INTERNATIONAL STANDARD

ISO/IEC 9593-4

First edition
1991-12-15

AMENDMENT 2
1998-12-15

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

Part 4:
C

AMENDMENT 2:
Incorporation of PHIGS amendments

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

Partie 4: C

AMENDEMENT 2: Incorporation des amendements de PHIGS

Reference number
ISO/IEC 9593-4:1991/Amd.2:1998 (E)

Foreword

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

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

Amendment 2 to ISO/IEC 9593-4:1991 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

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

Part 4:
C

Amendment 2: Incorporation of PHIGS amendments

Page 2

2 Normative references

The number of the first reference is changed to ISO/IEC 9592-1:1996.

Page 3

3 The C language binding of PHIGS

3.1 Conformance

The following text replaces the content of 3.1:

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

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

a) In order to conform to the *Basic PHIGS profile*, an implementation of the C binding of PHIGS shall implement at least the functionality defined for the *Basic PHIGS profile* as specified in ISO/IEC

9592-1. It shall make visible all of the declarations in the C binding specified in clauses 5, 6, and 7 of this part of ISO/IEC 9593.

b) In order to conform to the *PHIGS PLUS profile*, an implementation of the C binding of PHIGS shall implement at least the functionality defined for the *PHIGS PLUS profile* as specified in ISO/IEC 9592-1. It shall make visible all of the declarations in the C binding specified in clauses 5 through 10 of this part of ISO/IEC 9593.

c) In order to conform to the *Full PHIGS profile*, an implementation of the C binding of PHIGS shall implement all of the functionality defined for the *Full PHIGS profile* as specifed in ISO/IEC 9592-1. It shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 9593.

d) The syntax of the function names shall be precisely as specified in this part of ISO/IEC 9593 and the parameters shall be of the data types stated in this part of ISO/IEC 9593.

Pages 10 to 13

4 Tables

4.2 Table of abbreviations

The following abbreviations are added alphabetically to Table 1

"

Table 1 - Abbreviations ordered alphabetically

Word or Phrase	Abbreviation
acknowledgement	ack
address	addr
associate	assoc
association	assoc
attribute	attr
automatic	auto
background	backg
boundary	bound
buffer	buf
centre	ctr
channel	chan
complete	comp
composition	compos
conditions	conds
configuration	config
continuity	cont
correlate, correlation	corr
define	def
destination	dest
disassociate	disassoc
disposition	dispos

Table 1 - Abbreviations ordered alphabetically (Continued)

Word or Phrase	Abbreviation
distance	dist
frequency	freq
group	grp
groups	grps
heuristics	heur
include	incl
incompatible	incompat
incomplete	incomp
indicator	indic
instance	inst
logical input device	lid
manipulation	manip
measure	meas
non-atomic	na
non-retained	nr
non-retained data	nrd
optimization	opt
parameterization	param
parameters	params
picture	pict
posting	post
process, processing	proc
reference	ref
registered	reg
render, rendering	rend
resource	res
sampling	sampl
source	src
target	targ
texture	textr
transfer	trans
traverse, traversal	trav
trigger	trig
uncompressed	uncomp
undefine	undef
weight	wt
which	NULL

”

Pages 14 to 20

4.3 Function names

4.3.1 List ordered alphabetically by bound name

The following function names are merged alphabetically by bound name in Table 2

“

Table 2 - Function names ordered by bound name

C Name	PHIGS and PHIGS PLUS Name
`passoc_image_res (...)`	ASSOCIATE IMAGE RESOURCE
`passoc_trav_res (...)`	ASSOCIATE TRAVERSAL RESOURCE
`pattach_lid_to_light_src (...)`	ATTACH LOGICAL INPUT DEVICE TO LIGHT SOURCE
`pattach_lid_to_view (...)`	ATTACH LOGICAL INPUT DEVICE TO VIEW
`pcircle (...)`	CIRCLE
`pcircle3 (...)`	CIRCLE 3
`pcircular_arc (...)`	CIRCULAR ARC
`pcircular_arc3 (...)`	CIRCULAR ARC 3
`pcircular_arc_close (...)`	CIRCULAR ARC CLOSE
`pcircular_arc_close3 (...)`	CIRCULAR ARC CLOSE 3
`pclear_targ (...)`	CLEAR TARGET
`pclose_di_struct (...)`	CLOSE DIRECT INTERPRETATION STRUCTURE
`pcond_exec_struct (...)`	CONDITIONAL EXECUTE STRUCTURE
`pcond_inst_struct (...)`	CONDITIONAL INSTANCE STRUCTURE
`pcond_return (...)`	CONDITIONAL RETURN
`pcond_skip_elements (...)`	CONDITIONAL SKIP ELEMENTS
`pcond_skip_to_label (...)`	CONDITIONAL SKIP TO LABEL
`pcopy_elem_struct (...)`	COPY ELEMENT FROM STRUCTURE
`pcopy_elem_range_struct (...)`	COPY ELEMENT RANGE FROM STRUCTURE
`pcopy_elems_between_labels_struct (...)`	COPY ELEMENTS BETWEEN LABELS FROM STRUCTURE
`pcopy_targ (...)`	COPY TARGET
`pcreate_composite_measure (...)`	CREATE COMPOSITE MEASURE
`pcreate_mipmap_texture (...)`	CREATE MIPMAP TEXTURE
`pcreate_set_measure (...)`	CREATE SET MEASURE
`pcreate_targ (...)`	CREATE TARGET
`pdefine_choice (...)`	DEFINE CHOICE
`pdefine_composite (...)`	DEFINE COMPOSITE
`pdefine_linetype (...)`	DEFINE LINETYPE
`pdefine_locator (...)`	DEFINE LOCATOR
`pdefine_marker_type (...)`	DEFINE MARKER TYPE
`pdefine_pick (...)`	DEFINE PICK
`pdefine_post_grp (...)`	DEFINE POSTING GROUP
`pdefine_set (...)`	DEFINE SET
`pdefine_string (...)`	DEFINE STRING
`pdefine_stroke (...)`	DEFINE STROKE

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
`pdefine_valuator (...)`	DEFINE VALUATOR
`pdestroy_composite_measure (...)`	DESTROY COMPOSITE MEASURE
`pdestroy_set_measure (...)`	DESTROY SET MEASURE
`pdestroy_targ (...)`	DESTROY TARGET
`pdetach_lid_from_light_src (...)`	DETACH LOGICAL INPUT DEVICE FROM LIGHT SOURCE
`pdetach_lid_from_view (...)`	DETACH LOGICAL INPUT DEVICE FROM VIEW
`pdisable_di_pick (...)`	DISABLE DIRECT INTERPRETATION PICK
`pdisassoc_image_res (...)`	DISASSOCIATE IMAGE RESOURCE
`pdisassoc_trav_res (...)`	DISASSOCIATE TRAVERSAL RESOURCE
`pellipse (...)`	ELLIPSE
`pellipse3 (...)`	ELLIPSE 3
`pelliptical_arc (...)`	ELLIPTICAL ARC
`pelliptical_arc3 (...)`	ELLIPTICAL ARC 3
`pelliptical_arc_close (...)`	ELLIPTICAL ARC CLOSE
`pelliptical_arc_close3 (...)`	ELLIPTICAL ARC CLOSE 3
`penable_di_pick (...)`	ENABLE DIRECT INTERPRETATION PICK
`pend_watch_on_elem_range (...)`	END WATCH ON ELEMENT RANGE
`pfill_circle (...)`	FILL CIRCLE
`pfill_circle3 (...)`	FILL CIRCLE 3
`pfill_ellipse (...)`	FILL ELLIPSE
`pfill_ellipse3 (...)`	FILL ELLIPSE 3
`pget_composite (...)`	GET COMPOSITE
`pget_composite3 (...)`	GET COMPOSITE 3
`pget_set (...)`	GET SET
`pget_set3 (...)`	GET SET 3
`pinit_composite (...)`	INITIALIZE COMPOSITE
`pinit_composite3 (...)`	INITIALIZE COMPOSITE 3
`pinit_di_pick (...)`	INITIALIZE DIRECT INTERPRETATION PICK
`pinit_di_pick3 (...)`	INITIALIZE DIRECT INTERPRETATION PICK 3
`pinit_set (...)`	INITIALIZE SET
`pinit_set3 (...)`	INITIALIZE SET 3
`pinq_alpha_facs (...)`	INQUIRE ALPHA FACILITIES
`pinq_appl_filter (...)`	INQUIRE APPLICATION FILTER
`pinq_assoc_image_res (...)`	INQUIRE ASSOCIATED IMAGE RESOURCES
`pinq_back_clip_indicator (...)`	INQUIRE BACK CLIPPING INDICATOR
`pinq_back_plane_dist (...)`	INQUIRE BACK PLANE DISTANCE
`pinq_choice_facs (...)`	INQUIRE CHOICE FACILITIES
`pinq_composite_facs (...)`	INQUIRE COMPOSITE FACILITIES
`pinq_composite_meas_def (...)`	INQUIRE COMPOSITE MEASURE DEFINITION
`pinq_composite_st (...)`	INQUIRE COMPOSITE DEVICE STATE
`pinq_composite_st3 (...)`	INQUIRE COMPOSITE DEVICE STATE 3
`pinq_cond_trav_facs (...)`	INQUIRE CONDITIONAL TRAVERSAL FACILITIES
`pinq_config_setting_facs (...)`	INQUIRE CONFIGURATION SETTING FACILITIES
`pinq_dc_clip_regions (...)`	INQUIRE DEVICE COORDINATE CLIP REGIONS
`pinq_dc_clip_regions3 (...)`	INQUIRE DEVICE COORDINATE CLIP REGIONS 3

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
pinq_dc_clip_regions_facs(...)	INQUIRE DEVICE COORDINATE CLIP REGIONS FACILITIES
pinq_def_composite_data (...)	INQUIRE DEFAULT COMPOSITE DEVICE DATA
pinq_def_composite_data3 (...)	INQUIRE DEFAULT COMPOSITE DEVICE DATA 3
pinq_def_di_pick_data (...)	INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA
pinq_def_di_pick_data3 (...)	INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA 3
pinq_def_set_data (...)	INQUIRE DEFAULT SET DEVICE DATA
pinq_def_set_data3 (...)	INQUIRE DEFAULT SET DEVICE DATA 3
pinq_def_targ_disp (...)	INQUIRE DEFAULT TARGET DISPOSITION
pinq_di_mode (...)	INQUIRE DIRECT INTERPRETATION MODE
pinq_di_pick_corr_point (...)	INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT
pinq_di_pick_corr_point3 (...)	INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT 3
pinq_di_pick_set_status (...)	INQUIRE DIRECT INTERPRETATION PICK SET STATUS
pinq_di_pick_st (...)	INQUIRE DIRECT INTERPRETATION PICK STATE
pinq_di_pick_st3 (...)	INQUIRE DIRECT INTERPRETATION PICK STATE 3
pinq_disp_targ (...)	INQUIRE DISPLAY TARGET
pinq_di_trav_facs (...)	INQUIRE DIRECT INTERPRETATION TRAVERSAL FACILITIES
pinq_dyns_post_grps (...)	INQUIRE DYNAMICS OF POSTING GROUPS
pinq_dyns_ws_attrs_texture (...)	INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES TEXTURE
pinq_edge_rep_full (...)	INQUIRE EDGE REPRESENTATION FULL
pinq_ext_pat_facs (...)	INQUIRE EXTENDED PATTERN FACILITIES
pinq_ext_pat_rep (...)	INQUIRE EXTENDED PATTERN REPRESENTATION
pinq_front_clip_indicator (...)	INQUIRE FRONT CLIPPING INDICATOR
pinq_front_plane_dist (...)	INQUIRE FRONT PLANE DISTANCE
pinq_grps_posted (...)	INQUIRE SET OF GROUPS TO WHICH POSTED
pinq_highl_facs (...)	INQUIRE HIGHLIGHTING FACILITIES
pinq_highl_rep (...)	INQUIRE HIGHLIGHTING REPRESENTATION
pinq_image_res (...)	INQUIRE IMAGE RESOURCE
pinq_image_res_facs (...)	INQUIRE IMAGE RESOURCE FACILITIES
pinq_lid_attached_to_light_src (...)	INQUIRE LOGICAL INPUT DEVICE ATTACHED TO LIGHT SOURCE
pinq_lid_attached_to_view (...)	INQUIRE LOGICAL INPUT DEVICE ATTACHED TO VIEW
pinq_lid_def (...)	INQUIRE LOGICAL INPUT DEVICE DEFINITION
pinq_line_rep_full (...)	INQUIRE POLYLINE REPRESENTATION FULL
pinq_linetype_def (...)	INQUIRE LINETYPE DEFINITION
pinq_linetype_def_facs (...)	INQUIRE LINETYPE DEFINITION FACILITIES
pinq_linetype_def_support (...)	INQUIRE LINETYPE DEFINITION SUPPORT
pinq_list_def_appl_filters (...)	INQUIRE LIST OF DEFINED APPLICATION FILTERS
pinq_list_highl_inds (...)	INQUIRE LIST OF HIGHLIGHTING INDICES

