INVOKE IT FROM ANOTHER*/
 Pint    struct_id );             /* structure identifier          */

void ppost_struct( /* POST A STRUCTURE TO A WORKSTATION */
 Pint    ws_id,                   /* workstation identifier        */
 Pint    struct_id,               /* structure identifier          */
 Pfloat_arg priority );           /* priority         */

void punpost_struct( /* UNPOST A STRUCTURE FROM A WORKSTATION */
 Pint    ws_id,                   /* workstation identifier        */
 Pint    struct_id );             /* structure identifier          */

void punpost_all_structs( /* UNPOST ALL STRUCTURES FROM A WORKSTATION */
 Pint    ws_id );                 /* workstation identifier        */

void pinq_struct_st( ./* INQUIRE THE STRUCTURE STATE */
 Pstruct_st *struct_state );      /* OUT structure state: STOP or STCL */

void pinq_open_struct( /* RETURN THE ID OF THE CURRENTLY OPEN STRUCTURE */
 Pint              *error_ind,    /* OUT error indicator           */
 Popen_struct_status *status,     /* OUT status of open structure                */
 Pint              *struct_id );  /* OUT structure identifier      */

void pinq_struct_status( /* RETURN INFO ABOUT WHETHER A STRUCTURE EXISTS */
```

```
Pint           struct_id,       /* structure identifier          */
Pint           *error_ind,      /* OUT error indicator           */
Pstruct_status *status );       /* OUT existence status          */

void pinq_posted_structs( /* INQUIRE A LIST OF POSTED STRUCTURES */
Pint   ws,                      /* workstation identifier        */
Pint   length,                  /* length of application list    */
Pint   start,                   /* starting position             */
Pint   *error_ind,              /* OUT error indicator           */
Pposted_struct_list *list,      /* OUT list of posted structures              */
Pint   *total_length );         /* OUT length of list in PHIGS   */
```

Editing functions

```
void pcopy_all_elems_struct( /* COPY ALL ELEMENTS FROM A STRUCTURE */
Pint   struct_id );             /* structure identifier          */

void pcopy_elem_range( /* COPY A RANGE OF ELEMENTS FROM A STRUCTURE */
Pint           struct_id,       /* source structure identifier */
Pint           ep1_value,       /* element pointer 1 value */
Pint           ep2_value );     /* element pointer 2 value */

void pcopy_elems_labels( /* COPY ALL ELEMENTS BETWEEN TWO POINTERS */
Pint           struct_id,       /* source structure identifier */
Pint           label1_id,       /* label 1 identifier            */
Pint           label2_id );     /* label 2 identifier            */

void pdel_elem(); /* DELETE A SINGLE ELEMENT AT THE POS. OF POINTER */

void pdel_elem_range( /* DELETE A RANGE OF ELEMENTS */
Pint   ep1_value,               /* element pointer 1 value       */
Pint   ep2_value );             /* element pointer 2 value       */

void pdel_elems_labels( /* DELETE ALL ELEMENTS BETWEEN TWO POINTERS */
Pint   label1_id,               /* label 1 identifier            */
Pint   label2_id );             /* label 2 identifier            */

void pempty_struct( /* DELETE ALL ELEMENTS IN A STRUCTURE */
Pint   struct_id );             /* structure id   */

void pdel_struct( /* DELETE AN INDIVIDUAL STRUCTURE */
Pint   struct_id );             /* structure identifier          */

void pdel_struct_net( /* DELETE ALL STRUCTURES IN A STRUCTURE NETWORK */
Pint   struct_id,               /* structure identifier          */
Pref_flag      ref_flag );      /* reference handling flag: FLAG_DEL OR FLAG_KEEP *

void pdel_all_structs(); /* DELETE ALL STRUCTURES IN THE CSS */

void pchange_struct_id( /* CHANGE THE NAME OF A STRUCTURE */
Pint   orig_struct_id,          /* originial structure id        */
Pint   result_struct_id );      /* result structure id           */

void pchange_struct_refs( /* REPLACE ALL REF'S TO A STRUCTURE WITH REFS TO ANOTHER */
Pint   orig_struct_id,          /* originial structure id        */
Pint   result_struct_id );      /* result structure id           */

void pchange_struct_id_refs( /* CHANGE THE NAME OF STRUCTURE, UPDATE ALL REF'S TO IT */
Pint   orig_struct_id,          /* originial structure id        */
Pint   result_struct_id );      /* result structure id           */

void pset_elem_ptr( /* MOVE ELEMENT POINTER TO A SPECIFIC POSITION */
Pint   ep_value );              /* element pointer value         */

void poffset_elem_ptr( /* MOVE POINTER FWD OR BWD FROM ITS POSITION */
Pint   ep_offset );             /* element pointer offset        */
```

```
void plabel( /* ADD A LABEL TO A STRUCTURE */
 Pint    label_id );                 /* label identifier                 */

void pset_elem_ptr_label( /* MOVE ELEMENT POINTER TO A LABEL ELEMENT */
 Pint    label_id );                 /* label identifier                 */

void pset_elem_ptr_pick_id( /* MOVE ELEMENT POINTER TO A PICK IDENTIFIER */
 Pint            pick_id );          /* pick identifier */

void pelem_search( /* SEARCH FOR A PARTICULAR ELEMENT TYPE IN A STRUCTURE */
 Pint            struct_id,          /* structure identifier             */
 Pint            start_el,           /* starting element pointer         */
 Psearch_dir     dir,                /* search direction                 */
 Pelem_type_list *incl,              /* element incl. list               */
 Pelem_type_list *excl,              /* element excl. list               */
 Pint            *error_ind,         /* OUT error indicator              */
 Psearch_status *status,             /* OUT search status                */
 Pint            *found_el );        /* OUT found element pointer                    */

void pset_edit_mode( /* SET EDIT MODE */
 Pedit_mode   mode );                /* edit mode: PEDIT_INSERT (default) or PEDIT_REPLACE */

void pinq_elem_ptr( /* INQUIRE THE CURRENT ELEMENT POINTER POSITION */
 Pint    *error_ind,                 /* OUT error indicator              */
 Pint    *ep_value );                /* OUT element pointer value */

void pinq_edit_mode( /* INQUIRE THE CURRENT EDIT MODE */
 Pint    *error_ind,                 /* OUT error indicator              */
 Pedit_mode   *edit_mode );  /* OUT edit mode                    */

void pinq_cur_elem_type_size( /* RETURNS THE TYPE OF AN ELEMENT IN THE OPEN STRUCTURE */
Pint    *error_ind,                  /* OUT error indicator              */
 Pelem_type    *type,                /* OUT element type                 */
 size_t  *size );                    /* OUT element size                 */

void pinq_elem_type_size( /* RETURNS THE TYPE OF AN ELEMENT IN A STRUCTURE */
 Pint    struct_id,                  /* structure identifier             */
 Pint    element,                    /* element number                   */
 Pint    *error_ind,                 /* OUT error indicator              */
 Pelem_type *type,                   /* OUT element type                 */
 size_t  *size );                    /* OUT element size                 */

void pinq_cur_elem_content( /* RETURNS THE CONTENTS OF AN ELEMENT IN THE OPEN STRUCTURE */
 Pstore store,                       /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator                          */
 Pelem_data   **data );              /* OUT data record                  */

void pinq_elem_content(/* RETURNS THE CONTENTS OF AN ELEMENT IN A STRUCTURE */
 Pint    struct_id,                  /* structure identifier             */
 Pint    element,                    /* element number                   */
 Pstore store,                       /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator              */
 Pelem_data   **data );              /* OUT data record                  */

void pinq_struct_ids( /* RETURN A LIST OF ALL STRUCTURES IN CSS */
 Pint    length,                     /* length of appl. list             */
 Pint    start,                      /* starting position                */
 Pint    *error_ind,                 /* OUT error indicator              */
 Pint_list *struct_ids,              /* OUT list of structure ids        */
 Pint    *total_length );            /* OUT length of list in PHIGS      */

void pinq_paths_ances( /* RETURNS THE ANCESTORS OF A STRUCTURE */
 Pint    struct_id,                  /* structure identifier             */
 Ppath_order order,                  /* path order    */
 Pint    depth,                      /* path depth   */
 Pstore          store,              /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator              */
```

```
 Pelem_ref_list_list **paths );   /* OUT structure path list        */

void pinq_paths_descs( /* RETURNS THE DESCENDANTS OF A STRUCTURE */
 Pint    struct_id,               /* structure identifier           */
 Ppath_order order,               /* path order   */
 Pint    depth,                   /* path depth   */
 Pstore          store,           /* handle to Store object */
 Pint    *error_ind,              /* OUT error indicator            */
 Pelem_ref_list_list **paths );   /* OUT structure path list        */

void pinq_wss_posted( /* RETURNS A LIST OF WORKSTATIONS A STRUCTURE IS POSTED TO */
 Pint    struct_id,               /* structure identifier           */
 Pint    length,                  /* length of application list     */
 Pint    start,                   /* starting position              */
 Pint    *error_ind,              /* OUT error indicator            */
 Pint_list        *ws,            /* OUT list of workstations       */
 Pint    *total_length );         /* OUT length of list in PHIGS    */
```

Output primitives

This section contains output primitives and their attributes

Polylines

```
void ppolyline( /* CREATE A 2D POLYLINE */
Ppoint_list       *point_list );  /* a list of points */

void ppolyline3( /* CREATE A 3D POLYLINE */
Ppoint_list3 *point_list );       /* list of points */

void ppolyline_set3_data( /* CREATE A 3D POLYLINE  SET WITH COLOUR VERTEX SPECIFICATION */
 Pint            vflag,           /* __vertexdataflags: see PHIGS constants */
 Pint            colour_type,     /* colour type */
 Pint            npl,             /* number of polylines in the set */
 Pline_vdata_list3 *vdata );      /* pointer to line vertex data list */

void pset_linetype( /* SET POLYLINE LINE TYPE */
 Pint    linetype );              /*  __linetype or __linetypeSUN, see PHIGS constants */

void pset_linewidth( /* SET POLYLINE WIDTH */
 Pfloat_arg      width );         /* linewidth scale factor         */

void pset_line_colr_ind( /* SET POLYLINE COLOUR INDEX */
 Pint    colour  );               /* polyline colour index          */

void pset_line_colr( /* SET POLYLINE RGB COLOUR */
 Pgcolr          *colour );       /* polyline colour */

void pset_line_rep( /* SET POLYLINE BUNDLED ATTRIBUTE */
 Pint    ws,                      /* workstation identifier         */
 Pint    index,                   /* polyline bundle index          */
 Pline_bundle *rep );             /* polyline representation pointer                */

void pset_line_ind( /* SET POLYLINE LINE INDEX */
 Pint    index   );               /* polyline bundle index          */

void pset_line_rep_plus( /* SET POLYLINE BUNDLED EXTENDED ATTRIBUTES */
 Pint    ws,                      /* workstation identifier         */
 Pint    index,                   /* polyline bundle index          */
 Pline_bundle_plus *rep );        /* extended polyline representation pointer */

void pset_line_shad_meth( /* CONTROL HOW THE VERTEX COLOURS AFFECT THE LINE */
 Pint    method );                /* __lineshading: PSD_NONE/PSD_COLOUR */

void pinq_line_rep( /* INQUIRE A POLYLINE BUNDLED ATTRIBUTE INDEX */
```

```
Pint     ws,                    /* workstation identifier         */
Pint     index,                 /* polyline index                 */
Pinq_type type,                 /* type of returned value         */
Pint     *error_ind,            /* OUT error indicator            */
Pline_bundle *rep );            /* OUT polyline representation             */

void pinq_line_rep_plus( /* INQUIRE A POLYLINE EXTENDED BUNDLED ATTRIBUTE INDEX */
Pint     ws,                    /* workstation identifier         */
Pint     index,                 /* polyline index                 */
Pinq_type type,                 /* type of returned value         */
Pint    *error_ind,             /* OUT error indicator            */
Pline_bundle_plus *rep );       /* OUT polyline representation*/

void pinq_list_line_inds( /* INQUIRE LIST OF POLYLINE INDICES */
Pint    ws,                     /* workstation identifier         */
Pint    length,                 /* length of application list     */
Pint    start,                  /* starting position              */
Pint    *error_ind,             /* OUT error indicator            */
Pint_list *indices,             /* OUT list of polyline indices   */
Pint    *total_length );        /* OUT length of list in PHIGS    */

void pinq_line_facs( /* INQUIRE AVAILABLE POLYLINE LINETYPES, LINEWIDTHS, PREDEFINE BUNDLES */
Pint    type,                   /* workstation type               */
Pint    length,                 /* length of application list     */
Pint    start,                  /* starting position              */
Pint    *error_ind,             /* OUT error indicator            */
Pline_facs *facilities,         /* OUT polyline facilities        */
Pint    *total_length );        /* OUT length of list in PHIGS    */

void pinq_line_facs_plus( /* INQUIRE AVAILABLE PHIGS+ POLYLINE LINETYPES, SHADING METHODS, AND
                             LINEWIDTH, PREDEFINED BUNDLES */
Pint    type,                   /* workstation type               */
Pint    lt_length,              /* number of line types to return */
Pint    lt_start,               /* starting position              */
Pint    sm_length,              /* number of shading methods to return */
Pint    sm_start,               /* starting position              */
Pint    *error_ind,             /* OUT error indicator            */
Pint    *tot_lt_length,         /* OUT length of line type list in PHIGS            */
Pint    *tot_sm_length,         /* OUT length of shading method list in PHIGS, __lineshading */
Pline_facs_plus *facs );        /* OUT extended polyline facilities            *;

void pinq_pred_line_rep( /* INQUIRE PREDEFINED POLYLINE REPRESENTATION */
Pint    type,                   /* workstation type               */
Pint    index,                  /* predefined index               */
Pint    *error_ind,             /* OUT error indicator            */
Pline_bundle *bundle );         /* OUT predefined polyline rep             */

void pinq_pred_line_rep_plus(/* INQUIRE PREDEFINED EXTENDED POLYLINE REPRESENTATION */
Pint            type,           /* workstation type               */
Pint            index,          /* predefined index               */
Pint            *error_ind,     /* OUT error indicator            */
Pline_bundle_plus *bundle ); /* OUT predefined polyline rep             */
```

Polymarker

```
void ppolymarker3(/* CREATE A 3D POLYMARKER */
Ppoint_list3     *point_list );

void ppolymarker( /* CREATE 2 2D POLYMARKER */
 Ppoint_list     *point_list );

void pset_marker_type( /* SET POLYMARKER TYPE */
Pint    markertype ); /* __markertype or __markertypeSUN  */

void pset_marker_size( /* SET POLYMARKER SCALE FACTOR */
Pfloat_arg      size );         /* markersize scale factor        */
```

```
void pset_marker_colr_ind( /* SET POLYMARKER COLOUR INDICES */
Pint      colour  );                 /* polymarker colour index      */

void pset_marker_colr( /* SET POLYMARKER COLOUR */
 Pgcolr          *colour );        /* polymarker colour */

void pset_marker_ind( /* SET POLYMARKER INDEX */
 Pint    index );                  /* polymarker index              */

void pset_marker_rep(/* SET POLYMARKER REPRESENTATION */
 Pint    ws,                       /* workstation identifier        */
 Pint    index,                    /* polymarker bundle index   */
 Pmarker_bundle *rep );            /* polymarker representation pointer          */

void pset_marker_rep_plus(/* SET EXTENDED POLYMARKER REPRESENTATION */
 Pint    ws,                        /* workstation identifier       */
 Pint    index,                     /* polymarker bundle index   */
 Pmarker_bundle_plus *rep );   /* polymarker representation pointer          */

void pinq_marker_rep(/* INQUIRE POLYMARKER REPRESENTATION */
 Pint     ws,                      /* workstation identifier        */
 Pint     index,                   /* polymarker index              */
 Pinq_type       type,             /* type of returned value        */
 Pint     *error_ind,              /* OUT error indicator           */
 Pmarker_bundle *rep );            /* OUT polymarker representation               */

void pinq_marker_rep_plus(/* INQUIRE EXTENDED POLYMARKER REPRESENTATION */
 Pint     wsid,                    /* workstation identifier         */
 Pint     index,                   /* polymarker index               */
 Pinq_type type,                   /* type of returned value         */
 Pint    *error_ind,               /* OUT error indicator            */
 Pmarker_bundle_plus *rep );  /* OUT polymarker representation */

void pinq_list_marker_inds( /* INQUIRE LIST OF POLYMARKER INDICES */
 Pint    ws,                       /* workstation identifier        */
 Pint    length,                   /* length of application list    */
 Pint    start,                    /* starting position             */
 Pint    *error_ind,               /* OUT error indicator           */
 Pint_list *indices,               /* OUT list of polymarker indices*/
 Pint    *total_length );          /* OUT length of list in PHIGS   */

void pinq_pred_marker_rep( /* INQUIRE PREDEDINED POLYMARKER REPRESENTATION */
 Pint    type,                     /* workstation type              */
 Pint    index,                    /* predefined index              */
 Pint    *error_ind,               /* OUT error indicator           */
 Pmarker_bundle *bundle );    /* OUT predefined polymarker rep              */


void pinq_pred_marker_rep_plus( /* INQUIRE PREDEDINED EXTENDED POLYMARKER REPRESENTATION */
 Pint    type,                     /* workstation type              */
 Pint    index,                    /* predefined index              */
 Pint    *error_ind,               /* OUT error indicator           */
 Pmarker_bundle_plus *bundle );/* OUT predefined polymarker rep          */

void pinq_marker_facs( /* INQUIRE MARKER TYPES, SIZES, PREDEFINED BUNDLES */
 Pint    type,                     /* workstation type              */
 Pint    length,                   /* length of application list    */
 Pint    start,                    /* starting position             */
 Pint    *error_ind,               /* OUT error indicator           */
 Pmarker_facs *facs,               /* OUT polymarker facilities     */
 Pint    *total_length );          /* OUT length of list in PHIGS   */
```

Fill Area

```
void pfill_area3( /* CREATE A 3D FILL AREA */
 Ppoint_list3     *point_list );
```

```
void pfill_area( /* CREATE A 2D FILL AREA */
 Ppoint_list        *point_list );

void pfill_area_set3( /* CREATE A 3D FILL AREA SET */
 Ppoint_list_list3 *point_list_list );                    /* list of point lists */

void pfill_area_set3_data( /* CREATE A 3D FILL AREA SET WITH DATA */
 Pint               fflag,              /* what data is specified per facet, __facetdataflags */
 Pint               eflag,              /* what data is specified per edge, __edgedataflags */
 Pint               vflag,              /* what data is specified per vertex, __vertexdataflags */
 Pint               colour_type,   /* colour type                         */
 Pfacet_data3 *fdata,           /* facet data                          */
 Pint               nfa,                /* number of fill areas in the set */
 Pedge_data_list *edata,       /* edge data list                      */
 Pfacet_vdata_list3 *vdata );  /* facet vertex data list          */

void pfill_area_set( /* CREATE A 2D FILL AREA SET */
 Ppoint_list_list *point_list_list  ); /* list of point_lists */

void pset_of_fill_area_set3_data( /* CREATE A 3D SET OF FILL AREA SET WITH DATA */
 Pint               fflag,              /* data per facet flag, __facetdataflags */
 Pint               eflag,              /* data per edge flag, __edgedataflags */
 Pint               vflag,              /* data per vertex flag, __vertexdataflags */
 Pint               colour_type,   /* colour type */
 Pint               num_sets,        /* number of fill area sets */
 Pfacet_data_arr3 *fdata,      /* facet data */
 Pedge_data_list_list *edata,  /* array of L(L(E)) edge data */
 Pint_list_list      *vlist,            /* array of L(L(I)) vertex indices */
 Pfacet_vdata_list3 n*vdata); /* facet vertex data */

void ptri_strip3_data( /* CREATE A 3D TRIANGLE STRIP WITH DATA */
 Pint    fflag,                          /* what data is specified per facet , __facetdataflags*/
 Pint    vflag,                          /* what data is specified per vertex, __vertexdatafalgs */
 Pint    colour_type,                 /* colour type */
 Pint                nv,                /* number of vertices */
 facet_data_arr3 *fdata,        /* facet data */
 Pfacet_vdata_arr3 *vdata );  /* facet vertex data */

void pquad_mesh3_data( /* CREATE A 3D QUADRILATERAL MESH WITH DATA */
 Pint               fflag,              /* what data is specified per facet, __facetdataflags */
 Pint               vflag,              /* what data is specified per vertex, __vertexdataflags */
 Pint               colour_type,   /* colour type */
 Pint_size         *dim,              /* dimension of vertices */
 Pfacet_data_arr3 *fdata,      /* facet data */
 Pfacet_vdata_arr3 *vdata );  /* facet vertex data */

void pset_int_ind( /* SET INTERIOR INDEX */
  Pint    index    );                   /* interior index */

void pset_edge_ind( /* SET EDGE INDEX */
 Pint    index );                        /* edge index   */

void pset_int_style( /* SET INTERIOR STYLE */
 Pint_style        style );             /* interior style   */

void pset_back_int_style( /* SET BACK INTERIOR STYLE */
 Pint_style        style );             /* back interior style               */

void pset_int_style_ind( /* SET INTERIOR STYLE INDEX */
 Pint    index ); /* interior style index                */

void pset_back_int_style_ind( /* SET BACK INTERIOR STYLE INDEX */
 Pint    index ); /* back interior sytle index        */

void pset_int_colr_ind( /* SET INTERIOR COLOUR INDEX */
 Pint    index ); /* interior colour index             */
```

```
void pset_int_colr( /* SET INTERIOR COLOUR */
 Pgcolr          *colour );       /* interior colour */

void pset_back_int_colr( /* SET BACKFACE INTERIOR COLOUR */
 Pgcolr          *colour );       /* back interior colour */

void pset_edge_flag( /* SET EDGE FLAG */
 Pedge_flag    edge_flag );    /* edge flag     */

void pset_edgetype( /* SET EDGE TYPE */
 Pint    edgetype );              /* edgetype     */

void pset_edgewidth( /* SET EDGE WIDTH SCALE FACTOR */
 Pfloat_arg      scale );          /* edgewidth scale factor      */

void pset_edge_colr_ind(/* SET EDGE COLOUR INDEX */
 Pint    index );                  /* edge colour index             */

void pset_edge_colr( /* SET EDGE COLOUR */
 Pgcolr          *colour );       /* edge colour */

void pset_pat_size( /* SET PATTERN SIZE */
 const Pfloat_size *pat_size );

void pset_pat_ref_point_vecs( /* SET PATTERN REFERENCE POINT VECTORS */
 Ppoint3         *ref_pt,         /* pattern reference point      */
 Pvec3pat_ref_vec(2) );         /* direction vector                */

void pset_pat_ref_point( /* SET PATTERN REFERENCE POINT */
 Ppoint          *ref_pt          ); /* pattern reference point   */

void pset_int_rep( /* SET INTERIOR REPRESENTATION */
 Pint    ws,                        /* workstation identifier */
 Pint    index,                     /* interior bundle index */
 Pint_bundle *rep );              /* interior representation pointer */

void pset_edge_rep( /* SET EDGE REPRESENTATION */
 Pint    ws,                        /* workstation identifier         */
 Pint    index,                     /* edge bundle index              */
 Pedge_bundle *rep );            /* edge representation pointer                    */

void pset_edge_rep_plus( /* SET EXTENDED EDGE REPRESENTATION */
 Pint              ws,               /* workstation identifier                         */
 Pint              index,            /* edge bundle index                              */
 Pedge_bundle_plus *rep );   /* edge representation pointer                    */

void pset_pat_rep( /* SET PATTERN REPRESENTATION */
 Pint    ws,                        /* workstation identifier                         */
 Pint    index,                     /* pattern bundle index                           */
 Ppat_rep *rep );                 /* pattern representation pointer                 */

void pset_pat_rep_plus( /* SET EXTENDED PATTERN REPRESENTATION */
 Pint              ws,               /* workstation identifier                         */
 Pint              index,            /* pattern bundle index                           */
 Ppat_rep_plus*rep );            /* pattern representation pointer                 */

void pset_int_shad_meth( /* SET INTERIOR SHADING METHOD */
 Pint    method );                  /* __intershading, see PHIGS constants           */

void pset_back_int_shad_meth( /* SET BACK INTERIOR SHADING METHOD */
 Pint    method );                  /* __intershading, see PHIGS constants */

void pset_int_rep_plus( /* SET EXTENDED INTERIOR REPRESENTATION */
 Pint              wsid,             /* workstation id */
 Pint              index,            /* interior index */
 Pint_bundle_plus *rep );        /* representation */
```

```
void pinq_int_rep( /* INQUIRE INTERIOR REPRESENTATION */
 Pint          ws,           /* workstation identifier      */
 Pint          index,        /* interior index*/
 Pinq_type     type,         /* type of returned value      */
 Pint          *error_ind,   /* OUT error indicator         */
 Pint_bundle   *rep );       /* OUT interior representation */

void pinq_int_rep_plus( /* INQUIRE EXTENDED INTERIOR REPRESENTATION */
 Pint          ws,           /* workstation identifier      */
 Pint          index,        /* interior index*/
 Pinq_type      type,        /* type of returned value      */
 Pint          *error_ind,   /* OUT error indicator         */
 Pint_bundle_plus *rep );    /* OUT interior representation */

void pinq_edge_rep( /* INQUIRE EDGE REPRESENTATION */
 Pint          ws,           /* workstation identifier      */
 Pint          index,        /* edge index  */
 Pinq_type      type,        /* type of returned value      */
 Pint          *error_ind,   /* OUT error indicator         */
 Pedge_bundle *rep );        /* OUT edge representation   */

void pinq_edge_rep_plus( /* INQUIRE EXTENDED EDGE REPRESENTATION */
 Pint          ws,           /* workstation identifier      */
 Pint          index,        /* edge index  */
 Pinq_type      type,        /* type of returned value      */
 Pint          *error_ind,   /* OUT error indicator         */
 Pedge_bundle_plus *rep );   /* OUT edge representation   */

void pinq_pat_rep( /* INQUIRE PATTERN REPRESENTATION */
 Pint           ws,          /* workstation identifier      */
 Pint           index,       /* pattern index               */
 Pinq_type      type,        /* type of returned value      */
 Pstore         store,       /* handle to Store object */
 Pint          *error_ind,   /* OUT error indicator         */
 Ppat_rep      **rep );      /* OUT pattern representation */

void pinq_pat_rep_plus( /* INQUIRE EXTENDED PATTERN REPRESENTATION */
 Pint           ws,          /* workstation identifier      */
 Pint           index,       /* pattern index               */
 Pinq_type      type,        /* type of returned value      */
 Pstore         store,       /* OUT pointer to buffer       */
 Pint          *error_ind,   /* OUT error indicator         */
 Ppat_rep_plus**rep );       /* OUT pattern representation*/

void pinq_list_int_inds( /* INQUIRE LIST OF INTERIOR INDICES */
 Pint          ws,           /* workstation identifier      */
 Pint          length,       /* length of application list  */
 Pint          start,        /* starting position           */
 Pint          *error_ind,   /* OUT error indicator         */
 Pint_list     *indices,     /* OUT list of interior indices  */
 Pint          *total_length );/* OUT length of list in PHIGS  */

void pinq_list_edge_inds( /* INQUIRE LIST OF EDGE INDICES */
 Pint          ws,           /* workstation identifier      */
 Pint          length,       /* length of application list  */
 Pint          start,        /* starting position           */
 Pint          *error_ind,   /* OUT error indicator         */
 Pint_list     *indices,     /* OUT list of edge indices    */
 Pint          *total_length );/* OUT length of list in PHIGS  */

void pinq_list_pat_inds( /* INQUIRE LIST OF PATTERN INDICES */
 Pint          ws,           /* workstation identifier      */
 Pint          length,       /* length of application list  */
 Pint          start,        /* starting position           */
 Pint          *error_ind,   /* OUT error indicator         */
 Pint_list     *indices,     /* OUT list of pattern indices  */
 Pint          *total_length );/* OUT length of list in PHIGS  */
```