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
pinq_list_post_grps (...)	INQUIRE POSTED STRUCTURES FROM POSTING GROUP
pinq_list_texture_inds (...)	INQUIRE LIST OF TEXTURE INDICES
pinq_lists_avail_lids (...)	INQUIRE LISTS OF AVAILABLE LOGICAL INPUT DEVICES
pinq_loc_facs (...)	INQUIRE LOCATOR FACILITIES
pinq_marker_type_def (...)	INQUIRE MARKER TYPE DEFINITION
pinq_marker_type_def_facs (...)	INQUIRE MARKER TYPE DEFINITION FACILITIES
pinq_marker_type_def_support (...)	INQUIRE MARKER TYPE DEFINITION SUPPORT
pinq_mipmap_facs (...)	INQUIRE MIPMAP FACILITIES
pinq_num_avail_na_in (...)	INQUIRE NUMBER OF AVAILABLE NON-ATOMIC LOGICAL INPUT DEVICES
pinq_num_def_linetypes (...)	INQUIRE NUMBER OF DEFINED LINETYPES
pinq_num_def_marker_types (...)	INQUIRE NUMBER OF DEFINED MARKER TYPES
pinq_num_pred_appl_filters (...)	INQUIRE NUMBER OF PREDEFINED APPLICATION FILTERS
pinq_pick_facs (...)	INQUIRE PICK FACILITIES
pinq_pick_mapping_facs (...)	INQUIRE PICK MAPPING FACILITIES
pinq_pick_mapping_state (...)	INQUIRE PICK MAPPING STATE
pinq_pict_status (...)	INQUIRE PICTURE STATUS
pinq_posted_di_struct (...)	INQUIRE POSTED DIRECT INTERPRETATION STRUCTURE
pinq_posted_structs_from_post_grp (...)	INQUIRE POSTED STRUCTURES FROM POSTING GROUP
pinq_post_grp (...)	INQUIRE POSTING GROUP
pinq_post_grp_facs (...)	INQUIRE POSTING GROUP FACILITIES
pinq_pred_appl_filter (...)	INQUIRE PREDEFINED APPLICATION FILTER
pinq_pred_assoc_targ_trav_res (...)	INQUIRE PREDEFINED ASSOCIATION OF TARGET WITH TRAVERSAL RESOURCES
pinq_pred_assoc_trav_res_targ (...)	INQUIRE PREDEFINED ASSOCIATION OF TRAVERSAL RESOURCE WITH TARGET
pinq_pred_composite_meas_def (...)	INQUIRE PREDEFINED COMPOSITE MEASURE DEFINITION
pinq_pred_edge_rep_full (...)	INQUIRE PREDEFINED EDGE REPRESENTATION FULL
pinq_pred_ext_pat_rep (...)	INQUIRE PREDEFINED EXTENDED PATTERN REPRESENTATION
pinq_pred_highl_rep (...)	INQUIRE PREDEFINED HIGHLIGHTING REPRESENTATION
pinq_pred_image_res (...)	INQUIRE PREDEFINED IMAGE RESOURCE
pinq_pred_line_rep_full (...)	INQUIRE PREDEFINED POLYLINE REPRESENTATION FULL
pinq_pred_linetype_def (...)	INQUIRE PREDEFINED LINETYPE DEFINITION
pinq_pred_marker_type_def (...)	INQUIRE PREDEFINED MARKER TYPE DEFINITION
pinq_pred_post_grp (...)	INQUIRE PREDEFINED POSTING GROUP
pinq_pred_set_meas_def (...)	INQUIRE PREDEFINED SET MEASURE DEFINITION
pinq_pred_texture_rep (...)	INQUIRE PREDEFINED TEXTURE REPRESENTATION

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
`pinq_proj_ref_point (...)`	INQUIRE PROJECTION REFERENCE POINT
`pinq_proj_type (...)`	INQUIRE PROJECTION TYPE
`pinq_proj_vp (...)`	INQUIRE PROJECTION VIEWPORT
`pinq_proj_vp3 (...)`	INQUIRE PROJECTION VIEWPORT 3
`pinq_rend_targ (...)`	INQUIRE RENDERING TARGET
`pinq_set_facs (...)`	INQUIRE SET FACILITIES
`pinq_set_meas_def (...)`	INQUIRE SET MEASURE DEFINITION
`pinq_set_st (...)`	INQUIRE SET DEVICE STATE
`pinq_set_st3 (...)`	INQUIRE SET DEVICE STATE 3
`pinq_string_facs (...)`	INQUIRE STRING FACILITIES
`pinq_stroke_facs (...)`	INQUIRE STROKE FACILITIES
`pinq_targ_assoc_trav_res (...)`	INQUIRE TARGET ASSOCIATED WITH TRAVERSAL RESOURCE
`pinq_targ_dispos (...)`	INQUIRE TARGET DISPOSITION
`pinq_targ_facs (...)`	INQUIRE TARGET FACILITIES
`pinq_targ_manip_mode (...)`	INQUIRE TARGET MANIPULATION MODE
`pinq_targ_st (...)`	INQUIRE TARGET STATE
`pinq_texture_facs (...)`	INQUIRE TEXTURE FACILITIES
`pinq_texture_map_facs (...)`	INQUIRE TEXTURE MAPPING FACILITIES
`pinq_texture_rep (...)`	INQUIRE TEXTURE REPRESENTATION
`pinq_transparency_mode (...)`	INQUIRE TRANSPARENCY MODE
`pinq_transparency_thresholds (...)`	INQUIRE TRANSPARENCY THRESHOLDS
`pinq_trav_res_assoc_targ (...)`	INQUIRE TRAVERSAL RESOURCES ASSOCIATED WITH TARGET
`pinq_trav_res_facs (...)`	INQUIRE TRAVERSAL RESOURCE FACILITIES
`pinq_val_facs (...)`	INQUIRE VALUATOR FACILITIES
`pinq_view_plane_dist (...)`	INQUIRE VIEW PLANE DISTANCE
`pinq_view_plane_norm (...)`	INQUIRE VIEW PLANE NORMAL
`pinq_view_ref_point (...)`	INQUIRE VIEW REFERENCE POINT
`pinq_view_ref_point3 (...)`	INQUIRE VIEW REFERENCE POINT 3
`pinq_view_status (...)`	INQUIRE VIEW STATUS
`pinq_view_up_vec (...)`	INQUIRE VIEW UP VECTOR
`pinq_view_up_vec3 (...)`	INQUIRE VIEW UP VECTOR 3
`pinq_view_win_limits (...)`	INQUIRE VIEW WINDOW LIMITS
`pinq_watch_elem_range (...)`	INQUIRE WATCH ON ELEMENT RANGE
`pinq_watch_range_st (...)`	INQUIRE WATCH RANGE STATE
`pinq_wss_di_struct_posted (...)`	INQUIRE SET OF WORKSTATIONS TO WHICH DIRECT INTERPRETATION STRUCTURE POSTED
`pinq_ws_st_table_highl (...)`	INQUIRE WORKSTATION STATE TABLE LENGTHS HIGHLIGHTING
`pinq_ws_st_table_texture (...)`	INQUIRE WORKSTATION STATE TABLE LENGTHS TEXTURE
`pinq_xy_clip_indicator (...)`	INQUIRE X-Y CLIPPING INDICATOR
`pinst_struct (...)`	INSTANCE STRUCTURE
`pmanip_trav_res (...)`	MANIPULATE TRAVERSAL RESOURCE
`pmap_dc_point_to_pick_paths (...)`	MAP DEVICE COORDINATE POINT TO PICK PATHS

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
pmap_dc_to_wc (...)	MAP DEVICE COORDINATES TO WORLD COORDINATES
pmap_dc_to_wsc (...)	MAP DEVICE COORDINATES TO WINDOW SYSTEM COORDINATES
pmap_wsc_to_dc (...)	MAP WINDOW SYSTEM COORDINATES TO DEVICE COORDINATES
pmark_multi_pass_compl (...)	MARK MULTI-PASS COMPLETION
pmark_multi_pass_start (...)	MARK MULTI-PASS START
pmark_pass_compl (...)	MARK PASS COMPLETION
pmark_pass_start (...)	MARK PASS START
pmove_elem_struct (...)	MOVE ELEMENT FROM STRUCTURE
pmove_elem_range_struct (...)	MOVE ELEMENT RANGE FROM STRUCTURE
pmove_elems_between_labels_struct (...)	MOVE ELEMENTS BETWEEN LABELS FROM STRUCTURE
popen_di_struct (...)	OPEN DIRECT INTERPRETATION STRUCTURE
ppop_st (...)	POP STATE
ppost_di_struct (...)	POST DIRECT INTERPRETATION STRUCTURE
ppost_struct_to_grp (...)	POST STRUCTURE TO GROUP
ppush_st (...)	PUSH STATE
predraw_all_from_grp_on_targ (	REDRAW ALL STRUCTURES FROM POSTING GROUP ON TARGET
predraw_all_structs_from_grp (...)	REDRAW ALL STRUCTURES FROM POSTING GROUP
predraw_all_structs_on_targ (...)	REDRAW ALL STRUCTURES ON TARGET
prenew_di_state (...)	RENEW DIRECT INTERPRETATION STATE
preq_composite (...)	REQUEST COMPOSITE
preq_composite3 (...)	REQUEST COMPOSITE 3
preq_set (...)	REQUEST SET
preq_set3 (...)	REQUEST SET 3
preset_all_trav_res (...)	RESET ALL TRAVERSAL RESOURCES
pret_num_passes_req (...)	RETRIEVE NUMBER OF PASSES REQUIRED
pret_window_system_colr (...)	RETRIEVE WINDOW SYSTEM COLOUR
psample_composite (...)	SAMPLE COMPOSITE
psample_composite3 (...)	SAMPLE COMPOSITE 3
psample_set (...)	SAMPLE SET
psample_set3 (...)	SAMPLE SET 3
pset_active_textures (...)	SET ACTIVE TEXTURES
pset_alpha_data_sel_ind (...)	SET ALPHA DATA SELECTION INDEX
pset_alpha_src_sel (...)	SET ALPHA SOURCE SELECTOR
pset_appl_filter (...)	SET APPLICATION FILTER
pset_appl_int (...)	SET APPLICATION INTEGER
pset_appl_real (...)	SET APPLICATION REAL
pset_back_active_textures (...)	SET BACK ACTIVE TEXTURES
pset_back_clip_indicator (...)	SET BACK CLIPPING INDICATOR
pset_back_plane_dist (...)	SET BACK PLANE DISTANCE
pset_back_transparency (...)	SET BACK TRANSPARENCY
pset_composite_mode (...)	SET COMPOSITE MODE