```
void pinq_int_facs( /* INQUIRE INTERIOR FACILITIES */
 Pint           type,         /* workstation type            */
 Pint           h_len,        /* length of hatch style list  */
 Pint           h_st,         /* starting position           */
 Pint           *error_ind,   /* OUT error indicator         */
 Pint_facs      *facil,       /* OUT interior facilities     */
 Pint           *tot_h_len ); /* OUT len of hatch list in PHIGS              */

void pinq_int_facs_plus( /* INQUIRE EXTENDED INTERIOR FACILITIES */
 Pint   type,                 /* workstation type                            */
 Pint   is_length,            /* length of application's interior style list */
 Pint   is_start,             /* starting position                           */
 Pint   hs_length,            /* length of application's hatch style list    */
 Pint   hs_start,             /* starting position                           */
 Pint   re_length,            /* length of application's reflectance equation list */
 Pint   re_start,             /* starting position                           */
 Pint   sh_length,            /* length of application's shading method list */
 Pint   sh_start,             /* starting position                           */
 Pint   *error_ind,           /* OUT error indicator                         */
 Pint_facs_plus *facil,       /* extended interior facilities                */
 Pint   *tot_is_length,       /* total length of interior styles list        */
 Pint   *tot_hs_length,       /* total length of hatch styles list           */
 Pint   *tot_re_length,       /* total length of reflectance equation list */
 Pint   *tot_sh_length );     /* total length of shading method list         */

void pinq_pred_int_rep( /* INQUIRE PREDEFINED INTERIOR REPRESENTATION */
 Pint           type,         /* workstation type            */
 Pint           index,        /* predefined index            */
 Pint           *error_ind,   /* OUT error indicator         */
 Pint_bundle    *bundle );    /* OUT predefined interior rep*/

void pinq_pred_int_rep_plus( /* INQUIRE PREDEFINED EXTENDED INTERIOR REPRESENTATION */
 Pint           type,         /* workstation type            */
 Pint           index,        /* interior index              */
 Pint           *error_ind,   /* OUT error indicator         */
 Pint_bundle_plus *rep );     /* OUT extended interior rep*/

void pinq_edge_facs( /* INQUIRE NO OF AVAILABLE EDGETYPES, EDGEWIDTHS, NO OF PREDEFINED EDGE BUNDLES */
 Pint           type,         /* workstation type            */
 Pint           length,       /* length of application list  */
 Pint           start,        /* starting position           */
 Pint           *error_ind,   /* OUT error indicator         */
 Pedge_facs     *facilities,  /* OUT edge facilities         */
 Pint           *total_length );/* OUT length of list in PHIGS   */

void pinq_pred_edge_rep( /* INQUIRE PREDEFINED EDGE REPRESENTATION */
 Pint           type,         /* workstation type            */
 Pint           index,        /* predefined index            */
 Pint           *error_ind,   /* OUT error indicator         */
 Pedge_bundle *bundle );      /* OUT predefined edge rep  */

void pinq_pred_edge_rep_plus( /* INQUIRE PREDEFINED EXTENDED EDGE REPRESENTATION */
 Pint           type,         /* workstation type            */
 Pint           index,        /* predefined index            */
 Pint           *error_ind,   /* OUT error indicator         */
 Pedge_bundle_plus  *bundle ); /* OUT predefined edge rep */

void pinq_pat_facs( /* INQUIRE PATTERN FACILITIES */
 Pint   type,                 /* workstation type                            */
 Pint   *error_ind,           /* OUT error indicator                         */
 Pint   *predefined );        /* OUT number of predefined pattern indices  */

void pinq_pred_pat_rep( /* INQUIRE PREDEFINED PATTERN REPRESENTATION */
 Pint           type,         /* workstation type            */
 Pint           index,        /* predefined index            */
 Pstore         store,        /* handle to Store object      */
 Pint           *error_ind,   /* OUT error indicator         */
```

```
Ppat_rep        **rep  );          /* OUT predefined pattern rep */

void pinq_pred_pat_rep_plus( /* INQUIRE PREDEFINED EXTENDED PATTERN REPRESENTATION */
 Pint           type,              /* workstation type              */
 Pint           index,             /* predefined index              */
 Pstore         store,             /* handle to Store object        */
 Pint           *error_ind,        /* OUT error indicator           */
 Ppat_rep_plus**rep );             /* OUT predefined pattern rep */
```

Cell array

```
void pcell_array( /* CREATE A 2D CELL ARRAY */
 Prect          *rectangle,        /* cell rectangle                */
 Ppat_rep        *colr_array );    /* colour array                  */

void pcell_array3( /* CREATE A 3D CELL ARRAY */
 Pparal         *parallelogram,/* cell parallelogram: (0)=P; (1)=Q; (2)=R    */
 Ppat_rep       *colr_array );    /* colour array                              */

void pcell_array3_plus( /* CREATE 2 3D CELL ARRAY WITH COLOUR SPECIFICATIONS */
 Pparal         *paral,            /* cell parallelogram: (0)=P; (1)=Q; (2)=R    */
 Pint_size      *dim,              /* dimension of cell: DX and DY               */
 Pint           colour_type,       /* colour type                                */
 Pcoval         *colrs );          /* colour array                               */
```

Text

```
void ptext3( /* CREATE 3D TEXT */
 Ppoint3        *text_pt,          /* position of text              */
 const Pvec3    dir(2),            /* directon vector               */
 char           *text );           /* text string                   */

void ptext( /* CREATE TEXT */
 Ppoint         *text_pt,          /* position of text              */
 char   *text    );                /* text string                   */

void pset_text_font( /* SET TEXT FONT */
 Pint    font );                   /* __font, see PHIGS constants */

void pset_text_prec( /* SET TEXT PRECISION */
Ptext_prec      precision );       /* text precision */

void pset_char_expan( /* SET CHARACTER EXPANSION FACTOR */
Pfloat_arg      exp_factor );      /* character expansion factor                 */

void pset_char_space( /* SET TEXT CHARACTER SPACING */
 Pfloat_arg     spacing );         /* character spacing             */

void pset_text_colr_ind( /* SET TEXT COLOUR INDEX */
 Pint    colour   );               /* text colour index             */

void pset_text_colr( /* SET TEXT COLOUR */
Pgcolr *colour );                  /* text colour    */

void pset_char_ht( /* SET TEXT CHARACTER HEIGHT */
 Pfloat_arg     height );          /* character height              */

void pset_char_up_vec( /* SET CHARACTER UP VECTOR STRUCTURE */
 Pvec  *up_vect );                 /* character up vector           */

void pset_text_path( /* SET TEXT PATH */
 Ptext_path     path );            /* text path: PPATH_xx where xx is one of RIGHT, LEFT, UP, or DOWN */

void pset_text_align( /* SET TEXT HORIZONTAL AND VERTICAL ALIGNMENT */
 Ptext_align    *text_align );     /* text alignment                */
```

```
void pset_text_ind( /* SET TEXT INDEX (BUNDLED TEXT ATTRIBUTES) */
 Pint    index );                      /* text index     */

void pset_text_rep( /* CREATE A TEXT BUNDLE */
 Pint    ws,                           /* workstation identifier          */
 Pint    index,                        /* text bundle index               */
 Ptext_bundle  *rep );                 /* text representation pointer */

void pset_text_rep_plus( /* CREATE A PHIGS+ TEXT BUNDLE */
 Pint    ws,                           /* workstation identifier          */
 Pint    index,                        /* text bundle index               */
 Ptext_bundle_plus *rep );             /* text representation pointer */

void pinq_text_extent( /* INQUIRE TEXT EXTENT RECTANGLE */
 Pint              wst,                /* workstation type                */
 Pint              font,               /* __font, see PHIGS const.        */
 Pfloat_arg        exp,                /* char expansion factor           */
 Pfloat_arg        sp,                 /* char spacing                    */
 Pfloat_arg        ht,                 /* char height  */
 Ptext_path        path,               /* text path      */
 Phor_text_align hor,                  /* horizontal alignment            */
 Pvert_text_align ver,                 /* vertical alignment              */
 char              *str,               /* text string      */
 Pint              *error_ind,         /* OUT error indicator             */
 Prect             *rect,              /* OUT extent rectangle            */
 Ppoint            *offset );          /* OUT concatenation offset  */

void panno_text_rel3( /* CREATE 3D ANNOTATION TEXT */
 Ppoint3 *ref_point,                   /* reference point                 */
 Pvec3*anno_offset,                    /* annotation offset               */
 char   *text );                       /* annotation text string          */

void panno_text_rel( /* CREATE 2D ANNOTATION TEXT */
 Ppoint *ref_point,                    /* reference point                 */
 Pvec  *anno_offset,                   /* annotation offset               */
 char   *text );                       /* annotation text string          */

void pset_anno_char_ht( /* SET ANNOTATION TEXT CHARACTER HEIGHT */
 Pfloat_arg        height );           /* character height                */

void pset_anno_char_up_vec( /* SET ANNOTATION TEXT CHARACTER UP VECTOR STRUCTURE */
 Pvec              *up_vect );         /* character up vector             */

void pset_anno_path( /* SET ANNOTATION TEXT PATH */
 Ptext_path        path );             /* text path: PPATH_xx where xx is one of RIGHT, LEFT, UP, or DOWN */

void pset_anno_align( /* SET ANNOTATION TEXT HORIZONTAL AND VERTICAL ALIGNMENT */
 Ptext_align       *text_align );      /* text alignment                  */

void pset_anno_style( /* SET ANNOTATION STYLE */
 Pint    style );  /* __annostyles: PANNO_STYLE_UNCONNECTED or PANNO_STYLE_LEAD_LINE */

void pinq_list_text_inds(/* INQUIRE LIST OF TEXT INDICES */
 Pint              ws,                 /* workstation identifier          */
 Pint              length,             /* length of application list      */
 Pint              start,              /* starting position               */
 Pint              *error_ind,         /* OUT error indicator             */
 Pint_list         *indices,           /* OUT list of text indices        */
 Pint              *total_length );/* OUT length of list in PHIGS   */

void pinq_text_rep(/* INQUIRE TEXT REPRESENTATION */
 Pint              ws,                 /* workstation identifier          */
 Pint              index,              /* text index     */
 Pinq_type         type,               /* type of returned value          */
 Pint              *error_ind,         /* OUT error indicator             */
 Ptext_bundle  *rep );                 /* OUT text representation         */
```

```
void pinq_text_rep_plus(/* INQUIRE EXTENDED TEXT REPRESENTATION */
 Pint             ws,              /* workstation identifier       */
 Pint             index,           /* text index    */
 Pinq_type        type,            /* type of returned value       */
 Pint             *error_ind,      /* OUT error indicator          */
 Ptext_bundle_plus *rep );         /* OUT extended text representation */

void pinq_pred_text_rep( /* INQUIRE PREDEFINED TEXT REPRESENTATION */
 Pint             type,            /* workstation type             */
 Pint             index,           /* predefined index             */
 Pint             *error_ind,      /* OUT error indicator          */
 Ptext_bundle     *bundle );       /* OUT predefined text rep      */

void pinq_pred_text_rep_plus( /* INQUIRE PREDEFINED EXTENDED TEXT REPRESENTATION */
 Pint             type,            /* workstation type             */
 Pint             index,           /* predefined index             */
 Pint             *error_ind,      /* OUT error indicator          */
 Ptext_bundle_plus *bundle ); /* OUT predefined text rep      */

void pinq_anno_facs( /* INQUIRE AVAILABLE ANNOTATION STYLES AND ANNOTATION TEXT CHAR HEIGHTS */
 Pint             type,            /* workstation type             */
 Pint             length,          /* length of application list   */
 Pint             start,           /* starting position            */
 Pint             *error_ind,      /* OUT error indicator          */
 Pint_list        *styles,         /* OUT list of __annostyles     */
 Pint             *total_length,   /* OUT length of list in PHIGS  */
 Pint             *num_char_hts,/* OUT number of character heights */
 Pfloat           *min_char_ht,
 Pfloat           *max_char_ht );

void pinq_text_facs( /* INQUIRE AVAILABLE TEXT FONTS AND PRECISION, ETC. */
 Pint             type,            /* workstation type             */
 Pint             length,          /* length of application list   */
 Pint             start,           /* starting position            */
 Pint             *error_ind,      /* OUT error indicator          */
 Ptext_facs       *facilities,     /* OUT text facilities          */
 Pint             *total_length );/* OUT length of list in PHIGS  */
```

Generalized Drawing Primitive

```
void pgdp3( /* CREATE A 3D GENERALIZED DRAWING PRIMITIVE */
 Ppoint_list3     *point_list,
 Pint             gdp3_id,         /* gdp function identifier      */
 Pgdp_data3       *gdp_data );     /* data record pointer          */

void pgdp( /* CREATE A 2D GENERALIZED DRAWING PRIMITIVE */
 Ppoint_list      *point_list,     /* list of points */
 Pint             gdp_id,          /* gdp function identifier      */
 Pgdp_data        *gdp_data );     /* data record pointer          */

void pinq_list_avail_gdp3( /* INQUIRE LIST OF AVAILABLE WORKSTATION-DEPENDENT 3D GDP's */
 Pint    type,                     /* workstation type             */
Pint     length,                   /* length of application list   */
Pint     start,                    /* starting position            */
Pint     *error_ind,               /* OUT error indicator          */
Pint_list *gdps,                   /* OUT list of GDPs             */
Pint     *total_length );          /* OUT length of list in PHIGS  */

void pinq_list_avail_gdp( /* INQUIRE LIST OF AVAILABLE WORKSTATION-DEPENDENT 2D GDP's */
Pint     type,                     /* workstation type             */
Pint     length,                   /* length of application list   */
Pint     start,                    /* starting position            */
Pint     *error_ind,               /* OUT error indicator          */
Pint_list *gdps,                   /* OUT list of GDPs             */
Pint     *total_length );          /* OUT length of list in PHIGS  */

void pinq_gdp( /* INQUIRE THE ATTRIBUTES USED BY A 2D GDP */
```

```
 Pint    ws_type,                /* workstation type            */
 Pint    gdp,                    /* GDP function number         */
 Pint    *error_ind,             /* OUT error indicator         */
 Pint    *num_attrs,             /* OUT number of attributes */
 Pattrs attrs(5)  );             /* OUT list of attributes */

void pinq_gdp3( /* INQUIRE THE ATTRIBUTES USED BY A 3D GDP */
 Pint    ws_type,                /* workstation type            */
 Pint    gdp,                    /* GDP function number         */
 Pint    *error_ind,             /* OUT error indicator         */
 Pint    *num_attrs,             /* OUT number of attributes */
 Pattrs attrs(5) );              /* OUT list of attributes */
```

Bundled attributes

```
void pset_indiv_asf( /* CREATE AN ASPECT SOURCE FLAG ELEMENT */
 Paspect att_id,                 /* attribute identifier: PASPECT_xxx              */
 Pasf    att_source );           /* attribute source: PASF_BUNDLED or PASF_INDIV  */
```

Colour maps, indices, direct colour

```
void pset_colr_rep( /* CREATE AND SET A COLOUR TABLE ENTRY */
 Pint    ws,                     /* workstation identifier */
 Pint    index,                  /* colour bundle index         */
 Pcolr_rep *rep );               /* colour representation pointer */

void pset_colr_model( /* SET CURRENT COLOUR MODEL */
 Pint    ws,                     /* workstation identifier */
 Pint    model );                /* __clmodel: PMODEL_xx where xx IS ONE OF RGB, CIELUV, HLS, or HSV */

void pinq_list_colr_inds( /* INQUIRE LIST OF COLOUR INDICES */
 Pint    ws,                     /* workstation identifier      */
 Pint    length,                 /* length of application list  */
 Pint    start,                  /* starting position           */
 Pint    *error_ind,             /* OUT error indicator         */
 Pint_list *indices,             /* OUT list of colour indices  */
 Pint    *total_length );        /* OUT length of list in PHIGS */

void pinq_colr_model_facs( /* INQUIRE COLOUR MODEL FACILITIES */
Pint     type,                   /* workstation type            */
Pint     length,                 /* length of application list  */
Pint     start,                  /* starting position           */
Pint     *error_ind,             /* OUT error indicator         */
Pint_list *models,               /* OUT list of __colmodel      */
Pint     *total_length,          /* OUT length of list in PHIGS */
Pint     *def );                 /* OUT default colour model  */

void pinq_direct_colr_model_facs( /* INQUIRE WHICH DIRECT COLOUR MODELS ARE AVAILABLE */
Pint     type,                   /* workstation type            */
Pint     length,                 /* length of application list  */
Pint     start,                  /* starting position           */
Pint     *error_ind,             /* OUT error indicator         */
Pint     *total_length,          /* OUT length of list in PHIGS */
Pint_list *models );             /* OUT list of colour models   */

void pinq_colr_facs( /* INQUIRE COLOUR FACILITIES (MONO or B/W, NO OF COLOURS, ETC.) */
Pint     type,                   /* workstation type            */
Pint     *error_ind,             /* OUT error indicator         */
Pcolr_facs      *facilities );   /* OUT colour facilities       */

void pinq_pred_colr_rep( /* INQUIRE PREDEFINED COLOUR REPRESENTATION */
 Pint          type,             /* workstation type            */
 Pint          index,            /* predefined index            */
 Pint          *error_ind,       /* OUT error indicator         */
 Pcolr_rep     *bundle );        /* OUT predefined colour rep */
```

```
void pinq_colr_rep( /* INQUIRE A SPECIFIC COLOUR IN A COLOUR TABLE */
 Pint     ws,                    /* workstation identifier       */
 Pint     index,                 /* colour index */
 Pinq_type type,                 /* type of returned value       */
 Pint     *error_ind,            /* OUT error indicator          */
 Pcolr_rep *rep );               /* OUT colour representation  */

void pset_colr_map_ind( /* SELECT A COLOUR MAP INDEX */
 Pint    index );                /* colour mapping index         */

void pset_colr_map_rep( /* DEFINE A COLOUR MAP ENTRY */
 Pint             ws,            /* workstation identifier       */
 Pint             index,         /* colour mapping index         */
 Pint             map_method, /* __colmapmethodd               */
 Pcolr_map_data *map_data ); /* data record                    */

void pinq_list_colr_map_inds( /* INQUIRE A LIST OF COLOUR INDICES CURRENTLY DEFINED */
 Pint    ws,                     /* workstation identifier       */
 Pint    length,                 /* length of application list   */
 Pint    start,                  /* starting position            */
 Pint    *error_ind,             /* OUT error indicator          */
 Pint_list *indices,             /* OUT list of view indices     */
 Pint    *total_length );        /* OUT length of list in PHIGS  */

void pinq_colr_map_rep( /* INQUIRE THE CONTENTS OF A COLOUR MAPPING REPRESENTATION */
 Pint             ws,            /* workstation identifier       */
 Pint             index,         /*colour mapping index          */
 Pinq_type        type,          /* type of returned value       */
 Pstore           store,         /* store object                 */
 Pint             *error_ind,    /* OUT error indicator          */
 Pint             *map_method,/* OUT: __colmapmethod            */
 Pcolr_map_data **map_data ); /* OUT: data record pointer */

void pinq_colr_map_facs( /* INQUIRE NO COLOUR MAPPING METHODS, NO OF PREDEFINED COL MAP ENTRIES */
 Pint             type,          /* workstation type             */
 Pint             length,        /* length of application list   */
 Pint             start,         /* starting position            */
 Pint             *error_ind,    /* OUT error indicator          */
 Pcolr_map_facs *facilities,     /* OUT colour mapping facilities */
 Pint             *total_length );/* OUT length of list in PHIGS  */

void pinq_pred_colr_map_rep( /* INQUIRE PREDEFINED COLOUR MAP ENTRIES */
 Pint             ws_type,       /* IN                           */
 Pint             index,         /* representation index         */
 Pstore           store,         /* store object                 */
 Pint             *error_ind,    /* OUT error indicator          */
 Pint             *map_method,/* OUT: __colmapmethod            */
 Pcolr_map_data**map_data ); /* OUT: data record pointer */

void pinq_colr_map_method_facs( /*INQUIRE NO OF PCOLR_MAP_TRUE, PCOLR_MAP_PSEUDO COL'S AVAILABLE*/
 Pint             type,          /* workstation type             */
 Pint             map_method, /* __colmapmethod                 */
 Pint             *error_ind,    /* OUT error indicator          */
 Pcolr_map_st *map_st );         /* OUT mapping state            */

void pinq_colr_map_st( /* INQUIRE INFO ABOUT RESOURCES CURRENTLY AVAILABLE FOR COLOUR MAPPING */
 Pint             ws,            /* workstation id               */
 Pint             map_method, /* __colmapmethod                 */
 Pint             *error_ind,    /* OUT error indicator          */
 Pcolr_map_st *map_st );         /* OUT mapping state            */

void pinq_colr_model( /* INQUIRE THE CURRENT COLOUR MODEL */
 Pint    ws,                     /* workstation identifier       */
 Pint    *error_ind,             /* OUT error indicator          */
 Pint    *model  );              /* OUT current colour model  */
```

Hidden lines, hidden surface removal

```
void pset_hlhsr_id( /* SET HLHSR IDENTIFIER - THIS IS IGNORED !! */
 Pint    id );       /*  __hlhsrid: PHIGS_HLHSR_ID_OFF or PHIGS_HLHSR_ID_ON */

void pset_hlhsr_mode( /* SET HLHSR MODE */
 Pint    ws,        /* workstation id                    */
 Pint    mode); /* __hlhsrmode, see PHIGS constants */

void pinq_hlhsr_mode( /* INQUIRE THE CURRENT HLHSR MODE */
 Pint    ws,                      /* workstation identifier        */
 Pint    *error_ind,              /* OUT error indicator           */
 Pupd_st *state,                  /* OUT HLHSR update state        */
 Pint    *cur_mode,               /* OUT current HLHSR mode        */
 Pint    *req_mode );             /* OUT requested HLHSR mode*/

void pinq_hlhsr_id_facs( /* INQUIRE SUPPORTED HLHSR IDENTIFIERS */
Pint     type,                    /* workstation type              */
 Pint    length,                  /* length of id list             */
 Pint    start,                   /* starting position of id list  */
 Pint    *error_ind,              /* OUT error indicator           */
 Pint_list *ids,                  /* OUT list of HLHSR ids         */
 Pint    *length_list );          /* OUT len of id list in PHIGS   */

void pinq_hlhsr_mode_facs( /* INQUIRE AVAILABLE HLHSR MODES */
 Pint    type,                    /* workstation type              */
 Pint    length,                  /* length of id list             */
 Pint    start,                   /* starting position of id list  */
 Pint    *error_ind,              /* OUT error indicator           */
 Pint_list *modes,                /* OUT list of HLHSR modes       */
 Pint    *length_list );          /* OUT len of id list in PHIGS   */
```

Rendering

```
void pcomp_fill_area_set_gnorm( /* COMPUTE FILL AREA SET GEOMETRY NORMAL */
 Pint            vflag,
 Pint            nfa,
 Pfacet_vdata_list3 *vdata,
 Pint            *error_ind,
 Pvec3           *normal );

void pset_refl_props( /* SET AREA PROPERTIES */
 Prefl_props     *properties );   /* surface area properties */

void pset_back_refl_props( /* SET BACK AREA PROPERTIES */
 Prefl_props     *properties );   /* back surface area properties */

void pset_refl_eqn( /* SET INTERIOR REFLECTANCE EQUATION */
 Pint    equation );              /* reflectance equation, __reflequation */

void pset_back_refl_eqn( /* SET BACK INTERIOR REFLECTANCE EQUATION */
 Pint    equation );              /* back reflectance equation, __reflequation */

void pset_face_disting_mode( /* SET FACE DISTINGUISHING MODE */
 Pdisting_mode mode );            /* distinguishing mode */

void pset_face_cull_mode( /* SET FACE CULLING MODE */
 Pcull_mode    mode );            /* culling mode */

void pset_light_src_state( /* SET LIGHT SOURCE STATE */
 Pint_list *activation,           /* activation list */
 Pint_list *deactivation );       /* deactivation list */

void pset_light_src_rep( /* SET LIGHT SOURCE REPRESENTATION */
 Pint            ws,              /* workstation identifier */
 Pint            index,           /* light table index */
 Plight_src_bundle *rep );        /* light source representation */
```

```
void pinq_light_src_rep( /* INQUIRE LIGHT SOURCE REPRESENTATION */
 Pint             ws,             /* workstation identifier      */
 Pint             index,          /* light source index          */
 Pinq_type        type,           /* type of returned value      */
 Pint             *error_ind,     /* OUT error indicator         */
 Plight_src_bundle *rep );        /* OUT depth cue representation           */

void pinq_list_light_src_inds(/* INQUIRE LIST OF LIGHT SOURCE INDICES */
 Pint   ws,                       /* workstation identifier      */
 Pint   length,                   /* length of application list  */
 Pint   start,                    /* starting position           */
 Pint   *error_ind,               /* OUT error indicator         */
 Pint_list *indices,              /* OUT list of polyline indices */
 Pint   *total_length );          /* OUT length of list in PHIGS  */

void pset_dcue_ind( /* SET DEPTH CUE INDEX */
 Pint   index );                  /* depth cue index             */

void pset_dcue_rep( /* SET DEPTH CUE REPRESENTATION */
 Pint   ws,                       /* workstation identifier      */
 Pint   index,                    /* depth cue bundle index      */
 Pdcue_bundle *rep );             /* depth cue representation pointer       */

void pinq_dcue_rep( /* INQUIRE DEPTH CUE REPRESENTATION */
 Pint             ws,             /* workstation identifier      */
 Pint             index,          /* polyline index              */
 Pinq_type        type,           /* type of returned value      */
 Pint             *error_ind,     /* OUT error indicator         */
 Pdcue_bundle *rep );             /* OUT depth cue representation           */

void pinq_dcue_facs( /* INQUIRE DEPTH CUE FACILITIES: NO OF PREDEFINED DEPTH CUE TABLE ENTRIES */
Pint              type,           /* workstation type            */
Pint              *error_ind,     /* OUT error indicator         */
Pint              *facilities );  /* OUT depth cue facilities    *;

void pinq_list_dcue_inds( /* INQUIRE LIST OF DEPTH CUE INDICES */
 Pint             ws,             /*workstation identifier       */
 Pint             length,         /* length of application list  */
 Pint             start,          /* starting position           */
 Pint             *error_ind,     /* OUT error indicator         */
 Pint_list        *indices,       /* OUT list of polyline indices */
 Pint             *total_length );/* OUT length of list in PHIGS  */

void pinq_pred_dcue_rep( /* INQUIRE PREDEFINED DEPTH CUE REPRESENTATION */
 Pint             type,           /* workstation type            */
 Pint             index,          /* predefined index            */
 Pint             *error_ind,     /* OUT error indicator         */
 Pdcue_bundle*bundle );           /* OUT predefined depth cue rep */

void pinq_light_src_facs( /* INQUIRE LIGHT SOURCE FACILITIES */
 Pint             type,           /* workstation type            */
 Pint             length,         /* length of application list  */
 Pint             start,          /* starting position           */
 Pint             *error_ind,     /* OUT error indicator         */
 Plight_src_facs *facilities,     /* OUT light source facilities */
 Pint             *total_length );/* OUT length of list in PHIGS  */

void pinq_pred_light_src_rep( /* INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION */
 Pint             type,           /* workstation type            */
 Pint             index,          /* predefined index            */
 Pint             *error_ind,     /* OUT error indicator         */
 Plight_src_bundle *bundle );     /* OUT predefined light source rep */

void pset_rendering_colr_model(
 Pint   colour_model );           /* __renderingcolmodel         */

void pinq_rendering_colr_model_facs( /* INQUIRE WHICH COLOUR MODELS CAN BE STORED */
```

```
Pint           type,         /* workstation type            */
Pint           length,       /* length of application's list */
Pint           start,        /* starting position           */
Pint           *error_ind,    /* OUT: error indidator        */
Pint_list      *models,       /* OUT: __renderingcolmodel */
Pint     *total_length );    /* OUT: total number of rendering colour models available *
```

Curves and surfaces

```
void pnuni_bsp_curv( /* NON-UNIFORM B-SPLINE CURVE */
Pint           order,        /* spline order */
Pfloat_list    *knots,       /* list of knots */
Prational      rationality,
Ppoint_list34  *cpoints,     /* list of 3D or 4D control points */
Pfloat_arg     min,
Pfloat_arg     max );        /* parameter range */

void pset_curve_approx( /* SET CURVE APPROXIMATION CRITERIA */
Pint           type,         /* __curveapproximation */
Pfloat_arg     value );

void pnuni_bsp_surf( /* NON-UNIFORM B-SPLINE SURFACE */
Pint           uorder,       /* U spline order */
Pint           vorder,       /* V spline order */
Pfloat_list    *uknots,      /* U knots */
Pfloat_list    *vknots,      /* V knots */
Prational      rationality,  /* rationality selector */
Ppoint_grid34 *grid,         /* grid of 3D or 4D control points */
Pint           nloops,       /* number of trim curve loops */
Ptrimcurve_list *tloops );   /* trim curve loops */

void pset_surf_approx( /* SET SURFACE APPROXIMATION CRITERIA */
Pint   type,                 /* __surfaceapproximation */
Pfloat_arg     uvalue,
Pfloat_arg     vvalue );

void pinq_curv_surf_facs( /* INQUIRE CURVE & SURFACE FACILITIES */
Pint   type,                 /* workstation type                      */
Pint   cat_len,              /* length of curve approx types list     */
Pint   cat_st,               /* starting position                     */
Pint   sat_len,              /* length of surface approx types list   */
Pint   sat_st,               /* starting position                     */
Pint   tcat_len,             /* length of trim curve approx types list */
Pint   tcat_st,              /* starting position                     */
Pint   psc_len,              /* length of parametric surface characteristics list */
Pint   psc_st,               /* starting position                     */
Pint   *error_ind,           /* OUT: error indidator                  */
Pcurvsurf_facs*facil,        /* OUT: curve and surface facilities     */
Pint   *tot_cat_len,         /* OUT: total length of curve approx types list */
Pint   *tot_sat_len,         /* OUT: total length of surface approx types list */
Pint   *tot_tcat_len,        /* OUT: total length of trim curve approx types list */
Pint *tot_psc_len );         /* OUT: total length of parametric surface characteristics list */

void pset_para_surf_characs( /* SET PARAMETRIC SURFACE CHARACTERISTICS */
Pint   psc_type,              /* __surfacetypes, see PHIGS constants */
Ppara_surf_characs *data );  /* data record */
```

Transformation and Viewing