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
`pset_composite_pick_filter (...)`	SET COMPOSITE PICK FILTER
`pset_cond_flags (...)`	SET CONDITION FLAGS
`pset_cond_flags_from_tests (...)`	SET CONDITION FLAGS FROM TESTS
`pset_dc_clip_regions (...)`	SET DEVICE COORDINATE CLIP REGIONS
`pset_dc_clip_regions3 (...)`	SET DEVICE COORDINATE CLIP REGIONS 3
`pset_depth_cue_rep_mask (`	SET DEPTH CUE REPRESENTATION MASK
`pset_di_mode (...)`	SET DIRECT INTERPRETATION MODE
`pset_di_pick_corr_point (...)`	SET DIRECT INTERPRETATION PICK CORRELATION POINT
`pset_di_pick_corr_point3 (...)`	SET DIRECT INTERPRETATION PICK CORRELATION POINT 3
`pset_di_pick_filter (...)`	SET DIRECT INTERPRETATION PICK FILTER
`pset_disp_targ (...)`	SET DISPLAY TARGET
`pset_edgecap (...)`	SET EDGECAP
`pset_edgejoin (...)`	SET EDGEJOIN
`pset_edgemitre_limit (...)`	SET EDGEMITRE LIMIT
`pset_edge_rep_full (...)`	SET EDGE REPRESENTATION FULL
`pset_edge_rep_mask (...)`	SET EDGE REPRESENTATION MASK
`pset_edgetype_adapt (...)`	SET EDGETYPE ADAPTABILITY
`pset_edgetype_cont (...)`	SET EDGETYPE CONTINUITY
`pset_edgetype_offset (...)`	SET EDGETYPE OFFSET
`pset_ext_pat_rep (...)`	SET EXTENDED PATTERN REPRESENTATION
`pset_front_clip_indicator (...)`	SET FRONT CLIPPING INDICATOR
`pset_front_plane_dist (...)`	SET FRONT PLANE DISTANCE
`pset_highl_ind (...)`	SET HIGHLIGHTING INDEX
`pset_highl_method (...)`	SET HIGHLIGHTING METHOD
`pset_highl_rep (...)`	SET HIGHLIGHTING REPRESENTATION
`pset_int_rep_mask (...)`	SET INTERIOR REPRESENTATION MASK
`pset_linecap (...)`	SET LINECAP
`pset_linejoin (...)`	SET LINEJOIN
`pset_linemitre_limit (...)`	SET LINEMITRE LIMIT
`pset_line_rep_full (...)`	SET POLYLINE REPRESENTATION FULL
`pset_line_rep_mask (...)`	SET POLYLINE REPRESENTATION MASK
`pset_linetype_adapt (...)`	SET LINETYPE ADAPTABILITY
`pset_linetype_cont (...)`	SET LINETYPE CONTINUITY
`pset_linetype_offset (...)`	SET LINETYPE OFFSET
`pset_marker_rep_mask (...)`	SET POLYMARKER REPRESENTATION MASK
`pset_pat_rep_mask (...)`	SET PATTERN REPRESENTATION MASK
`pset_pick_mapping_data (...)`	SET PICK MAPPING DATA
`pset_post_grp_backg_method (...)`	SET POSTING GROUP BACKGROUND METHOD
`pset_post_grp_backg_style (...)`	SET POSTING GROUP BACKGROUND STYLE
`pset_post_grp_border_ind (...)`	SET POSTING GROUP BORDER INDEX
`pset_post_grp_border_indicator (...)`	SET POSTING GROUP BORDER INDICATOR
`pset_post_grp_priority (...)`	SET POSTING GROUP PRIORITY
`pset_post_grp_status (...)`	SET POSTING GROUP STATUS
`pset_proj_ref_point (...)`	SET PROJECTION REFERENCE POINT
`pset_proj_type (...)`	SET PROJECTION TYPE

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
pset_proj_vp (...)	SET PROJECTION VIEWPORT
pset_proj_vp3 (...)	SET PROJECTION VIEWPORT 3
pset_refl_rep_mask (...)	SET REFLECTANCE REPRESENTATION MASK
pset_rend_targ (...)	SET RENDERING TARGET
pset_set_mode (...)	SET SET MODE
pset_set_pick_filter (...)	SET SET PICK FILTER
pset_st_visual_rep (...)	SET STATE OF VISUAL REPRESENTATION
pset_targ_dispos (...)	SET TARGET DISPOSITION
pset_targ_manip_mode (...)	SET TARGET MANIPULATION MODE
pset_targ_st_visual_rep (...)	SET TARGET STATE OF VISUAL REPRESENTATION
pset_text_rep_mask (...)	SET TEXT REPRESENTATION MASK
pset_texture_binding (...)	SET TEXTURE BINDING
pset_texture_composition (...)	SET TEXTURE COMPOSITION
pset_texture_param (...)	SET TEXTURE PARAMETRIZATION
pset_texture_perspect_corr (...)	SET TEXTURE PERSPECTIVE CORRECTION
pset_texture_rep (...)	SET TEXTURE REPRESENTATION
pset_texture_rep_mask (...)	SET TEXTURE REPRESENTATION MASK
pset_texture_res_opt_heur (...)	SET TEXTURE RESOURCE OPTIMIZATION HEURISTICS
pset_texture_sampling (...)	SET TEXTURE SAMPLING
pset_texture_sampling_freq (...)	SET TEXTURE SAMPLING FREQUENCY
pset_transparency (...)	SET TRANSPARENCY
pset_transparency_mode (...)	SET TRANSPARENCY MODE
pset_transparency_thresholds (...)	SET TRANSPARENCY THRESHOLDS
pset_view_plane_dist (...)	SET VIEW PLANE DISTANCE
pset_view_plane_norm (...)	SET VIEW PLANE NORMAL
pset_view_ref_point (...)	SET VIEW REFERENCE POINT
pset_view_ref_point3 (...)	SET VIEW REFERENCE POINT 3
pset_view_up_vec (...)	SET VIEW UP VECTOR
pset_view_up_vec3 (...)	SET VIEW UP VECTOR 3
pset_view_win_limits (...)	SET VIEW WINDOW LIMITS
pset_watch_on_elem_range (...)	SET WATCH ON ELEMENT RANGE
pset_xy_clip_indicator (...)	SET X-Y CLIPPING INDICATOR
ptrans_di_pick_set (...)	TRANSFER DIRECT INTERPRETATION PICK SET
pundefine_choice (...)	UNDEFINE CHOICE
pundefine_composite (...)	UNDEFINE COMPOSITE
pundefine_locator (...)	UNDEFINE LOCATOR
pundefine_pick (...)	UNDEFINE PICK
pundefine_post_grp (...)	UNDEFINE POSTING GROUP
pundefine_set (...)	UNDEFINE SET
pundefine_string (...)	UNDEFINE STRING
pundefine_stroke (...)	UNDEFINE STROKE
pundefine_valuator (...)	UNDEFINE VALUATOR
punpost_all_structs_from_grp (...)	UNPOST ALL STRUCTURES FROM GROUP
punpost_di_struct (...)	UNPOST DIRECT INTERPRETATION STRUCTURE
punpost_structs_from_grps (...)	UNPOST STRUCTURES FROM GROUPS

Table 2 - Function names ordered by bound name (Continued)

C Name	PHIGS and PHIGS PLUS Name
pupd_targ (...)	UPDATE TARGET
pupd_view_rep (...)	UPDATE VIEW REPRESENTATION
pws_type_create (...)	WORKSTATION TYPE CREATE
pws_type_destroy (...)	WORKSTATION TYPE DESTROY
pws_type_get (...)	WORKSTATION TYPE GET
pws_type_set (...)	WORKSTATION TYPE SET

”

Pages 20 to 26

4.3.2 List ordered alphabetically by PHIGS and PHIGS PLUS name

The following function names are merged alphabetically by function name in Table 3:

“

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name

PHIGS and PHIGS PLUS Name	C Name
ASSOCIATE IMAGE RESOURCE	passoc_image_res (...)
ASSOCIATE TRAVERSAL RESOURCE	passoc_trav_res (...)
ATTACH LOGICAL INPUT DEVICE TO LIGHT SOURCE	pattach_lid_to_light_src (...)
ATTACH LOGICAL INPUT DEVICE TO VIEW	pattach_lid_to_view (...)
CIRCLE	pcircle (...)
CIRCLE 3	pcircle3 (...)
CIRCULAR ARC	pcircular_arc (...)
CIRCULAR ARC 3	pcircular_arc3 (...)
CIRCULAR ARC CLOSE	pcircular_arc_close (...)
CIRCULAR ARC CLOSE 3	pcircular_arc_close3 (...)
CLEAR TARGET	pclear_targ (...)
CLOSE DIRECT INTERPRETATION STRUCTURE	pclose_di_struct (...)
CONDITIONAL EXECUTE STRUCTURE	pcond_exec_struct (...)
CONDITIONAL INSTANCE STRUCTURE	pcond_inst_struct (...)
CONDITIONAL RETURN	pcond_return (...)
CONDITIONAL SKIP ELEMENTS	pcond_skip_elements (...)
CONDITIONAL SKIP TO LABEL	pcond_skip_to_label (...)
COPY ELEMENT FROM STRUCTURE	pcopy_elem_struct (...)
COPY ELEMENT RANGE FROM STRUCTURE	pcopy_elem_range_struct (...)
COPY ELEMENTS BETWEEN LABELS FROM STRUCTURE	pcopy_elems_between_labels_struct (...)
COPY TARGET	pcopy_targ (...)
CREATE COMPOSITE MEASURE	pcreate_composite_measure (...)
CREATE MIPMAP TEXTURE	pcreate_mipmap_texture (...)
CREATE SET MEASURE	pcreate_set_measure (...)
CREATE TARGET	pcreate_targ (...)
DEFINE CHOICE	pdefine_choice (...)
DEFINE COMPOSITE	pdefine_composite (...)
DEFINE LINETYPE	pdefine_linetype (...)
DEFINE LOCATOR	pdefine_locator (...)
DEFINE MARKER TYPE	pdefine_marker_type (...)
DEFINE PICK	pdefine_pick (...)
DEFINE POSTING GROUP	pdefine_post_grp (...)
DEFINE SET	pdefine_set (...)
DEFINE STRING	pdefine_string (...)
DEFINE STROKE	pdefine_stroke (...)
DEFINE VALUATOR	pdefine_valuator (...)
DESTROY COMPOSITE MEASURE	pdestroy_composite_measure (...)
DESTROY SET MEASURE	pdestroy_set_measure (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
DESTROY TARGET	pdestroy_targ (...)
DETACH LOGICAL INPUT DEVICE FROM LIGHT SOURCE	pdetach_lid_from_light_src (...)
DETACH LOGICAL INPUT DEVICE FROM VIEW	pdetach_lid_from_view (...)
DISABLE DIRECT INTERPRETATION PICK	pdisable_di_pick (...)
DISASSOCIATE IMAGE RESOURCE	pdisassoc_image_res (...)
DISASSOCIATE TRAVERSAL RESOURCE	pdisassoc_trav_res (...)
ELLIPSE	pellipse (...)
ELLIPSE 3	pellipse3 (...)
ELLIPTICAL ARC	pelliptical_arc (...)
ELLIPTICAL ARC 3	pelliptical_arc3 (...)
ELLIPTICAL ARC CLOSE	pelliptical_arc_close (...)
ELLIPTICAL ARC CLOSE 3	pelliptical_arc_close3 (...)
ENABLE DIRECT INTERPRETATION PICK	penable_di_pick (...)
END WATCH ON ELEMENT RANGE	pend_watch_on_elem_range (...)
FILL CIRCLE	pfill_circle (...)
FILL CIRCLE 3	pfill_circle3 (...)
FILL ELLIPSE	pfill_ellipse (...)
FILL ELLIPSE 3	pfill_ellipse3 (...)
GET COMPOSITE	pget_composite (...)
GET COMPOSITE 3	pget_composite3 (...)
GET SET	pget_set (...)
GET SET 3	pget_set3 (...)
INITIALIZE COMPOSITE	pinit_composite (...)
INITIALIZE COMPOSITE 3	pinit_composite3 (...)
INITIALIZE DIRECT INTERPRETATION PICK	pinit_di_pick (...)
INITIALIZE DIRECT INTERPRETATION PICK 3	pinit_di_pick3 (...)
INITIALIZE SET	pinit_set (...)
INITIALIZE SET 3	pinit_set3 (...)
INQUIRE ALPHA FACILITIES	pinq_alpha_facs (...)
INQUIRE APPLICATION FILTER	pinq_appl_filter (...)
INQUIRE ASSOCIATED IMAGE RESOURCES	pinq_assoc_image_res (...)
INQUIRE BACK CLIPPING INDICATOR	pinq_back_clip_indicator (...)
INQUIRE BACK PLANE DISTANCE	pinq_back_plane_dist (...)
INQUIRE CHOICE FACILITIES	pinq_choice_facs (...)
INQUIRE COMPOSITE DEVICE STATE	pinq_composite_st (...)
INQUIRE COMPOSITE DEVICE STATE 3	pinq_composite_st3 (...)
INQUIRE COMPOSITE FACILITIES	pinq_composite_facs (...)
INQUIRE COMPOSITE MEASURE DEFINITION	pinq_composite_meas_def (...)
INQUIRE CONDITIONAL TRAVERSAL FACILITIES	pinq_cond_trav_facs (...)
INQUIRE CONFIGURATION SETTING FACILITIES	pinq_config_setting_facs (...)
INQUIRE DEFAULT COMPOSITE DEVICE DATA	pinq_def_composite_data (...)
INQUIRE DEFAULT COMPOSITE DEVICE DATA 3	pinq_def_composite_data3 (...)
INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA	pinq_def_di_pick_data (...)
INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA 3	pinq_def_di_pick_data3 (...)
INQUIRE DEFAULT SET DEVICE DATA	pinq_def_set_data (...)
INQUIRE DEFAULT SET DEVICE DATA 3	pinq_def_set_data3 (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
INQUIRE DEFAULT TARGET DISPOSITION	pinq_def_targ_disp (...)
INQUIRE DEVICE COORDINATE CLIP REGIONS	pinq_dc_clip_regions (...)
INQUIRE DEVICE COORDINATE CLIP REGIONS 3	pinq_dc_clip_regions3 (...)
INQUIRE DEVICE COORDINATE CLIP REGIONS FACILITIES	pinq_dc_clip_regions_facs(...)
INQUIRE DIRECT INTERPRETATION MODE	pinq_di_mode (...)
INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT	pinq_di_pick_corr_point (...)
INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT 3	pinq_di_pick_corr_point3 (...)
INQUIRE DIRECT INTERPRETATION PICK SET STATUS	pinq_di_pick_set_status (...)
INQUIRE DIRECT INTERPRETATION PICK STATE	pinq_di_pick_st (...)
INQUIRE DIRECT INTERPRETATION PICK STATE 3	pinq_di_pick_st3 (...)
INQUIRE DIRECT INTERPRETATION TRAVERSAL FACILITIES	pinq_di_trav_facs (...)
INQUIRE DISPLAY TARGET	pinq_disp_targ (...)
INQUIRE DYNAMICS OF POSTING GROUPS	pinq_dyns_post_grps (...)
INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES TEXTURE	pinq_dyns_ws_attrs_texture (...)
INQUIRE EDGE REPRESENTATION FULL	pinq_edge_rep_full (...)
INQUIRE EXTENDED PATTERN FACILITIES	pinq_ext_pat_facs (...)
INQUIRE EXTENDED PATTERN REPRESENTATION	pinq_ext_pat_rep (...)
INQUIRE FRONT CLIPPING INDICATOR	pinq_front_clip_indicator (...)
INQUIRE FRONT PLANE DISTANCE	pinq_front_plane_dist (...)
INQUIRE HIGHLIGHTING FACILITIES	pinq_highl_facs (...)
INQUIRE HIGHLIGHTING REPRESENTATION	pinq_highl_rep (...)
INQUIRE IMAGE RESOURCE	pinq_image_res (...)
INQUIRE IMAGE RESOURCE FACILITIES	pinq_image_res_facs (...)
INQUIRE LINETYPE DEFINITION	pinq_linetype_def (...)
INQUIRE LINETYPE DEFINITION FACILITIES	pinq_linetype_def_facs (...)
INQUIRE LINETYPE DEFINITION SUPPORT	pinq_linetype_def_support (...)
INQUIRE LIST OF DEFINED APPLICATION FILTERS	pinq_list_def_appl_filters (...)
INQUIRE LIST OF HIGHLIGHTING INDICES	pinq_list_highl_inds (...)
INQUIRE LIST OF POSTING GROUPS	pinq_list_post_grps (...)
INQUIRE LIST OF TEXTURE INDICES	pinq_list_texture_inds (...)
INQUIRE LISTS OF AVAILABLE LOGICAL INPUT DEVICES	pinq_lists_avail_lids (...)
INQUIRE LOCATOR FACILITIES	pinq_loc_facs (...)
INQUIRE LOGICAL INPUT DEVICE ATTACHED TO LIGHT SOURCE	pinq_lid_attached_to_light_src (...)
INQUIRE LOGICAL INPUT DEVICE ATTACHED TO VIEW	pinq_lid_attached_to_view (...)
INQUIRE LOGICAL INPUT DEVICE DEFINITION	pinq_lid_def (...)
INQUIRE MARKER TYPE DEFINITION	pinq_marker_type_def (...)
INQUIRE MARKER TYPE DEFINITION FACILITIES	pinq_marker_type_def_facs (...)
INQUIRE MARKER TYPE DEFINITION SUPPORT	pinq_marker_type_def_support (...)
INQUIRE MIPMAP FACILITIES	pinq_mipmap_facs (...)
INQUIRE NUMBER OF AVAILABLE NON-ATOMIC LOGICAL INPUT DEVICES	pinq_num_avail_na_in (...)
INQUIRE NUMBER OF DEFINED LINETYPES	pinq_num_def_linetypes (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
INQUIRE NUMBER OF DEFINED MARKER TYPES	pinq_num_def_marker_types (...)
INQUIRE NUMBER OF PREDEFINED APPLICATION FILTERS	pinq_num_pred_appl_filters (...)
INQUIRE PICK FACILITIES	pinq_pick_facs (...)
INQUIRE PICK MAPPING FACILITIES	pinq_pick_mapping_facs (...)
INQUIRE PICK MAPPING STATE	pinq_pick_mapping_state (...)
INQUIRE PICTURE STATUS	pinq_pict_status (...)
INQUIRE POLYLINE REPRESENTATION FULL	pinq_line_rep_full (...)
INQUIRE POSTED DIRECT INTERPRETATION STRUCTURE	pinq_posted_di_struct (...)
INQUIRE POSTED STRUCTURES FROM POSTING GROUP	pinq_posted_structs_from_post_grp (...)
INQUIRE POSTING GROUP	pinq_post_grp (...)
INQUIRE POSTING GROUP FACILITIES	pinq_post_grp_facs (...)
INQUIRE PREDEFINED APPLICATION FILTER	pinq_pred_appl_filter (...)
INQUIRE PREDEFINED ASSOCIATION OF TARGET WITH TRAVERSAL RESOURCES	pinq_pred_assoc_targ_trav_res (...)
INQUIRE PREDEFINED ASSOCIATION OF TRAVERSAL RESOURCE WITH TARGET	pinq_pred_assoc_trav_res_targ (...)
INQUIRE PREDEFINED COMPOSITE MEASURE DEFINITION	pinq_pred_composite_meas_def (...)
INQUIRE PREDEFINED EDGE REPRESENTATION FULL	pinq_pred_edge_rep_full (...)
INQUIRE PREDEFINED EXTENDED PATTERN REPRESENTATION	pinq_pred_ext_pat_rep (...)
INQUIRE PREDEFINED HIGHLIGHTING REPRESENTATION	pinq_pred_highl_rep (...)
INQUIRE PREDEFINED IMAGE RESOURCE	pinq_pred_image_res (...)
INQUIRE PREDEFINED LINETYPE DEFINITION	pinq_pred_linetype_def (...)
INQUIRE PREDEFINED MARKER TYPE DEFINITION	pinq_pred_marker_type_def (...)
INQUIRE PREDEFINED POLYLINE REPRESENTATION FULL	pinq_pred_line_rep_full (...)
INQUIRE PREDEFINED POSTING GROUP	pinq_pred_post_grp (...)
INQUIRE PREDEFINED SET MEASURE DEFINITION	pinq_pred_set_meas_def (...)
INQUIRE PREDEFINED TEXTURE REPRESENTATION	pinq_pred_texture_rep (...)
INQUIRE PROJECTION REFERENCE POINT	pinq_proj_ref_point (...)
INQUIRE PROJECTION TYPE	pinq_proj_type (...)
INQUIRE PROJECTION VIEWPORT	pinq_proj_vp (...)
INQUIRE PROJECTION VIEWPORT 3	pinq_proj_vp3 (...)
INQUIRE RENDERING TARGET	pinq_rend_targ (...)
INQUIRE SET DEVICE STATE	pinq_set_st (...)
INQUIRE SET DEVICE STATE 3	pinq_set_st3 (...)
INQUIRE SET FACILITIES	pinq_set_facs (...)
INQUIRE SET MEASURE DEFINITION	pinq_set_meas_def (...)
INQUIRE SET OF GROUPS TO WHICH POSTED	pinq_grps_posted (...)
INQUIRE SET OF WORKSTATIONS TO WHICH DIRECT INTERPRETATION STRUCTURE POSTED	pinq_wss_di_struct_posted (...)
INQUIRE STRING FACILITIES	pinq_string_facs (...)
INQUIRE STROKE FACILITIES	pinq_stroke_facs (...)
INQUIRE TARGET ASSOCIATED WITH TRAVERSAL RESOURCE	pinq_targ_assoc_trav_res (...)
INQUIRE TARGET DISPOSITION	pinq_targ_dispos (...)
INQUIRE TARGET FACILITIES	pinq_targ_facs (...)
INQUIRE TARGET MANIPULATION MODE	pinq_targ_manip_mode (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
INQUIRE TARGET STATE	`pinq_targ_st (...)`
INQUIRE TEXTURE FACILITIES	`pinq_texture_facs (...)`
INQUIRE TEXTURE MAPPING FACILITIES	`pinq_texture_map_facs (...)`
INQUIRE TEXTURE REPRESENTATION	`pinq_texture_rep (...)`
INQUIRE TRANSPARENCY MODE	`pinq_transparency_mode (...)`
INQUIRE TRANSPARENCY THRESHOLDS	`pinq_transparency_thresholds (...)`
INQUIRE TRAVERSAL RESOURCE FACILITIES	`pinq_trav_res_facs (...)`
INQUIRE TRAVERSAL RESOURCES ASSOCIATED WITH TARGET	`pinq_trav_res_assoc_targ (...)`
INQUIRE VALUATOR FACILITIES	`pinq_val_facs (...)`
INQUIRE VIEW PLANE DISTANCE	`pinq_view_plane_dist (...)`
INQUIRE VIEW PLANE NORMAL	`pinq_view_plane_norm (...)`
INQUIRE VIEW REFERENCE POINT	`pinq_view_ref_point (...)`
INQUIRE VIEW REFERENCE POINT 3	`pinq_view_ref_point3 (...)`
INQUIRE VIEW STATUS	`pinq_view_status (...)`
INQUIRE VIEW UP VECTOR	`pinq_view_up_vec (...)`
INQUIRE VIEW UP VECTOR 3	`pinq_view_up_vec3 (...)`
INQUIRE VIEW WINDOW LIMITS	`pinq_view_win_limits (...)`
INQUIRE WATCH ON ELEMENT RANGE	`pinq_watch_elem_range (...)`
INQUIRE WATCH RANGE STATE	`pinq_watch_range_st (...)`
INQUIRE WORKSTATION STATE TABLE LENGTHS HIGHLIGHTING	`pinq_ws_st_table_highl (...)`
INQUIRE WORKSTATION STATE TABLE LENGTHS TEXTURE	`pinq_ws_st_table_texture (...)`
INQUIRE X-Y CLIPPING INDICATOR	`pinq_xy_clip_indicator (...)`
INSTANCE STRUCTURE	`pinst_struct (...)`
MANIPULATE TRAVERSAL RESOURCE	`pmanip_trav_res (...)`
MAP DEVICE COORDINATE POINT TO PICK PATHS	`pmap_dc_point_to_pick_paths (...)`
MAP DEVICE COORDINATES TO WINDOW SYSTEM COORDINATES	`pmap_dc_to_wsc (...)`
MAP DEVICE COORDINATES TO WORLD COORDINATES	`pmap_dc_to_wc (...)`
MAP WINDOW SYSTEM COORDINATES TO DEVICE COORDINATES	`pmap_wsc_to_dc (...)`
MARK MULTI-PASS COMPLETION	`pmark_multi_pass_compl (...)`
MARK MULTI-PASS START	`pmark_multi_pass_start (...)`
MARK PASS COMPLETION	`pmark_pass_compl (...)`
MARK PASS START	`pmark_pass_start (...)`
MOVE ELEMENT FROM STRUCTURE	`pmove_elem_struct (...)`
MOVE ELEMENT RANGE FROM STRUCTURE	`pmove_elem_range_struct (...)`
MOVE ELEMENTS BETWEEN LABELS FROM STRUCTURE	`pmove_elems_between_labels_struct (...)`
OPEN DIRECT INTERPRETATION STRUCTURE	`popen_di_struct (...)`
POP STATE	`ppop_st (...)`
POST DIRECT INTERPRETATION STRUCTURE	`ppost_di_struct (...)`
POST DIRECT INTERPRETATION STRUCTURE TO POSTING GROUP	`ppost_di_struct_to_grp (...)`
POST STRUCTURE TO GROUP	`ppost_struct_to_grp (...)`
PUSH STATE	`ppush_st (...)`