```
void pset_local_tran3( /* SET A LOCAL 3D TRANSFORMATION MATRIX */
Pmatrix3       xform,        /* transformation matrix       */
Pcompose_type compose_type ); /*  PTYPE_xx where xx is one of PRECONCAT, POSTCONCAT, or REPLACE */

void pset_local_tran( /* SET A LOCAL 2D TRANSFORMATION MATRIX */
Pmatrix        xform,        /* transformation matrix       */
Pcompose_type compose_type ); /* PTYPE_xx where xx is one of PRECONCAT, POSTCONCAT, or REPLACE */
```

```
void pset_global_tran3(
 Pmatrix3       xform );        /* transformation matrix        */

void pset_global_tran(
Pmatrix         xform );        /* transformation matrix        */

void ptranslate3( /* BUILD A 3D TRANSLATION TRANSFORMATION MATRIX */
 Pvec3*trans_vector,            /* translation vector           */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix3 m );                  /* OUT transformation matrix  */

void ptranslate( /* BUILD A 2D TRANSLATION TRANSFORMATION MATRIX */
 Pvec  *trans_vector,           /* translation vector           */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix m );                   /* OUT transformation matrix  */

void pscale3( /* BUILD A 3D SCALE TRANSFORMATION MATRIX */
 Pvec3*scale_vector,            /* scale factor vector          */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix3 m );                  /* OUT transformation matrix  */

void pscale( /* BUILD A 2D SCALE TRANSFORMATION MATRIX */
 Pvec  *scale_vector,           /* scale factor vector          */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix m );                   /* OUT transformation matrix  */

void protate_x( /* BUILD A 3D ROTATION ABOUT THE X AXIS TRANSFORMATION MATRIX */
 Pfloat_arg     angle,          /* rotation angle               */
 Pint           *error_ind,     /* OUT error indicator          */
 Pmatrix3        m );           /* OUT transformation matrix  */

void protate_y( /* BUILD A 3D ROTATION ABOUT THE Y AXIS TRANSFORMATION MATRIX */
Pfloat_arg      angle,          /* rotation angle               */
 Pint           *error_ind,     /* OUT error indicator          */
 Pmatrix3       m );            /* OUT transformation matrix  */

void protate_z( /* BUILD A 3D ROTATION ABOUT THE Z AXIS TRANSFORMATION MATRIX */
Pfloat_arg      angle,          /* rotation angle               */
 Pint           *error_ind,     /* OUT error indicator          */
 Pmatrix3       m );            /* OUT transformation matrix  */

void protate( /* BUILD A 2D ROTATION TRANSFORMATION MATRIX */
 Pfloat_arg     angle,          /* rotation angle               */
 Pint           *error_ind,     /* OUT error indicator          */
 Pmatrix        m );            /* OUT transformation matrix  */

void pcompose_matrix3(
 Pmatrix3       a,              /* matrix a     */
 Pmatrix3       b,              /* matrix b     */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix3       m );            /* OUT result matrix            */

void pcompose_matrix(
 Pmatrix        a,              /* matrix a     */
 Pmatrix        b,              /* matrix b     */
 Pint   *error_ind,             /* OUT error indicator          */
 Pmatrix        m );            /* OUT result matrix            */

void ptran_point3(
 Ppoint3        *p,             /* point        */
 Pmatrix3       m,              /* transformation matrix        */
 Pint           *error_ind,     /* OUT error indicator          */
 Ppoint3        *r );           /* OUT transformed point        */

void ptran_point(
 Ppoint         *p,             /* point        */
```

```
Pmatrix        m,            /* transformation matrix       */
Pint           *error_ind,   /* OUT error indicator         */
Ppoint         *r );         /* OUT transformed point       */

void pbuild_tran_matrix3(
Ppoint3        *pt,          /* fixed point   */
Pvec3          *shift,       /* shift vector  */
Pfloat_arg     x_angle,      /* rotation angle X            */
Pfloat_arg     y_angle,      /* rotation angle Y            */
Pfloat_arg     z_angle,      /* rotation angle Z            */
Pvec3          *scale,       /* scale vector */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix3       matrix );     /* OUT transformation matrix   */

void pbuild_tran_matrix(
Ppoint         *pt,          /* fixed point   */
Pvec           *shift,       /* shift vector  */
Pfloat_arg     angle,        /* rotation angle              */
Pvec           *scale,       /* scale vector */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix        matrix );     /* OUT transformation matrix   */

void pcompose_tran_matrix3(
Pmatrix3       m,            /* transformation matrix       */
Ppoint3        *pt,          /* fixed point   */
Pvec3          *shift,       /* shift vector  */
Pfloat_arg     x_ang,        /* rotation angle X            */
Pfloat_arg     y_ang,        /* rotation angle Y            */
Pfloat_arg     z_ang,        /* rotation angle Z            */
Pvec3          *scale,       /* scale vector */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix3       result );     /* OUT transformation matrix   */

void pcompose_tran_matrix(
Pmatrix        m,            /* transformation matrix       */
Ppoint         *pt,          /* fixed point   */
Pvec           *shift,       /* shift vector  */
Pfloat_arg     angle,        /* rotation angle              */
Pvec           *scale,       /* scale vector                */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix        result );     /* OUT transformation matrix   */

void peval_view_ori_matrix3(
Ppoint3        *vrp,         /* view reference point        */
Pvec3          *vpn,         /* view plane normal           */
Pvec3          *vup,         /* view up vector              */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix3       m );          /* OUT view orientation matrix */

void peval_view_ori_matrix(
Ppoint         *vrp,         /* view reference point        */
Pvec  *        vup,          /* view up vector              */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix        m );          /* OUT view orientation matrix */

void peval_view_map_matrix(
Pview_map      *map,         /* view mapping                */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix        m );          /* OUT view mapping matrix     */

void peval_view_map_matrix3(
Pview_map3     *map,         /* view mapping                */
Pint           *error_ind,   /* OUT error indicator         */
Pmatrix3       m );          /* OUT view mapping matrix     */

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

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

void pinq_pred_view_rep( /* INQUIRE PREDEFINED VIEW TABLE ENTRIES */
 Pint           type,                /* workstation type              */
 Pint           index,               /* predefined view index         */
 Pint           *error_ind,          /* OUT error indicator           */
 Pview_rep3     *rep );              /* OUT view representation       */

void pset_ws_win3(
 Pint           ws,                  /* workstation id                */
 Plimit3*window );                   /* workstation window limits     */

void pset_ws_win(
 Pint    ws,                         /* workstation id                */
 Plimit  *window );                  /* workstation window limits     */

void pset_ws_vp3(
 Pint    ws,                         /* workstation id                */
 Plimit3*viewport );                 /* workstation viewport limits   */

void pset_ws_vp(
 Pint    ws,                         /* workstation id                */
 Plimit  *viewport );                /* workstation viewport limits   */

void pinq_list_view_inds(
 Pint    ws,                         /* workstation identifier        */
 Pint    length,                     /* length of application list    */
 Pint    start,                      /* starting position             */
 Pint    *error_ind,                 /* OUT error indicator           */
 Pint_list *indices,                 /* OUT list of view indices      */
 Pint    *total_length );            /* OUT length of list in PHIGS   */

void pinq_view_rep(
 Pint    ws,                         /* workstation identifier        */
 Pint    view_index,                 /* view index                    */
 Pint    *error_ind,                 /* OUT error indicator           */
 Pupd_st *update_state,              /* OUT transformation update state          */
 Pview_rep3     *cur_rep,            /* OUT current view representation          */
 Pview_rep3     *req_rep );          /* OUT requested view representation        */

void pinq_ws_tran3(
 Pint    ws,                         /* workstation identifier        */
 Pint    *error_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         */

void pinq_ws_tran(
 Pint           ws,                  /* workstation identifier        */
 Pint           *error_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         */

void pset_view_rep3(
 Pint           ws,                  /* workstation id                */
 Pint           index,               /* view index                    */
 Pview_rep3     *rep );              /* view representation           */

void pset_view_rep(
```

```
 Pint          ws,           /* workstation id            */
 Pint          index,        /* view index                */
 Pview_rep     *rep );       /* view representation       */

void pset_view_tran_in_pri(
 Pint    ws,                 /* workstation id            */
 Pint    index,              /* view index                */
 Pint    ref_index,          /* reference view index      */
 Prel_pri priority );        /* relative priority         */
```

Modelling

```
void pinq_model_clip_facs(
 Pint    length,             /* length of application list    */
 Pint    start,              /* starting position             */
 Pint    *error_ind,         /* OUT error indicator           */
 Pint    *num,               /* OUT number of distinct planes in modelling clipping volume */
 Pint_list *ops,             /* OUT list of operators         */
 Pint    *total_length );    /* OUT length of list in PHIGS   */

void pset_model_clip_vol3(  /* SET MODELLING CLIPPING VOLUME 3D */
 Pint          op,           /* __modelclipoperator, see PHIGS constants */
 Phalf_space_list3 *half_spaces ); /* list of half spaces       */

void pset_model_clip_vol(   /* SET VOLUME CLIPPING VOLUME */
 Pint          op,           /* __modelclipoperator, see PHIGS constants */
 Phalf_space_list *half_spaces ); /* list of half spaces        */

void pset_model_clip_ind(   /* TURN MODEL CLIPPING ON OR OFF */
 Pclip_ind     ind );        /* clipping indicator        */

void prestore_model_clip_vol();
```

Input

```
void pset_loc_mode(
 Pint    ws,                 /* workstation identifier    */
 Pint    dev,                /* locator device number     */
 Pop_mode      mode,         /* operating mode            */
 Pecho_switch echo );        /* echo switch               */

void pset_stroke_mode(
 Pint    ws,                 /* workstation identifier    */
 Pint    dev,                /* stroke device number      */
 Pop_mode      mode,         /* operating mode            */
 Pecho_switch echo );        /* echo switch  */

void pset_val_mode(
 Pint    ws,                 /* workstation identifier    */
 Pint    dev,                /* valuator device number    */
 Pop_mode      mode,         /* operating mode            */
 Pecho_switch echo );        /* echo switch               */

void pset_choice_mode(
 Pint    ws,                 /* workstation identifier    */
 Pint    dev,                /* choice device number      */
 Pop_mode      mode,         /* operating mode            */
 Pecho_switch echo );        /* echo switch  */

void pset_pick_mode(
 Pint    ws,                 /* workstation identifier    */
 Pint    dev,                /* pick device number        */
 Pop_mode      mode,         /* operating mode            */
 Pecho_switch echo );        /* echo switch               */
```

```
void pset_string_mode(
 Pint    ws,                    /* workstation identifier       */
 Pint    dev,                   /* string device number         */
 Pop_mode      mode,            /* operating mode               */
 Pecho_switch echo );           /* echo switch                  */

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

void pflush_events(
Pint    ws,                     /* workstation identifier       */
 Pin_class     class,           /* device clas  s               */
 Pint    dev );                 /* logical input device number */

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

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

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

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

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

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

void pget_pick(
 Pint          depth,           /* depth of pick path to return */
 Pin_status    *in_status,       /* OUT pick status             */
 Ppick_path    *rpick );        /* OUT pick path                */

void pget_string(
 char   *string );              /* OUT pointer to string of bufsiz for the device + 1 for the terminator. */

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

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

void psample_stroke(
 Pint    ws,                    /* workstation identifier       */
 Pint    dev,                   /* stroke device number         */
 Pint    *view_ind,             /* OUT view index               */
 Ppoint_list *stroke );         /* OUT locator position         */
```

```
void psample_stroke3(
 Pint    ws,                      /* workstation identifier        */
 Pint    dev,                     /* stroke device number          */
 Pint    *view_ind,                /* OUT view index               */
 Ppoint_list3 *stroke );          /* OUT locator position          */

void psample_val(
 Pint    ws,                      /* workstation identifier        */
 Pint    dev,                     /* valuator device number        */
 Pfloat *value  );                /* OUT value                     */

void psample_choice(
 Pint    ws,                      /* workstation identifier        */
 Pint    dev,                     /* choice device number          */
 Pin_status *choice_in_status, /* OUT choice input status */
 Pint    *choice );               /* OUT choice  */

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

void psample_string(
 Pint    ws,                      /* workstation identifier        */
 Pint    dev,                     /* string device number          */
 char   *string );                /* OUT string                    */

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

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

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

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

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

void preq_choice(
 Pint    ws,                      /* workstation identifier        */
 Pint    dev,                     /* choice device number          */
 Pin_status *in_status,           /* OUT input status              */
```

```
 Pint    *choice );                 /* OUT choice                 */

void preq_pick(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* pick device number         */
 Pint    depth,                     /* max. depth of returned path */
 Pin_status *in_status,             /* OUT input status           */
Ppick_path *rpick );                /* OUT pick                   */

void preq_string(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* string device number       */
 Pin_status *in_status ,            /* OUT input status           */
 char *string );                    /* OUT string                 */;

void pinit_loc3(
 Pint          ws,                  /* workstation identifier     */
 Pint          dev,                 /* locator device number      */
 Pint          init_view_ind,       /* initial view indicator     */
 Ppoint3       *init_loc_pos,       /* initial locator position   */
 Pint          pet,                 /* __promptecholoc type       */
 Plimit3*echo_volume,               /* echo volume pointer        */
 Ploc_data3    *record );           /* data record pointer        */

void pinit_loc(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* locator device number      */
 Pint    init_view_ind,             /* initial view indicator     */
 Ppoint *init_loc_pos,              /* initial locator pointer    */
 Pint    pet,                       /* __prompyecholoc type       */
 Plimit  *echo_area,                /* echo area pointer          */
 Ploc_data     *record );           /* data record pointer        */

void pinit_stroke3(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* stroke device number       */
 Pint    init_view_ind,             /* initial view indicator     */
 Ppoint_list3 *init_stroke,         /* initial stroke pointer     */
 Pint    pet,                       /* __promptechostroke type    */
 Plimit3*echo_volume,               /* echo volume pointer        */
 Pstroke_data3 *rec );              /* data record pointer        */

void pinit_stroke(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* stroke device number       */
 Pint    init_view_ind,             /* initial view indicator     */
 Ppoint_list     *init_stroke,      /* initial stroke pointer     */
 Pint    pet,                       /* __promptecostroke type     */
 Plimit  *echo_area,                /* echo area pointer          */
 Pstroke_data *record );            /* data record pointer        */

void pinit_val3(
 Pint    ws,                        /* workstation identifier     */
 Pint    dev,                       /* valuator device number     */
 Pfloat_arg      init,              /* initial value              */
 Pint    pet,                       /* __promptechoval type       */
 Plimit3*echo_volume,               /* echo volume pointer        */
 Pval_data3    *record );           /* data record pointer        */

void pinit_val(
Pint     ws,                        /* workstation identifier     */
 Pint    dev,                       /* valuator device number     */
 Pfloat_arg      init,              /* initial value              */
 Pint    pet,                       /* __promptechoval type       */
 Plimit  *echo_area,                /* echo area pointer          */
 Pval_data     *record );           /* data record pointer        */
```

```
void pinit_choice3(
 Pint           ws,             /* workstation identifier      */
 Pint           dev,            /* choice device number        */
 Pin_status     istat,          /* initial choice status       */
 Pint           init,           /* initial choice              */
 Pint           pet,            /* __promptechochoice type*/
 Plimit3        *echo_volume,/* echo volume pointer        */
 Pchoice_data3 *record );       /* data record pointer         */

void pinit_choice(
 Pint           ws,             /* workstation identifier      */
 Pint           dev,            /* choice device number        */
 Pin_status     istat,          /* initial choice status       */
 Pint           init,           /* initial choice              */
 Pint           pet,            /* __promptechochoice type*/
 Plimit         *echo_area,     /* echo area pointer           */
 Pchoice_data *record );        /* data record pointer         */

void pinit_pick3(
 Pint    ws,                    /* workstation identifier      */
 Pint    dev,                   /* pick device number          */
 Pin_status     istat,          /* initial pick status         */
 Ppick_path     *init,          /* initial pick pointer        */
 Pint    pet,                   /* __promptechopick type       */
 Plimit3*echo_volume,           /* echo volume pointer         */
 Ppick_data3    *record,        /* data record pointer         */
 Ppath_order    order );        /* pick path order             */

void pinit_pick(
 Pint    ws,                    /* workstation identifier      */
 Pint    dev,                   /* pick device number          */
 Pin_status     istat,          /* initial pick status         */
 Ppick_path     *init,          /* initial pick pointer        */
 Pint    pet,                   /* __promptechopick type       */
 Plimit  *echo_area,            /* echo area pointer           */
 Ppick_data     *record,        /* data record pointer         */
 Ppath_order    order );        /* pick path order             */

void pinit_string3(
Pint     ws,                    /* workstation identifier      */
 Pint    dev,                   /* string device number        */
 char    *init,                 /* initial string              */
 Pint    pet,                   /* __promptechostring type     */
 Plimit3*echo_volume,           /* echo volume pointer         */
 Pstring_data3 *record );       /* data record pointer         */

void pinit_string(
 Pint    ws,                    /* workstation identifier      */
 Pint    dev,                   /* string device number        */
 char    *init,                 /* initial string              */
 Pint    pet,                   /* __promptechostrimg type     */
 Plimit  *echo_area,            /* echo area pointer           */
 Pstring_data *record );        /* data record pointer         */

void pinq_loc_st3(
 Pint    ws,                    /* workstation identifier      */
 Pint    dev,                   /* locator device number       */
 Pinq_type      type,           /* type of returned value      */
 Pstorestore,                   /* store handle                */
 Pint    *err,                  /* OUT error indicator         */
 Pop_mode *op_mode,             /* OUT operating mode          */
 Pecho_switch *echo_switch,     /* OUT echo switch             */
 Pint    *init_view_ind,        /* OUT initial view indicator  */
 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             */
```

```
void pinq_loc_st(
 Pint    ws,                          /* workstation identifier         */
 Pint    dev,                         /* locator device number          */
 Pinq_type       type,                /* type of returned value         */
 Pstorestore,                         /* pointer to buffer              */
 Pint    *err,                        /* OUT error indicator            */
 Pop_mode *op_mode,                   /* OUT operating mode             */
 Pecho_switch *echo_switch,           /* OUT echo switch                */
 Pint    *init_view_ind,              /* OUT initial view indicator     */
 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                */

void pinq_stroke_st3(
 Pint    ws,                          /* workstation identifier         */
 Pint    dev,                         /* stroke device number           */
 Pinq_type       type,                /* type of returned value         */
 Pstorestore,                         /* handle to Store object         */
 Pint    *err,                        /* OUT error indicator            */
 Pop_mode *op_mode,                   /* OUT operating mode             */
 Pecho_switch *echo_switch,           /* OUT echo switch                */
 Pint    *init_view_ind,              /* OUT initial view indicator */
 Ppoint_list3 **init_stroke,          /* OUT initial stroke             */
 Pint    *prompt_echo,                /* OUT prompt/echo type           */
 Plimit3 *echo_vol,                   /* OUT echo area                  */
 Pstroke_data3 **stroke_data ); /* OUT data record */  /* what is US T9 ? (IS p. 132) */

void pinq_stroke_st(
 Pint    ws,                          /* workstation identifier         */
 Pint    dev,                         /* stroke device number           */
 Pinq_type       type,                /* type of returned value         */
 Pstorestore,                         /* handle to Store object         */
 Pint    *err,                        /* OUT error indicator            */
 Pop_mode *op_mode                    /* OUT operating mode             */
 Pecho_switch *echo_switch,           /* OUT echo switch                */
 Pint    *init_view_ind,              /* OUT initial view indicator */
 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                */

void pinq_val_st3(
 Pint    ws,                          /* workstation identifier         */
 Pint    dev,                         /* valuator device number         */
 Pstore store,                        /* OUT pointer to buffer          */
 Pint    *err,                        /* 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 area                  */
 Pval_data3 **val_data );             /* OUT data record                */

void pinq_val_st(
 Pint    ws,                          /* workstation identifier         */
 Pint    dev,                         /* valuator device number         */
 Pstorestore,                         /* OUT pointer to buffer          */
 Pint    *err,                        /* 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                */

void pinq_choice_st3(
 Pint    ws,                          /* workstation identifier         */
```

```
 Pint    dev,                      /* choice device number      */
 Pstorestore,                      /* OUT pointer to buffer     */
 Pint    *err,                     /* 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 area             */
 Pchoice_data3 **choice_data ); /* OUT data record              */

void pinq_choice_st(
 Pint    ws,                       /* workstation identifier    */
 Pint    dev,                      /* choice device number      */
 Pstorestore,                      /* OUT pointer to buffer     */
 Pint    *err,                     /* 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               */

void pinq_pick_st3(
 Pint    ws,                       /* workstation identifier    */
 Pint    dev,                      /* pick device number        */
 Pinq_type      type,              /* type of returned value    */
 Pstorestore,                      /* pointer to buffer         */
 Pint    *err,                     /* OUT error indicator       */
 Pop_mode *op_mode,                /* OUT operating mode        */
 Pecho_switch *echo_switch, /* OUT echo switch                  */
 Pfilter **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 area             */
 Ppick_data3 **pick_data,          /* OUT data record           */
 Ppath_order   *path_order);  /* OUT path order */

void pinq_pick_st(
 Pint    ws,                       /* workstation identifier    */
 Pint    dev,                      /* pick device number        */
 Pinq_type      type,              /* type of returned value    */
 Pstorestore,                      /* pointer to buffer         */
 Pint    *err,                     /* OUT error indicator       */
 Pop_mode *op_mode,                /* OUT operating mode        */
 Pecho_switch *echo_switch, /* OUT echo switch                  */
 Pfilter **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   *path_order ); /* OUT path order                 */

void pinq_string_st3(
 Pint    ws,                       /* workstation identifier    */
 Pint    dev,                      /* string device number      */
 Pstore store,                     /* handle to Store object    */
 Pint    *err,                     /* 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 area             */
 Pstring_data3 **string_data ); /* OUT data record              */
```

```
void pinq_string_st(
 Pint    ws,                    /* workstation identifier          */
 Pint    dev,                   /* string device number            */
 Pstore store,                  /* OUT pointer to buffer           */
 Pint    *err,                  /* 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                  */

void pinq_more_simult_events(
 Pint     *error_ind,           /* OUT error indicator             */
 Pmore_simult_events *events);/* OUT simultaneous events  */

void pinq_in_overf(
 Pint    *error_ind,            /* OUT error indicator             */
 Pint    *ws,                   /* OUT workstation identifier      */
 Pin_class *class,              /* OUT input class                 */
 Pint    *dev );                /* OUT input device number         */

void pset_pick_id( /* SET A PICK IDENTIFIER, INHERITED BY EXECUTED STRUCTURES */
Pint     pick_id );             /* pick identifier                 */

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

void pinq_def_pick_data(
 Pint    type,                  /* workstation type */
 Pint    device,                /* logical input device number */
 Pstore store,                  /* handle to Store object          */
 Pint    *error_ind,            /* OUT error indicator             */
 Pint_list **pets,              /* OUT list of prompt and echo types , __promptechopick */
 Plimit  *echo_area,            /* OUT default echo volume */
 Ppick_data **pick_data );      /* OUT default data record         */

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

void pinq_def_loc_data(
 Pint type,                     /* workstation type */
 Pint device,                   /* logical input device number */
 Pstore store,                  /* handle to Store object */
 Pint *error_ind,               /* OUT error indicator */
Ppoint *loc_pos,                /* OUT default initial position */
Pint_list **pets,               /* OUT list of prompt and echo types, __promptechploc */
Plimit *echo_area,              /* OUT default echo volume */
Ploc_data **loc_data );         /* OUT default data record */

void pinq_def_stroke_data3(
 Pint    type,                  /* workstation type */
 Pint    device                 /* logical input device number */
 Pstore store,                  /* handle to Store object */
```

```
 Pint    *error_ind,                 /* OUT error indicator            */
 Pint    *max_buf_size,              /* OUT max. input buffer size */
 Pint_list **pets,                   /* OUT list of prompt and echo types , __promptechostroke */
 Plimit3*echo_volume,                /* OUT default echo volume */
 Pstroke_data3 **stroke_data ); /* OUT default data record */

void pinq_def_stroke_data(
 Pint    type,                       /* workstation type */
 Pint    device,                     /* logical input device number */
 Pstorestore,                        /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator            */
 Pint    *max_buf_size,              /* OUT max. input buffer size */
 Pint_list **pets,                   /* OUT list of prompt and echo types , __promptechostroke */
 Plimit  *echo_area,                 /* OUT default echo volume */
 Pstroke_data **stroke_data ); /* OUT default data record */

void pinq_def_val_data3(
 Pint    type,                       /* workstation type */
 Pint    device,                     /* logical input device number */
 Pstorestore,                        /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator            */
 Pfloat *def_value,                  /* OUT default initial value */
 Pint_list **pets,                   /* OUT list of prompt and echo types , __promptechoval */
 Plimit3*echo_vol,                   /* OUT default echo volume */
 Pval_data3 **val_data );            /* OUT default data record */

void pinq_def_val_data(
 Pint    type,                       /* workstation type */
 Pint    device,                     /* logical input device number */
 Pstorestore,                        /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator            */
 Pfloat *def_value,                  /* OUT default initial value */
 Pint_list **pets,                   /* OUT list of prompt and echo types, _promptechoval  */
 Plimit  *echo_area,                 /* OUT default echo volume */
 Pval_data **val_data  );            /* OUT default data record */

void pinq_def_choice_data3(
 Pint    type,                       /* workstation type */
 Pint    device,                     /* logical input device number */
 Pstorestore,                        /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator            */
 Pint    *max_choices,               /* OUT max. number of choices */
 Pint_list **pets,                   /* OUT list of prompt and echo types ,__promptechochoice */
 Plimit3*echo_vol,                   /* OUT default echo volume */
 Pchoice_data3 **choice_data ); /* OUT default data record */

void pinq_def_choice_data(
 Pint    type,                       /* workstation type */
 Pint    device,                     /* logical input device number */
 Pstorestore,                        /* handle to Store object */
 Pint    *error_ind,                 /* OUT error indicator            */
 Pint    *max_choices,               /* OUT max. number of choices */
 Pint_list **pets,                   /* OUT list of prompt and echo types , __promptechochoice */
 Plimit  *echo_area,                 /* OUT default echo volume */
 Pchoice_data **choice_data);/* OUT default data record */

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

void pinq_def_string_data(
 Pint            type,               /* workstation type */
```

```
 Pint              device,          /* logical input device number */
 Pstore            store,           /* handle to Store object */
 Pint    *error_ind,                /* OUT error indicator            */
 Pint    *max_buf_size,             /* OUT max. input buffer size */
 Pint_list         **pets,          /* OUT list of prompt and echo types , __promptechostring */
 Plimit  *echo_area,                /* OUT default echo volume */
 Pstring_data   **string_data); /* OUT default data record */

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

Error handling

```
void perr_hand(
 Pint    errnum,
 Pint    funcnum,
 char   *fname );

void pset_err_hand_mode( /* SET ERROR HANDLER MODE */
 Perr_mode      mode );             /* error handling mode: PERR_ON/PERR_OFF */

void pset_err_hand( /* TELL PHIGS TO USE ANOTHER ERROR HANDLER */
 void   (*new_err_hand)(),          /* my error handler function */
 void   (**old_err_hand)() );       /* OUT the old error handler */

void pinq_err_hand_mode( /* INQUIRE ERROR HANDLER MODE */
 Pint    *error_ind,                /* OUT error indicator            */
 Perr_mode *mode );                 /* OUT error mode                 */

void perr_log( /* LOG AN ERROR, I.E. PRINT IT IN THE ERROR FILE */
 Pint    errnum,
 Pint    funcnum,
 char   *fname );

void pescape( /* SYNCHRONIZE ERROR HANDLING */
 Pint    func_id,                   /* escape function identifier   */
 Pescape_in_data *in,               /* input data for the function */
 Pstore            store,
 Pescape_out_data **out );          /* OUT output data of the function              */
```