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
REDRAW ALL STRUCTURES FROM POSTING GROUP	predraw_all_structs_from_grp (...)
REDRAW ALL STRUCTURES FROM POSTING GROUP ON TARGET	predraw_all_from_grp_on_targ (...)
REDRAW ALL STRUCTURES ON TARGET	predraw_all_structs_on_targ (...)
RENEW DIRECT INTERPRETATION STATE	prenew_di_state (...)
REQUEST COMPOSITE	preq_composite (...)
REQUEST COMPOSITE 3	preq_composite3 (...)
REQUEST SET	preq_set (...)
REQUEST SET 3	preq_set3 (...)
RESET ALL TRAVERSAL RESOURCES	preset_all_trav_res (...)
RETRIEVE NUMBER OF PASSES REQUIRED	pret_num_passes_req (...)
RETRIEVE WINDOW SYSTEM COLOUR	pret_window_system_colr (...)
SAMPLE COMPOSITE	psample_composite (...)
SAMPLE COMPOSITE 3	psample_composite3 (...)
SAMPLE SET	psample_set (...)
SAMPLE SET 3	psample_set3 (...)
SET ACTIVE TEXTURES	pset_active_textures (...)
SET ALPHA DATA SELECTION INDEX	pset_alpha_data_sel_ind (...)
SET ALPHA SOURCE SELECTOR	pset_alpha_src_sel (...)
SET APPLICATION FILTER	pset_appl_filter (...)
SET APPLICATION INTEGER	pset_appl_int (...)
SET APPLICATION REAL	pset_appl_real (...)
SET BACK ACTIVE TEXTURES	pset_back_active_textures (...)
SET BACK CLIPPING INDICATOR	pset_back_clip_indicator (...)
SET BACK PLANE DISTANCE	pset_back_plane_dist (...)
SET BACK TRANSPARENCY	pset_back_transparency (...)
SET COMPOSITE MODE	pset_composite_mode (...)
SET COMPOSITE PICK FILTER	pset_composite_pick_filter (...)
SET CONDITION FLAGS	pset_cond_flags (...)
SET CONDITION FLAGS FROM TESTS	pset_cond_flags_from_tests (...)
SET DEPTH CUE REPRESENTATION MASK	pset_depth_cue_rep_mask (...)
SET DEVICE COORDINATE CLIP REGIONS	pset_dc_clip_regions (...)
SET DEVICE COORDINATE CLIP REGIONS 3	pset_dc_clip_regions3 (...)
SET DIRECT INTERPRETATION MODE	pset_di_mode (...)
SET DIRECT INTERPRETATION PICK CORRELATION POINT	pset_di_pick_corr_point (...)
SET DIRECT INTERPRETATION PICK CORRELATION POINT 3	pset_di_pick_corr_point3 (...)
SET DIRECT INTERPRETATION PICK FILTER	pset_di_pick_filter (...)
SET DISPLAY TARGET	pset_disp_targ (...)
SET EDGECAP	pset_edgecap (...)
SET EDGEJOIN	pset_edgejoin (...)
SET EDGEMITRE LIMIT	pset_edgemitre_limit (...)
SET EDGE REPRESENTATION FULL	pset_edge_rep_full (...)
SET EDGE REPRESENTATION MASK	pset_edge_rep_mask (...)
SET EDGETYPE ADAPTABILITY	pset_edgetype_adapt (...)
SET EDGETYPE CONTINUITY	pset_edgetype_cont (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
SET EDGETYPE OFFSET	pset_edgetype_offset (...)
SET EXTENDED PATTERN REPRESENTATION	pset_ext_pat_rep (...)
SET FRONT CLIPPING INDICATOR	pset_front_clip_indicator (...)
SET FRONT PLANE DISTANCE	pset_front_plane_dist (...)
SET HIGHLIGHTING INDEX	pset_highl_ind (...)
SET HIGHLIGHTING METHOD	pset_highl_method (...)
SET HIGHLIGHTING REPRESENTATION	pset_highl_rep (...)
SET INTERIOR REPRESENTATION MASK	pset_int_rep_mask (...)
SET LINECAP	pset_linecap (...)
SET LINEJOIN	pset_linejoin (...)
SET LINEMITRE LIMIT	pset_linemitre_limit (...)
SET LINETYPE ADAPTABILITY	pset_linetype_adapt (...)
SET LINETYPE CONTINUITY	pset_linetype_cont (...)
SET LINETYPE OFFSET	pset_linetype_offset (...)
SET PATTERN REPRESENTATION MASK	pset_pat_rep_mask (...)
SET PICK MAPPING DATA	pset_pick_mapping_data (...)
SET POLYLINE REPRESENTATION FULL	pset_line_rep_full (...)
SET POLYLINE REPRESENTATION MASK	pset_line_rep_mask (...)
SET POLYMARKER REPRESENTATION MASK	pset_marker_rep_mask (...)
SET POSTING GROUP BACKGROUND METHOD	pset_post_grp_backg_method (...)
SET POSTING GROUP BACKGROUND STYLE	pset_post_grp_backg_style (...)
SET POSTING GROUP BORDER INDEX	pset_post_grp_border_ind (...)
SET POSTING GROUP BORDER INDICATOR	pset_post_grp_border_indicator (...)
SET POSTING GROUP PRIORITY	pset_post_grp_priority (...)
SET POSTING GROUP STATUS	pset_post_grp_status (...)
SET PROJECTION REFERENCE POINT	pset_proj_ref_point (...)
SET PROJECTION TYPE	pset_proj_type (...)
SET PROJECTION VIEWPORT	pset_proj_vp (...)
SET PROJECTION VIEWPORT 3	pset_proj_vp3 (...)
SET REFLECTANCE REPRESENTATION MASK	pset_refl_rep_mask (...)
SET RENDERING TARGET	pset_rend_targ (...)
SET SET MODE	pset_set_mode (...)
SET SET PICK FILTER	pset_set_pick_filter (...)
SET STATE OF VISUAL REPRESENTATION	pset_st_visual_rep (...)
SET TARGET DISPOSITION	pset_targ_dispos (...)
SET TARGET MANIPULATION MODE	pset_targ_manip_mode (...)
SET TARGET STATE OF VISUAL REPRESENTATION	pset_targ_st_visual_rep (...)
SET TEXT REPRESENTATION MASK	pset_text_rep_mask (...)
SET TEXTURE BINDING	pset_texture_binding (...)
SET TEXTURE COMPOSITION	pset_texture_composition (...)
SET TEXTURE PARAMETRIZATION	pset_texture_param (...)
SET TEXTURE PERSPECTIVE CORRECTION	pset_texture_perspect_corr (...)
SET TEXTURE REPRESENTATION	pset_texture_rep (...)
SET TEXTURE REPRESENTATION MASK	pset_texture_rep_mask (...)
SET TEXTURE RESOURCE OPTIMIZATION HEURISTICS	pset_texture_res_opt_heur (...)
SET TEXTURE SAMPLING	pset_texture_sampling (...)