Advanced CSS Features

Name sets let you group primitives and give them names. Filters let you highlight, make invisible, make pickable. these primitives. Names are stored in structures. Filters are defined (set/removed) on a per workstation basis.

```
void padd_names_set( /* ADD A SET OF NAMES TO THE CURRENT NAME LIST */
  Pint_list          *set );        /* set of names to be added  */

void premove_names_set( /* REMOVE A SET OF NAMES FROM CURRENT NAME LIST */
 Pint_list           *set );        /* set of names to be removed                   */

void pset_highl_filter( /* SET HIGHLIGHTNING FILTER */
 Pint    ws,                        /* workstation identifier         */
 Pfilter *filter  );                /* highlighting filter            */

void pset_pick_filter( /* SET DETACTABILITY FILTER - EMPTY INITIALLY */
 Pint    ws,                        /* workstation identifier         */
 Pint    dev_num,                   /* pick device number             */
 Pfilter *filter );                 /* pick filter                    */

void pset_invis_filter( /* SET THE INVISIBILITY FILTER */
 Pint    ws,                        /* workstation identifier         */
```

```
 Pfilter *filter );                    /* invisibility filter             */

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

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

void pincr_spa_search3(
 Ppoint3         *ref,                 /* search reference point          */
 Pfloat_arg      dist,                 /* search distance                 */
 Pelem_ref_list  *sp,                  /* starting path list              */
 Pclip_ind       mclip_flag,           /* model clip flag                 */
 Pint            ceil,                 /* search ceiling index            */
 Pfilter_list    *norm,                /* normal filter list              */
 Pfilter_list    *inv,                 /* inverted filter list            */
 Pint            len,                  /* length of application list      */
 Pint            st,                   /* starting position               */
 Pint            *error_ind,           /* OUT error indicator             */
 Pelem_ref_list  *fp,                  /* OUT found path                  */
 Pint            *tot_len );           /* OUT length of list in PHIGS     */

void pincr_spa_search(
 Ppoint          *ref,                 /* search reference point          */
 Pfloat_arg      dist,                 /* search distance                 */
 Pelem_ref_list  *sp,                  /* starting path list              */
 Pclip_ind       mclip_flag,           /* model clip flag                 */
 Pint            ceil,                 /* search ceiling index            */
 Pfilter_list    *norm,                /* normal filter list              */
 Pfilter_list    *inv,                 /* inverted filter list            */
 Pint            len,                  /* length of application list      */
 Pint            st,                   /* starting position               */
 Pint            *error_ind,           /* OUT error indicator             */
 Pelem_ref_list  *fp,                  /* OUT found path                  */
 Pint            *tot_len );           /* OUT length of list in PHIGS     */
```

PHIGS Datatypes and Enumerations

typedef struct { Pgcolr colr; } **Pamb_light_src_rec;**

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

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

```
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_LINE_SHAD_METH,  PASPECT_INT_SHAD_METH,
  PASPECT_REFL_PROPS,     PASPECT_INT_REFL_EQN,         PASPECT_BACK_INT_STYLE,   PASPECT_BACK_INT_STYLE_IND,
  PASPECT_BACK_INT_COLR,        PASPECT_BACK_INT_SHAD_METH,   PASPECT_BACK_REFL_PROPS,
  PASPECT_BACK_INT_REFL_EQN,  PASPECT_CURVE_APPROX_CRIT,     PASPECT_SURF_APPROX_CRIT } Paspect;
```

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

typedef struct { Pint num_attrs; Pattrs *attrs; } **Pattrs_list**;

typedef struct {
union Pchoice_pets { struct Pchoice_pet_r1 { Pint unused; } pet_r1;

struct Pchoice_pet_r2 { Pint num_prompts; Ppr_switch *prompts; } pet_r2;
struct Pchoice_pet_r3 { Pint num_strings; char **strings; } pet_r3;
struct Pchoice_pet_r4 { Pint num_strings; char **strings; } pet_r4;
struct Pchoice_pet_r5 { Pint struct_id; Pint num_pick_ids; Pint *pick_ids; } pet_r5;
} pets; } **Pchoice_data**;

typedef Pchoice_data **Pchoice_data3**;

typedef struct { Pfloat cieluv_x; Pfloat cieluv_y; Pfloat cieluv_y_lum; } **Pcieluv**;

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

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

typedef struct { Pint num_colrs; Pcolr_avail colr_avail; Pint num_pred_inds; Pcieluv prim_colrs(3); } **Pcolr_facs**;

typedef union { struct { Pint colr_model; Pfloat_list weights; Pcolr_rep_list colrs; } meth_r2; struct { Pint colr_model; Pfloat_list_list colr_lists; } meth_r3; } **Pcolr_map_data**;

typedef struct { Pint_list meths; Pint num_pred_inds; } **Pcolr_map_facs**;

typedef struct { Pint int_data; } **Pcolr_map_st;**

typedef union { Prgb rgb; Pcieluv cieluv; Phls hls; Phsv hsv; Pdata unsupp; } **Pcolr_rep**;

typedef struct { Pint num_colr_reps; Pcolr_rep *colr_reps; } **Pcolr_rep_list**;

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

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

typedef void * **Pconnid;**

typedef struct { Display *display; XID drawable_id; } **Pconnid_x_drawable**;

typedef struct { Pcoval colr; Pvec3 norm; } **Pconorm3**;

typedef union { Pint ind; Pcolr_rep direct; } **Pcoval;**

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

typedef enum { PCULL_NONE, PCULL_BACKFACE, PCULL_FRONTFACE } **Pcull_mode**;

typedef struct { Pint max_bsp_order, max_tc_order; Pint_list cat_types, sat_types, tcat_types, psc_types; } Pcurv-surf_facs; typedef enum { PCP_UNIFORM, PCP_NON_UNIFORM } **Pcurve_placement**;

typedef struct { size_t size; void *data; } **Pdata**;

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

typedef struct { Pdcue_mode mode; Pfloat ref_planes(2); Pfloat scaling(2); Pgcolr colr; } **Pdcue_bundle**;

typedef enum { PSUPPRESSED, PALLOWED } **Pdcue_mode**;

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

typedef struct { Pgcolr colr; Pvec3 dir; } **Pdir_light_src_rec**;

typedef struct { Pdc_units dc_units; Pfloat_size size_dc; Pint_size size_raster; } **Pdisp_space_size;**

typedef struct { Pdc_units dc_units; Pfloat_size3 size_dc; Pint_size3 fsize_raster; } **Pdisp_space_size3**;

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

typedef enum { PDISTING_NO, PDISTING_YES } **Pdisting_mode;**

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

typedef struct { Pdyn_mod line_bundle, marker_bundle, text_bundle, int_bundle, edge_bundle, pat_rep, colr_rep, view_rep, ws_tran, highl_filter, invis_filter, hlhsr_mode; } **Pdyns_ws_attrs;**

typedef struct { Pdyn_mod light_src_rep; Pdyn_mod dcue_rep; Pdyn_mod colr_map_rep; } **Pdyns_ws_attrs_plus**;

typedef struct { Pdyn_mode content, post, unpost, del, ref; } Pdyns_structs; typedef enum { PSWITCH_NO_ECHO, PSWITCH_ECHO } **Pecho_switch**;

typedef struct { Pasf flag_asf, type_asf, width_asf, colr_ind_asf; Pint ind; Pedge_bundle bundle; } **Pedge_attrs**;

typedef struct { Pedge_flag flag; Pint type; Pfloat fwidth; Pint colr_ind; } **Pedge_bundle**;

typedef struct { Pedge_flag flag; Pint type; Pfloat width; Pgcolr colr; } **Pedge_bundle_plus**;

typedef union { Pedge_flag *edges; } **Pedge_data_arr**;

typedef struct { Pint num_edges; Pedge_data_arr edgedata; } **Pedge_data_list**;

typedef struct { Pint num_lists; Pedge_data_list *edgelist; } **Pedge_data_list_list**;

typedef struct { Pint_list types; Pint num_widths; Pfloat nom_width, min_width, max_width, Pint num_pred_inds; } **Pedge_facs**;

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

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

typedef struct { Pint struct_id; Pint elem_pos; } **Pelem_ref;**

typedef struct { Pint num_elem_refs; Pelem_ref *elem_refs; } **Pelem_ref_list**;

typedef struct { Pint num_elem_ref_lists; Pelem_ref_list *elem_ref_lists; } **Pelem_ref_list_list**;

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

typedef enum { PERRSYNC_OFF, PERRSYNC_ON} **Perrsync**;

typedef union { Pcoval *colrs; Pvec3 *norms; Pconorm3 *conorms; } **Pfacet_data_arr3**;

typedef union { Pcoval colr; Pvec3 norm; Pconorm3 conorm; } **Pfacet_data3**;

typedef union { Ppoint3 *points; Pptco3 *ptcolrs; Pptnorm3 *ptnorms; Pptconorm3 *ptconorms; } **Pfacet_vdata_arr3**;

typedef struct { Pint num_vertices; Pfacet_vdata_arr3 vertex_data; } **Pfacet_vdata_list3**;

typedef struct { Pint_list incl_set; Pint_list excl_set; } **Pfilter**;

typedef struct { Pint num_filters; Pfilter *filters; } **Pfilter_list**;

typedef float **Pfloat**;

typedef double **Pfloat_arg;**

typedef struct { Pint num_floats; Pfloat *floats; } **Pfloat_list**;

typedef struct { Pint num_lists; Pfloat_list *lists; } **Pfloat_list_list**;

typedef struct { Pfloat size_x; Pfloat size_y; } **Pfloat_size**;

typedef struct { Pfloat size_x; Pfloat size_y; Pfloat size_z; } **Pfloat_size3**;

typedef struct { Pint type; union { Pint ind; struct { Pfloat x, y, z } general; } val; } **Pgcolr;**

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

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

typedef struct { Pint num_half_spaces; Phalf_space *half_spaces; } **Phalf_space_list;**

typedef struct { Pint num_half_spaces; Phalf_space3 *half_spaces; } **Phalf_space_list3**;

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

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

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

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

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

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

typedef int **Pint**;

typedef struct { Pasf style_asf, style_ind_asf, colr_ind_asf; Pint ind; Pint_bundle bundle; } **Pint_attrs**;

typedef struct { Pint_style style; Pint style_ind; Pint colr_ind; } **Pint_bundle**;

typedef struct { Pint_style style; Pint style_ind; Pgcolr colr; Pint refl_eqn; Pint shad_meth; Prefl_props refl_props; Pint_style back_style; Pint back_style_ind;
Pgcolr back_colr; Pint back_refl_eqn; Pint back_shad_meth;
Prefl_props back_refl_props; Pint approx_type; Pfloat approx_val(2); } **Pint_bundle_plus**;

typedef struct { Pint fnum_int_styles; Pint_style int_styles(5); Pint_list hatch_styles; Pint num_pred_inds; } **Pint_facs**;

typedef struct { Pint num_int_styles; Pint_style *int_styles; Pint_list hatch_styles; Pint num_pred_inds; Pint_list refl_eqns; Pint_list shad_meths; } **Pint_facs_plus**;

typedef struct { Pint num_ints; Pint *ints; } **Pint_list**;

typedef struct { Pint num_lists; Pint_list *lists; } **Pint_list_list;**

typedef struct { Pint size_x; Pint size_y; } **Pint_size**;

typedef struct { Pint size_x; Pint size_y; Pint size_z; } **Pint_size3**;

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

typedef struct { Pint type; Plight_src_rec rec; } **Plight_src_bundle**; /* type is __lighttypes */

typedef struct { Pint_list types; Pint max; Pint num_pred_inds; } **Plight_src_facs**;

typedef union { Pamb_light_src_rec ambient; Pdir_light_src_rec directional;
Ppos_light_src_rec positional; Pspot_light_src_rec spot; } **Plight_src_rec**;

typedef struct { Pfloat x_min; Pfloat x_max; Pfloat y_min; Pfloat y_max; } **Plimit**;

typedef struct { Pfloat x_min; Pfloat x_max; Pfloat y_min; Pfloat y_max; Pfloat z_min; Pfloat z_max; } **Plimit3**;

typedef struct { Pasf type_asf, width_asf, Pasf colr_ind_asf; Pint ind; Pline_bundle bundle; } **Pline_attrs**;

typedef struct { Pint type; Pfloat width; Pint colr_ind; } **Pline_bundle**;

typedef struct { Pint type; Pfloat width; Pgcolr colr; Pint shad_meth; Pint approx_type; Pfloat approx_val; } **Pline_bundle_plus**;

typedef struct { Pint_list types; Pint num_widths; Pfloat nom_width, min_width, max_width; Pint num_pred_inds;} Pline_facs; typedef struct { Pint_list types; Pint num_widths; Pfloat nom_width, min_width, max_width; Pint num_pred_inds; Pint_list shads; } **Pline_facs_plus**;

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

typedef union { Ppoint3 *points; Pptco3 *ptcolrs; } **Pline_vdata_arr3**;

typedef struct { Pint num_vertices; Pline_vdata_arr3 vertex_data; } **Pline_vdata_list3**;

typedef struct { union Ploc_pets { struct Ploc_pet_r1 { Pint unused; } pet_r1; struct Ploc_pet_r2 { Pint unused; } pet_r2;
struct Ploc_pet_r3 { Pint unused; } pet_r3;
struct Ploc_pet_r4 { Pline_attrs line_attrs; } pet_r4;
struct Ploc_pet_r5 { Pline_fill_ctrl_flag line_fill_ctrl_flag;
union { Pline_attrs line_attrs; Pint_attrs int_attrs;
struct { Pint_attrs int_attrs; Pedge_attrs edge_attrs; } fill_set;
} attrs; } pet_r5;
struct Ploc_pet_u2 { Pint crosshair_colr; } pet_u2;
struct Ploc_pet_u4 { Pline_bundle line_bundle; } pet_u4;
struct Ploc_pet_u5 { Pline_bundle line_bundle; } pet_u5;
} pets; } **Ploc_data**;

typedef Ploc_data **Ploc_data3**;

typedef long **Plong**;

typedef struct { Pasf type_asf, size_asf, colr_ind_asf; Pint ind; Pmarker_bundle bundle; } **Pmarker_attrs;**

typedef struct { Pint type; Pfloat size; Pint colr_ind; } **Pmarker_bundle**;

typedef struct { Pint type; Pfloat size; Pgcolr colr; } **Pmarker_bundle_plus**;

typedef struct { Pint_list types; Pint num_sizes; Pfloat nom_size, min_size, max_size; Pint num_pred_inds; } **Pmarker_facs;**

typedef Pfloat **Pmatrix3**(4)(4);

typedef Pfloat **Pmatrix**(3)(3);

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

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

typedef struct { Pint loc, stroke, val, choice, pick, string; } **Pnum_in**;

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

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

typedef union { struct { Pint unused; } psc_1; struct { Pint unused; } psc_2;
struct { Pcurve_placement placement; Pint u_count; Pint v_count; } psc_3;
struct { Ppoint3 origin; Pvec3 direction; Pfloat_list params;} psc_4;
struct { Ppoint3 origin; Pvec3 direction; Pfloat_list params;} psc_5;
Pdata unsupp; } **Ppara_surf_characs**;

typedef struct { Ppoint3 p; Ppoint3 q; Ppoint3 r; } **Pparal**;

typedef struct { Pint_size fdims; Pint *colr_array; } **Ppat_rep;**

typedef struct { Pint_size dims; Pint type; Pcoval *colr_array; } **Ppat_rep_plus;**