Table 3 - Function names ordered by PHIGS and PHIGS PLUS function name (Continued)

PHIGS and PHIGS PLUS Name	C Name
SET TEXTURE SAMPLING FREQUENCY	`pset_texture_sampling_freq (...)`
SET TRANSPARENCY	`pset_transparency (...)`
SET TRANSPARENCY MODE	`pset_transparency_mode (...)`
SET TRANSPARENCY THRESHOLDS	`pset_transparency_thresholds (...)`
SET VIEW PLANE DISTANCE	`pset_view_plane_dist (...)`
SET VIEW PLANE NORMAL	`pset_view_plane_norm (...)`
SET VIEW REFERENCE POINT	`pset_view_ref_point (...)`
SET VIEW REFERENCE POINT 3	`pset_view_ref_point3 (...)`
SET VIEW UP VECTOR	`pset_view_up_vec (...)`
SET VIEW UP VECTOR 3	`pset_view_up_vec3 (...)`
SET VIEW WINDOW LIMITS	`pset_view_win_limits (...)`
SET WATCH ON ELEMENT RANGE	`pset_watch_on_elem_range (...)`
SET X-Y CLIPPING INDICATOR	`pset_xy_clip_indicator (...)`
TRANSFER DIRECT INTERPRETATION PICK SET	`ptrans_di_pick_set (...)`
UNDEFINE CHOICE	`pundefine_choice (...)`
UNDEFINE COMPOSITE	`pundefine_composite (...)`
UNDEFINE LOCATOR	`pundefine_locator (...)`
UNDEFINE PICK	`pundefine_pick (...)`
UNDEFINE POSTING GROUP	`pundefine_post_grp (...)`
UNDEFINE SET	`pundefine_set (...)`
UNDEFINE STRING	`pundefine_string (...)`
UNDEFINE STROKE	`pundefine_stroke (...)`
UNDEFINE VALUATOR	`pundefine_valuator (...)`
UNPOST ALL STRUCTURES FROM GROUP	`punpost_all_structs_from_grp (...)`
UNPOST DIRECT INTERPRETATION STRUCTURE	`punpost_di_struct (...)`
UNPOST STRUCTURES FROM GROUPS	`punpost_structs_from_grps (...)`
UPDATE TARGET	`pupd_targ (...)`
UPDATE VIEW REPRESENTATION	`pupd_view_rep (...)`
WORKSTATION TYPE CREATE	`pws_type_create (...)`
WORKSTATION TYPE DESTROY	`pws_type_destroy (...)`
WORKSTATION TYPE GET	`pws_type_get (...)`
WORKSTATION TYPE SET	`pws_type_set (...)`

”

Page 27

5 C PHIGS type definitions

Change the clause heading to “**C *Basic PHIGS profile* type definitions**”.

5.1 Mapping of PHIGS data types

Change the subclause heading to “**Mapping of *Basic PHIGS profile* data types**”.

The following text replaces the text of the first paragraph:

"The *Basic PHIGS profile* specifies a set of abstract data types. Table 4 gives the mapping from those data types defined in the *Basic PHIGS profile*."

Change the table title to "**Table - 4 Mapping of *Basic PHIGS profile* data types to C**"**.**

*Change the left table heading to "**Basic PHIGS profile*** **data type**"*.*

Page 27

5.2 Environmental type definitions

Append the phrase "of all profiles" *to the second sentence of the first paragraph.*

Page 28

5.3 Implementation dependent type definitions

Change the subclause heading to "**Implementation dependent *Basic PHIGS profile* type definitions**"*.*

The following text replaces the phrases of the "xxx" description concerned with the escape data records.

> ', "escape_in" for ESCAPE input data records, "escape_out" for ESCAPE output data records, "targ_op" for target operation types, "trav_res_op" for traversal resource operations, "in_type" for new classes of input measures'

Pages 29 and 30

The following type definitions replace the Pchoice_data *and* Pchoice_data3 *type definitions:*

"

`Pchoice_data` **CHOICE DATA RECORD**

```
 typedef struct {
    union Pchoice_pets {
       struct Pchoice_pet_other {
          Pint      unused;
       } pet_other; /* When no echo-specific data is required */
       struct Pchoice_pet_r1{
          . . .     /* impl. dependent */
       } pet_r1;  /* For PET 1 */
       struct Pchoice_pet_r2 {
          Pint         num_prompts;     /* number of prompts*/
          Ppr_switch   *prompts;        /* array of prompts*/
          . . .                         /* impl. defined*/
       } pet_r2;       /* For PET 2 */
```

```
        struct Pchoice_pet_r3 {
           Pint      num_strings;        /* number of choice strings*/
           char      **strings;          /* array of choice strings*/
           . . .                         /* impl. defined*/
        } pet_r3;  /* For PET 3 */
        struct Pchoice_pet_r4 {
           Pint      num_strings;        /* number of alternatives*/
           char      **strings;          /* array of strings*/
           . . .                         /* impl. defined*/
        } pet_r4;  /* For PET 4 */
        struct Pchoice_pet_r5 {
           Pint      struct_id;          /* structure identifier*/
           Pint      num_pick_ids;       /* number of alternatives*/
           Pint      *pick_ids;          /* array of pick identifiers*/
           . . .                         /* impl. defined*/
        } pet_r5;  /* For PET 5 */
        . . .        /* data for impl. defined pets */
     } pets;
     union Pchoice_measure_data {
        struct Pchoice_meas_other {
           Pint      unused;
        } meas_other; /* When no measure-specific data is required */
        . . .        /* data for impl. defined measure processes */
     } measure_data;
     union Pchoice_trigger_data {
        struct Pchoice_trig_other {
           Pint      unused;
        } trig_other; /* When no trigger-specific data is required */
        . . .        /* data for impl. defined trigger processes */
     } trigger_data;
     union Pchoice_ack_data {
        struct Pchoice_ack_other {
           Pint      unused;
        } ack_other;  /* When no ack.-specific data is required */
        . . .        /* data for impl. defined ack. processes */
     } acknowledgement_data;
  } Pchoice_data;
```

`Pchoice_data3` **CHOICE DATA RECORD 3**

```
 typedef struct {
    union Pchoice3_pets {
       struct Pchoice3_pet_other {
          Pint      unused;
       } pet_other; /* When no echo-specific data is required */
       struct Pchoice3_pet_r1{
          . . .                      /* impl. dependent*/
       } pet_r1;  /* For PET 1 */
       struct Pchoice3_pet_r2 {
          Pint      num_prompts;     /* number of prompts*/
          Ppr_switch*prompts;        /* array of prompts*/
          . . .                      /* impl. defined*/
       } pet_r2;      /* For PET 2 */
       struct Pchoice3_pet_r3 {
          Pint      num_strings;     /* number of choice strings*/
          char      **strings;       /* array of choice strings*/
          . . .                      /* impl. defined*/
       } pet_r3;  /* For PET 3 */
       struct Pchoice3_pet_r4 {
          Pint      num_strings;     /* number of alternatives*/
          char      **strings;       /* array of strings*/
          . . .                      /* impl. defined*/
       } pet_r4;  /* For PET 4 */
       struct Pchoice3_pet_r5 {
          Pint      struct_id;       /* structure identifier*/
          Pint      num_pick_ids;    /* number of alternatives*/
          Pint      *pick_ids;       /* array of pick identifiers*/
          . . .                      /* impl. defined*/
       } pet_r5;  /* For PET 5 */
       . . .        /* data for impl. defined pets */
    } pets;
    union Pchoice3_measure_data {
       struct Pchoice_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       . . .        /* data for impl. defined measure processes */
    } measure_data;
    union Pchoice3_trigger_data {
       struct Pchoice3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       . . .        /* data for impl. defined trigger processes */
    } trigger_data;
    union Pchoice3_ack_data {
       struct Pchoice3_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       . . .        /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pchoice_data3;”
```

Pages 33 and 34

The following type definition replaces the definitions of Ploc_data and Ploc_data3 respectively:

“

Ploc_data **LOCATOR DATA RECORD**

```
 typedef struct {
    union Ploc_pets {
       struct Ploc_pet_other {
          Pint      unused;
       } pet_other;   /* When no echo-specific data is required */
       struct Ploc_pet_r1 {
          . . .                        /* impl. dependent*/
       } pet_r1;      /* Supports standard PET 1 */
       struct Ploc_pet_r2 {
          . . .                        /* impl. dependent*/
       } pet_r2;      /* Supports standard PET 2 */
       struct Ploc_pet_r3 {
          . . .                        /* impl. dependent*/
       } pet_r3;      /* Supports standard PET 3 */
       struct Ploc_pet_r4 {
          Pline_attrs line_attrs;      /* polyline attributes*/
          . . .                        /* impl. dependent*/
       } pet_r4;      /* Supports standard PET 4 */
```

```
            struct Ploc_pet_r5 {
               Pline_fill_ctrl_flag  line_fill_ctrl_flag;   /* ctrl. flag*/
               union Ploc_attrs {
                  Pline_attrs   line_attrs; /* polyline attributes*/
                  Pint_attrs    int_attrs;  /* interior attributes*/
                  struct Ploc_fill_set {
                     Pint_attrs   int_attrs;   /* interior attributes*/
                     Pedge_attrs edge_attrs; /* edge attributes*/
                  } fill_set;
               } attrs;
               . . .       /* data for impl. defined PET's */
            } pet_r5;      /* Supports standard PET 5 */
            struct Ploc_pet_r6 {
               . . .                       /* impl. dependent*/
            } pet_r6;      /* Supports standard PET 6 */
            . . .          /* data for impl. defined PET's */
         } pets;
         union Ploc_measure_data {
            struct Ploc_meas_other {
               Pint      unused;
            } meas_other; /* When no measure-specific data is required */
            . . .       /* data for impl. defined measure processes */
         } measure_data;
         union Ploc_trigger_data{
            struct Ploc_trig_other {
               Pint      unused;
            } trig_other; /* When no trigger-specific data is required */
            . . .       /* data for impl. defined trigger processes */
         } trigger_data;
         union Ploc_ack_data {
            struct Ploc_ack_other {
               Pint      unused;
            } ack_other;  /* When no ack.-specific data is required */
            . . .       /* data for impl. defined ack. processes */
         } acknowledgement_data;
      } Ploc_data;
```

Ploc_data3 **LOCATOR DATA RECORD 3**

```
 typedef struct {
    union Ploc3_pets {
       struct Ploc3_pet_other {
          Pint      unused;
       } pet_other;   /* When no echo-specific data is required */
       struct Ploc3_pet_r1 {
          . . .                       /* impl. dependent*/
       } pet_r1;      /* Supports standard PET 1 */
       struct Ploc3_pet_r2 {
          . . .                       /* impl. dependent*/
       } pet_r2;      /* Supports standard PET 2 */
       struct Ploc3_pet_r3 {
          . . .                       /* impl. dependent*/
       } pet_r3;      /* Supports standard PET 3 */
       struct Ploc3_pet_r4 {
          Pline_attrs line_attrs;     /* polyline attributes*/
            . . .                     /* impl. dependent*/
       } pet_r4;      /* Supports standard PET 4 */
       struct Ploc3_pet_r5 {
          Pline_fill_ctrl_flag line_fill_ctrl_flag;  /* ctrl. flag*/
          union Ploc3_attrs {
             Pline_attrs   line_attrs; /* polyline attributes*/
             Pint_attrs    int_attrs;  /* interior attributes*/
             struct Ploc3_fill_set {
                Pint_attrs   int_attrs;   /* interior attributes*/
                Pedge_attrs edge_attrs; /* edge attributes*/
             } fill_set;
          } attrs;
          . . .       /* data for impl. defined PET's */
       } pet_r5;      /* Supports standard PET 5 */
       struct Ploc3_pet_r6 {
          . . .                       /* impl. dependent*/
       } pet_r6;      /* Supports standard PET 6 */
       . . .       /* data for impl. defined PET's */
    } pets;
    union Ploc3_measure_data {
       struct Ploc3_meas_other {
          Pint      unused;
       } meas_other;  /* When no measure-specific data is required */
```

```
        . . .        /* data for impl. defined measure processes */
     } measure_data;
     union Ploc3_trigger_data {
        struct Ploc3_trig_other {
           Pint      unused;
        } trig_other;  /* When no trigger-specific data is required */
        . . .        /* data for impl. defined trigger processes */
     } trigger_data;
     union Ploc3_ack_data {
        struct Ploc3_ack_other {
           Pint      unused;
        } ack_other;   /* When no ack.-specific data is required */
        . . .        /* data for impl. defined ack. processes */
     } acknowledgement_data;
  } Ploc_data3;"
```