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

typedef struct { Pint u_dim; Pint v_dim; } **Ppcs_dims**;

typedef struct { union Ppick_pets { struct Pick_pet_r1 { Pint unused; } pet_r1; struct Pick_pet_r2 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_r2;
struct Pick_pet_r3 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_r3;
struct Pick_pet_u1 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_u1;
struct Pick_pet_u2 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_u2;
struct Pick_pet_u3 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_u3;
struct Pick_pet_u4 { Pint highl_colr, highl_count; Pfloat highl_duration, ap_size; } pet_u4;
} pets; } **Ppick_data**;

typedef Ppick_data **Ppick_data3**;

typedef struct { Pint depth; Ppick_path_elem *path_list; } **Ppick_path;**

typedef struct { Pint struct_id; Pint pick_id; Pint elem_pos; } **Ppick_path_elem**;

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

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

typedef struct { Ppcs_dims num_points; union { Ppoint3 *point3d; Ppoint4 *point4d; } points; } **Ppoint_grid34**;

typedef struct { Pint num_points; Ppoint *points; } **Ppoint_list**;

typedef struct { Pint num_point_lists; Ppoint_list *point_lists; } **Ppoint_list_list**;

typedef struct { Pint num_point_lists; Ppoint_list3 *point_lists; } **Ppoint_list_list3;**

typedef struct { Pint num_points; union { Ppoint *point2d; Ppoint3 *point3d; } points; } **Ppoint_list23;**

typedef struct { Pint num_points; Ppoint3 *points; } **Ppoint_list3**;

typedef struct { Pint num_points; union { Ppoint3 *point3d; fPpoint4 *point4d; } points;} **Ppoint_list34**;

typedef struct { Pint num_points; Ppoint4 *points; } **Ppoint_list4**;

typedef struct { Pfloat x; Pfloat y; Pfloat z; Pfloat w; } **Ppoint4;**

typedef struct { Pgcolr colr; Ppoint3 pos; Pfloat coef(2); } **Ppos_light_src_rec**;

typedef struct { Pint id; Pfloat disp_pri; } **Pposted_struct**;

typedef struct { Pint num_postings; Pposted_struct *postings; } **Pposted_struct_list;**

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

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

typedef struct { Ppoint3 point; Pcoval colr; } **Pptco3**;

typedef struct { Ppoint3 point; Pcoval colr; Pvec3 norm; } **Pptconorm3**;

typedef struct { Ppoint3 point; Pvec3 norm; Pptnorm3;

typedef enum { PNON_RATIONAL = 0, PRATIONAL = 1 } **Prational**;

typedef struct { Ppoint p; Ppoint q; } **Prect**;

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

typedef struct { Pfloat ambient_coef, diffuse_coef, specular_coef; Pgcolr specular_colr;
Pfloat specular_exp, transpar_coef; } **Prefl_props**;

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

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

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

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

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

typedef struct { Pgcolr colr; Ppoint3 pos; Pvec3 dir; Pfloat exp; Pfloat coef(2); Pfloat angle; } **Pspot_light_src_rec**;

typedef void ***Pstore**;

typedef struct { Pint num_strings; char **strings; } **Pstring_list**;

typedef struct { int buffer_size, init_pos; union { struct Pstring_pet_r1 { Pint unused; } pet_r1; } pets; } **Pstring_data**;

typedef Pstring_data **Pstring_data3**;

typedef struct { Pint buffer_size, init_pos; Pfloat x_interval, y_interval, time_interval; union { struct Pstroke_pet_r1 { Pint unused; } pet_r1; struct Pstroke_pet_r2 { Pint unused; } pet_r2;
struct Pstroke_pet_r3 { Pmarker_attrs marker_attrs; } pet_r3;
struct Pstroke_pet_r4 { Pline_attrs line_attrs; } pet_r4;
struct Pstroke_pet_u3 { Pmarker_bundle marker_bundle; } pet_u3;
struct Pstroke_pet_u4 { Pline_bundle line_bundle; } pet_u4;
} pets; } **Pstroke_data**;

typedef struct { Pint buffer_size, init_pos; Pfloat x_interval, y_interval, z_interval, time_interval; union { struct Pstroke3_pet_r1 { Pint unused; } pet_r1; struct Pstroke3_pet_r2 { Pint unused; } pet_r2;
struct Pstroke3_pet_r3 { Pmarker_attrs marker_attrs; } pet_r3;
struct Pstroke3_pet_r4 { Pline_attrs line_attrs; } pet_r4;
struct Pstroke3_pet_u3 { Pmarker_bundle marker_bundle; } pet_u3;
struct Pstroke3_pet_u4 { Pline_bundle line_bundle; } pet_u4;
} pets; } **Pstroke_data3**;

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

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

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

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

typedef struct { Phor_text_align hor; Pvert_text_align vert; } **Ptext_align**;

typedef struct { Pint font; Ptext_prec prec; Pfloat char_expan; Pfloat char_space; Pint colr_ind; } **Ptext_bundle**;

typedef struct { Pint font; Ptext_prec prec; Pfloat char_expan; Pfloat char_space; Pgcolr colr; } **Ptext_bundle_plus**;

typedef struct { Pint num_font_precs; Ptext_font_prec *font_precs; Pint num_char_hts; Pfloat min_char_ht, max_char_ht; Pint num_char_expans;
Pfloat min_char_expan, max_char_expan; Pint num_pred_inds; } **Ptext_facs**;

typedef struct { Pint font; Ptext_prec prec; } **Ptext_font_prec**;

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

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

typedef struct { Pedge_flag visible; Prational rationality; Pint order; Pint approx_type; Pfloat approx_val; Pfloat_list knots; Pfloat tmin, tmax; Ppoint_list23 cpts; } **Ptrimcurve;**

typedef struct { Pint num_curves; Ptrimcurve *curves; } **Ptrimcurve_list**;

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

typedef struct { Pfloat low, high; union { struct Pval_pet_r1 { Pint unused; } pet_r1; struct Pval_pet_u1 { char *label; char *format; char *low_label; char *high_label; } pet_u1;
struct Pval_pet_u4 { enum { PDIAL_WRAPAROUND, PDIAL_STICK } dial_limit; Pint threshold; } pet_u4;
} pets;} **Pval_data**;

typedef Pval_data **Pval_data3**;

typedef struct { Pfloat delta_x; Pfloat delta_y; } **Pvec**;

typedef struct { Pfloat delta_x; Pfloat delta_y; Pfloat delta_z; } **Pvec3;**

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

typedef struct { Plimit win; Plimit proj_vp; } **Pview_map**;

typedef struct { Plimit win; Plimit3 proj_vp; Pproj_type proj_type; Ppoint3 proj_ref_point; Pfloat view_plane; Pfloat back_plane; Pfloat front_plane; } **Pview_map3**;

typedef struct { Pmatrix ori_matrix; Pmatrix map_matrix; Plimit clip_limit; Pclip_ind xy_clip; } **Pview_rep**;

typedef struct { Pmatrix3 ori_matrix; Pmatrix3 map_matrix; Plimit3 clip_limit; Pclip_ind xy_clip; Pclip_ind back_clip; Pclip_ind front_clip; } **Pview_rep3**;

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

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

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

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

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

typedef struct { Pint line_bundles, mark_bundles, text_bundles, int_bundles, edge_bundles, pat_reps, colr_reps, view_reps; } **Pws_st_tables**;

typedef struct { Pint line_bundles, mark_bundles, text_bundles, int_bundles, edge_bundles, pat_reps, colr_reps, view_reps, dcue_rep, light_src_rep, colr_map_rep; } **Pws_tables_plus**;

PHIGS Constants

PHIGS constants (for consistency most of these should be enumerations instead, someone has been lazy. In phigs.h they look like #define ABCSDSD (1), but I have brought connected constants together into groups and written them as enumerations instead.

IMPORTANT: these "enumeration variables" do NOT exist in phigs, you can use only the literal constants, THEY are defined with #define statements in phigs.h

Beware: Not everything here is necessarily legal C

enum __annostyles = { PANNO_STYLE_UNCONNECTED = 1, PANNO_STYLE_LEAD_LINE = 2 }

enum __charset = { PCS_ASCII = 0, PCS_GREEK = -1, PCS_SYMBOL = -2, PCS_CARTOGRAPHIC = -3, PCS_KANJI= -4,

```
                        PCS_KATAKANA = -5}

enum __colmapmethod = { PCOLR_MAP_TRUE = 1, PCOLR_MAP_PSEUDO = 2, PCOLR_MAP_PSEUDO_N = 3 }

enum __colmodel { PINDIRECT = 0, PMODEL_RGB = 1, PMODEL_CIELUV = 2, PMODEL_HSV = 3, PMODEL_HLS = 4 };

enum __ curveapproximation = {
  PCURV_WS_DEP = 1, PCURV_CONSTANT_PARAMETRIC_BETWEEN_KNOTS = 2,
  PCURV_CHORDAL_SIZE_WC = 3, PCURV_CHORDAL_SIZE_NPC = 4,   PCURV_CHORDAL_SIZE_DC = 5,
  PCURV_CHORDAL_DEVIATION_WC = 6, PCURV_CHORDAL_DEVIATION_NPC = 7,
  PCURV_CHORDAL_DEVIATION_DC = 8, PCURV_RELATIVE_WC = 9, PCURV_RELATIVE_NPC = 10,
  PCURV_RELATIVE_DC = 11 }

enum __edgedataflags = { PEDGE_NONE = 0, PEDGE_VISIBILITY = 1 }

enum __facetdataflags = {  PFACET_NONE=0, PFACET_COLOUR=1, PFACET_NORMAL=2,
                                  PFACET_COLOUR_NORMAL=3 }

enum __fonts = {  PFONT_MONO = 1, PFONT_SIMPLEX = 2, PFONT_DUPLEX = -1, PFONT_COMPLEX = -2,
                 PFONT_TRIPLEX = -3, PFONT_ITALIC_COMPLEX = -4, PFONT_ITALIC_TRIPLEX = -5,
                  PFONT_SCRIPT_SIMPLEX = -6, PFONT_SCRIPT_COMPLEX = -7 }

enum __hlhsrid = { PHIGS_HLHSR_ID_OFF = 0, PHIGS_HLHSR_ID_ON = 1 }

enum __hlhsrmode = {
 PHIGS_HLHSR_MODE_NONE = 0, PHIGS_HLHSR_MODE_ZBUFF = 1,  PHIGS_HLHSR_MODE_PAINTERS = 2,
 PHIGS_HLHSR_MODE_SCANLINE = 3, PHIGS_HLHSR_MODE_LINE_ONLY = 4, PHIGS_HLHSR_MODE_ZBUFF_NO_ID = 5 }

enum __intershading = { PSD_NONE = 1, PSD_COLOUR = 2, PSD_DOT_PRODUCT = 3, PSD_NORMAL = 4 }

enum __lighttypes = { PLIGHT_AMBIENT = 1, PLIGHT_DIRECTIONAL = 2, PLIGHT_POSITIONAL = 3, PLIGHT_SPOT = 4 }

enum __lineshading = { PSD_NONE = 1, PSD_COLOUR = 2 }

enum __linetype = {  PLINE_SOLID = 1, PLINE_DASH = 2, PLINE_DOT = 3, PLINE_DASH_DOT = 4 }
enum __linetypeSUN = {  PLINE_LONG_DASH = 0, PLINE_DOT_DASH_DOT = -1, PLINE_CENTER = -2,
                              PLINE_PHANTOM = -3 }

enum __markertype = {  PMARKER_DOT = 1, PMARKER_PLUS = 2, PMARKER_ASTERISK = 3, PMARKER_CIRCLE = 4,
                               PMARKER_CROSS = 5 }
enum __markertypsSUN = {  PMARKER_SQUARE = 0, PMARKER_BOWTIE_NE = -1, PMARKER_BOWTIE_NW = -2 }

enum __modelclipoeratorp = { PMC_REPLACE = 1, PMC_INTERSECT = 2 }

enum __promptechochoice = {  PCHOICE_DEF = 1, PCHOICE_PR_ECHO = 2, PCHOICE_STRING_PR = 3,
                                     PCHOICE_STRING_IN = 4,   PCHOICE_STRUCT = 5 }

enum __promptecholoc = { PLOC_DEF = 1, PLOC_CROSS_HAIR = 2, PLOC_TRACK_CROSS = 3, PLOC_RUB_BAND = 4,
                                PLOC_RECT = 5, PLOC_DIGIT = 6}

enum __promptechopick = { PPICK_DEF = 1, PPICK_GROUP_HIGHL = 2, PPICK_STRUCT_NETWORK = 3 }

enum __promptechostring = { PSTRING_DEF = 1 }

enum __promptechostroke = {  PSTROKE_DEF = 1, PSTROKE_DIGIT = 2, PSTROKE_MARKER = 3, PSTROKE_LINE = 4 }

enum __prompechovalr = {  PVAL_DEF = 1, PVAL_GRAPH = 2, PVAL_DIGIT = 3 }

enum __reflequation = { PREFL_NONE = 1, PREFL_AMBIENT = 2, PREFL_AMB_DIFF = 3, PREFL_AMB_DIFF_SPEC = 4 }

enum __renderingcolrmodel {  PRCM_WS_DEP = 0, PRCM_RGB = PMODEL_RGB, PRCM_CIE = PMODEL_CIELUV,
                                    PRCM_HSV = PMODEL_HSV,  PRCM_HLS = PMODEL_HLS };

enum __surfaceapproximation = {
 PSURF_WS_DEP = 1,PSURF_CONSTANT_PARAMETRIC_BETWEEN_KNOTS = 2,
```

PSURF_CHORDAL_SIZE_WC = 3, PSURF_CHORDAL_SIZE_NPC = 4, PSURF_CHORDAL_SIZE_DC = 5,
PSURF_PLANAR_DEVIATION_WC = 6 PSURF_PLANAR_DEVIATION_NPC = 7, PSURF_PLANAR_DEVIATION_DC = 8,
PSURF_RELATIVE_WC = 9 PSURF_RELATIVE_NPC = 10, PSURF_RELATIVE_DC = 11 }

enum __surfacetypes = { PSC_NONE = 1, PSC_WS_DEP = 2, PSC_ISOPARAMETRIC_CURVES = 3,
PSC_LEVEL_CURVES_MC = 4, PSC_LEVEL_CURVES_WC = 5 }

enum __vertexdataflags = { PVERT_COORD = 0, PVERT_COORD_COLOUR = 1, PVERT_COORD_NORMAL = 2,
PVERT_COORD_COLOUR_NORMAL = 3 };

Miscellaneous

PHIGS basic workstation types

```
extern    Pint    phigs_ws_type_x_tool;
extern    Pint    phigs_ws_type_x_drawable;
extern    Pint    phigs_ws_type_cgm_out;
extern    Pint    phigs_ws_type_x_drawable_region;
```

Preferred argument to popen_phigs

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