Pages 35 through 38

The following type definitions replace the corresponding definitions respectively:

"

`Ppick_data` **PICK DATA RECORD**

```
  typedef struct {
     union Ppick_pets {
        struct Ppick_pet_other {
           Pint      unused;
        } pet_other;  /* When no echo-specific data is required */
        struct Ppick_pet_r1 {
           . . .                       /* impl. dependent*/
        } pet_r1;  /* For PET 1 */
        . . .        /* data for impl. defined pets */
        struct Ppick_pet_r2 {
           ...                         /* implementation dependent*/
        } pet_r2;  /* For PET 2 */
        struct Ppick_pet_r3 {
           ...                         /* implementation dependent*/
        } pet_r3;  /* For PET 3 */
        struct Ppick_pet_r4 {
           Pint      highl_ind;         /* highlighting index*/
           . . .                        /* impl. dependent*/
        } pet_r4;  /* For PET 4 */
        struct Ppick_pet_r5 {
           Pint      highl_ind;         /* highlighting index*/
           . . .                        /* impl. dependent*/
        } pet_r5;  /* For PET 5 */
        struct Ppick_pet_r6 {
           Pint      highl_ind;         /* highlighting index*/
           . . .                        /* impl. dependent*/
        } pet_r6;  /* For PET 6 */
        ...                             /* implementation defined PET's*/
     } pets;
     union Ppick_measure_data {
        struct Ppick_meas_other {
           Pint      unused;
        } meas_other; /* When no measure-specific data is required */
        . . .       /* data for impl. defined measure processes */
     } measure_data;
     union Ppick_trigger_data {
        struct Ppick_trig_other {
           Pint      unused;
        } trig_other; /* When no trigger-specific data is required */
        . . .       /* data for impl. defined pick processes */
     } trigger_data;
     union Ppick_ack_data {
        struct Ppick_ack_other {
           Pint      unused;
        } ack_other; /* When no ack.-specific data is required */
        . . .       /* data for impl. defined ack. processes */
     } acknowledgement_data;
  } Ppick_data;
```

Ppick_data3 **PICK DATA RECORD 3**

```
typedef struct {
   union Ppick3_pets {
      struct Ppick3_pet_other {
         Pint       unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Ppick3_pet_r1 {
         . . .                       /* impl. dependent*/
      } pet_r1;  /* For PET 1 */
      struct Ppick3_pet_r2 {
         ...                         /* implementation dependent*/
      } pet_r2;  /* For PET 2 */
      struct Ppick3_pet_r3 {
         ...                         /* implementation dependent*/
      } pet_r3;  /* For PET 3 */
      struct Ppick3_pet_r4 {
         Pint       highl_ind;       /* highlighting index*/
         . . .                       /* impl. dependent*/
      } pet_r4;  /* For PET 4 */
      struct Ppick3_pet_r5 {
         Pint       highl_ind;       /* highlighting index*/
         . . .                       /* impl. dependent*/
      } pet_r5;  /* For PET 5 */
      struct Ppick3_pet_r6 {
         Pint       highl_ind;       /* highlighting index*/
         . . .                       /* impl. dependent*/
      } pet_r6;
      . . .      /* data for impl. defined pets */
   } pets;  /* For PET 6 */
   union Ppick3_measure_data {
      struct Ppick3_meas_other {
         Pint       unused;
      } meas_other; /* When no measure-specific data is required */
      . . .      /* data for impl. defined measure processes */
   } measure_data;
   union Ppick3_trigger_data {
      struct Ppick3_trig_other {
         Pint       unused;
      } trig_other; /* When no trigger-specific data is required */
      . . .        /* data for impl. defined pick processes */
    } trigger_data;
    union Ppick3_ack_data {
      struct Ppick3_ack_other {
        Pint      unused;
      } ack_other; /* When no ack.-specific data is required */
      . . .        /* data for impl. defined ack. processes */
    } acknowledgement_data;
  } Ppick_data3;
```

Pstring_data **STRING DATA RECORD**

```
   typedef struct {
    Pint                 in_buf_size;   /* input buffer size*/
    Pint                 init_pos;      /* initial editing position*/
    union Pstring_pets {
      struct Pstring_pet_other {
        Pint      unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Pstring_pet_r1{
        . . .                           /* impl. dependent*/
      } pet_r1;  /* For PET 1 */
      . . .        /* data for impl. defined pets */
    } pets;
    union Pstring_measure_data {
      struct Pstring_meas_other {
        Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      . . .        /* data for impl. defined measure processes */
    } measure_data;
    union Pstring_trigger_data {
      struct Pstring_trig_other {
        Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      . . .        /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstring_ack_data {
       struct Pstring_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       . . .        /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstring_data;
```

Pstring_data3 **STRING DATA RECORD 3**

```
 typedef struct {
    Pint                 in_buf_size;   /* input buffer size*/
    Pint                 init_pos;      /* initial editing position*/
    union Pstring3_pets {
       struct Pstring_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstring3_pet_r1{
          . . .                         /* impl. dependent*/
       } pet_r1;  /* For PET 1 */
       . . .        /* data for impl. defined pets */
    } pets;
    union Pstring3_measure_data {
       struct Pstring3_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       . . .        /* data for impl. defined measure processes */
    } measure_data;
    union Pstring3_trigger_data {
       struct Pstring3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       . . .        /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstring3_ack_data {
       struct Pstring3_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
```

```
        . . .       /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstring_data3;
```

Pstroke_data **STROKE DATA RECORD**

```
 typedef struct {
    Pint                buffer_size;   /* input buffer size */
    Pint                init_pos;      /* initial editing position */
    Pfloat              x_interval;    /* x trigger interval */
    Pfloat              y_interval;    /* y trigger interval */
    Pfloat              time_interval; /* time trigger interval */
    union {
       struct Pstroke_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstroke_pet_r1 {
          . . .                        /* impl. dependent*/
       } pet_r1;  /* For std PET 1 */
       struct Pstroke_pet_r2 {
          . . .                        /* impl. dependent*/
       } pet_r2;  /* For std PET 2 */
       struct Pstroke_pet_r3 {
          Pmarker_attrs  marker_attrs; /* marker attributes*/
          . . .                        /* impl. dependent*/
       } pet_r3;  /* For std PET 3 */
       struct Pstroke_pet_r4 {
          Pline_attrsline_attrs;       /* line attributes*/
          . . .                        /* impl. dependent*/
       } pet_r4;  /* For std PET 4 */
       . . .       /* data for impl. defined PET's */
          } pets;
    union Pstroke_measure_data {
       struct Pstroke_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       . . .       /* data for impl. defined measure processes */
    } measure_data;
    union Pstroke_trigger_data {
       struct Pstroke_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       . . .        /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstroke_ack_data {
       struct Pstroke_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       . . .        /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstroke_data;
```

Pstroke_data3 **STROKE DATA RECORD 3**

```
 typedef struct {
    Pint                 in_buf_size;    /* input buffer size*/
    Pint                 init_pos;       /* initial editing position*/
    Pfloat               x_interval;     /* x trigger interval*/
    Pfloat               y_interval;     /* y trigger interval*/
    Pfloat               z_interval;     /* z trigger interval*/
    Pfloat               time_interval;  /* time trigger interval*/
    union Pstroke3_pets {
       struct Pstroke3_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstroke3_pet_r1 {
          . . .                          /* impl. dependent*/
       } pet_r1;  /* For std PET 1 */
       struct Pstroke3_pet_r2 {
          . . .     /* impl. dependent */
       } pet_r2;  /* For std PET 2 */
       struct Pstroke3_pet_r3 {
          Pmarker_attrs  marker_attrs; /* marker attributes*/
          . . .                          /* impl. dependent*/
       } pet_r3;  /* For std PET 3 */
```

```
      struct Pstroke3_pet_r4 {
         Pline_attrs   line_attrs;    /* line attributes*/
         . . .                        /* impl. dependent*/
      } pet_r4;  /* For std PET 4 */
      . . .       /* data for impl. defined PET's */
   } pets;

   union Pstroke3_measure_data {
      struct Pstroke3_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      . . .       /* data for impl. defined measure processes */
   } measure_data;

   union Pstroke3_trigger_data {
      struct Pstroke3_trig_other {
         Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      . . .       /* data for impl. defined trigger processes */
   } trigger_data;

   union Pstroke3_ack_data {
      struct Pstroke3_ack_other {
         Pint      unused;
      } ack_other;  /* When no ack.-specific data is required */
      . . .       /* data for impl. defined ack. processes */
   } acknowledgement_data;

} Pstroke_data3;
```

Pval_data **VALUATOR DATA RECORD**

```
 typedef struct {

    Pfloat             low;           /* Low value of valuator range*/

    Pfloat             high;          /* High value of valuator range*/

    union Pval_pets {
       struct Pval_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       . . .       /* data for impl. defined pets */
    } pets;
    union Pval_measure_data {
       struct Pval_measure_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       . . .        /* data for impl. defined measure processes */
    } measure_data;
    union Pval_trigger_data {
       struct Pval_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       . . .        /* data for impl. defined trigger processes */
    } trigger_data;
    union Pval_ack_data {
       struct Pval_ack_other {
          Pint      unused;
       } ack_other; /* When no ack.-specific data is required */
       . . .        /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pval_data;
```

Pval_data3 **VALUATOR DATA RECORD 3**

```
 typedef struct {
    Pfloat            low;            /* Low value of valuator range*/
    Pfloat            high;           /* High value of valuator range*/
    union Pval3_pets {
       struct Pval3_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       . . .        /* data for impl. defined pets */
    } pets;
    union Pval3_measure_data {
       struct Pval3_measure_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       . . .        /* data for impl. defined measure processes */
    } measure_data;
    union Pval3_trigger_data {
       struct Pval3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       . . .      /* data for impl. defined trigger processes */
    } trigger_data;
    union Pval3_ack_data {
       struct Pval3_ack_other {
          Pint      unused;
       } ack_other; /* When no ack.-specific data is required */
       . . .      /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pval_data3;”
```

Page 38

5.4 Implementation independent type definitions

*Change the subclause heading to “***Implementation independent *Basic PHIGS profile* type definitions***”.*

Page 65

The following text replaces item PSTRUCT_ST_STOP *in the definition of Pstruct_st:*

```
   “PSTRUCT_ST_STOP,
    PSTRUCT_ST_DISO”
```

Page 73

6 C PHIGS macro definitions

*Change the clause heading to “***C *Basic PHIGS profile* macro definitions***”.*

Page 83

6.3 Linetypes

6.3.1 Linetypes

The following entry is appended:

```
“
#define PLINE_DASH_DOT_DOT                    (5)”
```

Page 76

6.4 Error codes

The following definition replaces error 5:

```
“
#define PE_NOT_STOP           (5)   /* Ignoring function, function requires
                                       state (PHOP,*,STOP | DISO,*) */”
```

Page 85

7 C PHIGS functions

Change the clause heading to “**C *Basic PHIGS profile* functions**”.

Page 17 (ISO/IEC 9593-4:1992/Amd. 1:1994)

8 C PHIGS PLUS type definitions

Change the subclause heading to “**C *PHIGS PLUS profile* type definitions**”.

8.1 Mapping of PHIGS PLUS data types

Change the clause heading to “**Mapping of *PHIGS PLUS profile* data types**”.

The following text replaces the text of the first paragraph:

“The *PHIGS PLUS profile* specifies a set of abstract data types beyond the types defined in the *Basic PHIGS profile*. Table 4 gives the mapping from those additional data types defined in the *PHIGS PLUS profile*.”

Change the table title to “**Table - 4 Mapping of *PHIGS PLUS profile* data types to C**”.

Change the left table heading to “***PHIGS PLUS profile* data type**”.

Page 18 (ISO/IEC 9593-4:1992/Amd. 1:1994)

8.2 Modifications to PHIGS data types

Change the subclause heading to “**Modifications to *Basic PHIGS profile* data types**”.

Page 24 (ISO/IEC 9593-4:1992/Amd. 1:1994)

8.3 Implementation dependent PHGIS PLUS type definitions

Change the subclause heading to “**Implementation dependent *PHIGS PLUS profile* type definitions**”.

Page 34 (ISO/IEC 9593-4:1992/Amd. 1:1994)

8.4 Implementation independent PHIGS PLUS type definitions

*Change the subclause heading to "***Implementation independent** ***PHIGS PLUS profile*** **type definitions***".*

Page 56 (ISO/IEC 9593-4:1992/Amd. 1:1994)

9 C PHIGS PLUS macro definitions

*Change the clause heading to "***C*** ***PHIGS PLUS profile*** **macro definitions***".*

Page 63 (ISO/IEC 9593-4:1992/Amd. 1:1994)

10 C PHIGS PLUS functions

*Change the clause heading to "***C*** ***PHIGS PLUS profile*** **functions***".*

Page 95 (ISO/IEC 9593-4:1992/Amd. 2:1994)

The following new clauses are inserted following clause 10.

"

11 C *Full PHIGS profile* type definitions

11.1 Mapping of *Full PHIGS profile* data types

The *Full PHIGS profile* specifies a set of abstract data types beyond the types defined in the *PHIGS PLUS profile*. Table 4 gives the mapping from those additional data types defined in the *Full PHIGS profile*.

Table 4 - Mapping of *Full PHIGS profile* data types to C

PHIGS data type		C binding data type
@t	location of t	*t
AP	acknowledgement process	Pack_process
BIT	bit	unsigned int:1
CF	configuration setting list	Pconfig_settings
CFT	condition flag test	Pcond_test
EP	echo process	Pecho_process
M	measure	Pmeasure
MD	marker descriptor	Pmarker_desc
MP	measure process	Pmeas_process
MS	measure set	Pmeasure_set
TA	target address	Ptarg_addr
TO	target operation	Ptarg_op
TP	trigger process	Ptrig_process
TR	traversal resource	Ptrav_res

11.2 Modifications to *Basic PHIGS profile* type definitions

Paspect **ASPECT**

```
typedef enum {
    /* start of PHIGS enumerations */
    ...,
    /* end of PHIGS enumerations */
    /* start of PHIGS PLUS enumerations */
    ...,
    /* end of PHIGS PLUS enumerations */
    /* start of PHIGS FULL enumerations */
    PASPECT_HIGHL_METHOD
    /* end of PHIGS FULL enumerations */
} Paspect;
```

Pcolr_rep **COLOUR REPRESENTATION**

```
typedef union {
    /* start of Basic PHIGS union members */
    Prgb      rgb;     /* RGB colour specification                       */
    Pcieluv   cieluv;  /* CIE L*u*v* colour specification                */
    Phls      hls;     /* HLS colour specification                       */
    Phsv      hsv;     /* HSV colour specification                       */
    /* end of Basic PHIGS union members */
    /* start of Full PHIGS union members */
    Prgba     rgba;    /* RGB specification with alpha                   */
    Pcieluva  cieluva;/* CIE L*u*v* colour specfication with alpha       */
    Phlsa     hlsa;    /* HLS colour specificaiton with alpha            */
    Phsva     hsva;    /* HSV colour specification with alpha            */
    /* end of Full PHIGS union members */
    ...                /* implementation defined */
} Pcolr_rep;
```

Pelem_data **ELEMENT DATA**

```
typedef union {
  /* start of Basic PHIGS element data */
  ...
  /* end of Basic PHIGS element data */
  /* start of PHIGS PLUS element data */
  ...
  /* end of PHIGS PLUS element data */
  /* start of Full PHIGS element data */
  struct Pappl_int {
   Pint   integer_id;    /* application integer identifier         */
   Pint   value;         /* value for application integer          */
  } appl_int;
  struct Pappl_real {
   Pint      real_id;       /* application real identifier         */
   Pfloat    value;         /* value for application real          */
  } appl_real;
  struct Pcircle3 {
      Ppoint3     center_point;      /* center point                */
      Pfloat      radius;            /* radius                      */
      Pvec3       ref_vecs[2];       /* reference vectors           */
  } circle3;
  struct Pcircle {
   Ppoint      center_point;         /* center point                */
   Pfloat      radius;               /* radius                      */
  } circle;
  struct Pcircular_arc3 {
   Ppoint3             center_point;  /* center point               */
   Pfloat              radius;        /* radius                     */
   Pvec3               ref_vecs[2];   /* reference vectors          */
   Pfloat              start;         /* start angle                */
   Pfloat              end;           /* end angle                  */
  } circular_arc3;
  struct Pcircular_arc {
   Ppoint              center_point;  /* center point               */
   Pfloat              radius;        /* radius                     */
   Pfloat              start;         /* start angle                */
   Pfloat              end;           /* end angle                  */
  } circular_arc;
  struct Pellipse3 {
   Ppoint3        center_point;      /* center point                */
   Pvec3          major_ref_vec;     /* major axis reference vector */
   Pvec3          minor_ref_vec;     /* minor axis reference vector */
  } ellipse3;
  struct Pellipse {
   Ppoint         center_point;      /* center point                */
   Pvec           major_ref_vec;     /* major axis reference vector */
   Pvec           minor_ref_vec;     /* minor axis reference vector */
  } ellipse;
```

```
struct Pelliptical_arc3 {
  Ppoint3        center_point;      /* center point                */
  Pvec3          major_ref_vec;     /* major axis reference vector */
  Pvec3          minor_ref_vec;     /* minor axis reference vector */
  Pfloat         start;             /* start angle                 */
  Pfloat         end;               /* end angle                   */
} elliptical_arc3;
struct Pelliptical_arc {
  Ppoint         center_point;      /* center point                */
  Pvec           major_ref_vec;     /* major axis reference vector */
  Pvec           minor_ref_vec;     /* minor axis reference vector */
  Pfloat         start;             /* start angle                 */
  Pfloat         end;               /* end angle                   */
} elliptical_arc;
struct Pfill_circle3 {
  Ppoint3     center_point;         /* center point                */
  Pfloat      radius;               /* radius                      */
  Pvec3       ref_vecs[2];          /* reference vectors           */
} fill_circle3;
struct Pfill_circle {
  Ppoint      center_point;         /* center point                */
  Pfloat      radius;               /* radius                      */
} fill_circle;
struct Pcircular_arc_close3 {
  Ppoint3            center_point;  /* center point                */
  Pfloat             radius;        /* radius                      */
  Pvec3              ref_vecs[2];   /* reference vectors           */
  Pfloat             start;         /* start angle                 */
  Pfloat             end;           /* end angle                   */
  Pclosure           type;          /* closure type                */
} circular_arc_close3;
struct Pcircular_arc_close {
  Ppoint             center_point;  /* center point                */
  Pfloat             radius;        /* radius                      */
  Pfloat             start;         /* start angle                 */
  Pfloat             end;           /* end angle                   */
  Pclosure           type;          /* closure type                */
} circular_arc_close;
struct Pfill_ellipse3 {
  Ppoint3        center_point;      /* center point                */
  Pvec3          major_ref_vec;     /* major axis reference vector */
  Pvec3          minor_ref_vec;     /* minor axis reference vector */
} fill_ellipse3;
struct Pfill_ellipse {
  Ppoint         center_point;      /* center point                */
  Pvec           major_ref_vec;     /* major axis reference vector */
  Pvec           minor_ref_vec;     /* minor axis reference vector */
} fill_ellipse;
```

```
struct Pelliptical_arc_close3 {
  Ppoint3        center_point;      /* center point                */
  Pvec3          major_ref_vec;     /* major axis reference vector */
  Pvec3          minor_ref_vec;     /* minor axis reference vector */
  Pfloat         start;             /* start angle                 */
  Pfloat         end;               /* end angle                   */
  Pclosure       type;              /* closure type                */
} elliptical_arc_close3;
struct Pelliptical_arc_close {
  Ppoint         center_point;      /* center point                */
  Pvec           major_ref_vec;     /* major axis reference vector */
  Pvec           minor_ref_vec;     /* minor axis reference vector */
  Pfloat         start;             /* start angle                 */
  Pfloat         end;               /* end angle                   */
  Pclosure       type;              /* closure type                */
} elliptical_arc_close;
Plinetype_adapt linetype_adapt;     /* linetype adaptability       */
Plinetype_cont  linetype_cont;      /* linetype continuity         */
Phighl_method   highl_method;       /* highlighting method         */
Pint_list       active_textures;      /* active textures           */
Pint_list       back_active_textures; /* back active textures      */
Pperspect_corr  perspect_corr;    /* textr perspective correction  */
struct Ptexture_res_opt_heur {
    Pint          opt_hint;        /* optimization hint            */
    Pint_list     usage_priorities; /* texture usage priorities    */
} texture_res_opt_heur;
Pint_list       alpha_src_sel;    /* alpha source selector         */
struct Pcond_exec_struct {
  Pint           struct_id;         /* structure identifier        */
  Ptest          test;              /* condition test              */
} cond_exec_struct;
struct Pcond_inst_struct {
  Pint           struct_id;         /* structure identifier        */
  Ptest          test;              /* condition test              */
} cond_inst_struct;
Ptest           cond_return;        /* condition test              */
struct Pcond_skip_elems {
  Pint           skip_count;        /* number of elements to skip  */
  Ptest          test;              /* condition test              */
} cond_skip_elems;
```

```
  struct Pcond_skip_to_label {
   Pint           label;            /* label to which to skip      */
   Ptest          test;             /* condition test              */
  } cond_skip_to_label;
  /* end of Full PHIGS element data */

 } Pelem_data;
```

`Pelem_type` **ELEMENT TYPE**

```
 typedef enum {

    /* start of PHIGS enumerations */
    ...,
    /* end of PHIGS enumerations */
    /* start of PHIGS PLUS enumerations */
    ...,
    /* end of PHIGS PLUS enumerations */
    /* start of Full PHIGS enumerations */
    PELEM_APPL_INT,
    PELEM_APPL_REAL,
    PELEM_CIRCLE3,
    PELEM_CIRCLE,
    PELEM_CIRCULAR_ARC3,
    PELEM_CIRCULAR_ARC,
    PELEM_ELLIPSE3,
    PELEM_ELLIPSE,
    PELEM_ELLIPTICAL_ARC3,
    PELEM_ELLIPTICAL_ARC,
    PELEM_FILL_CIRCLE3,
    PELEM_FILL_CIRCLE,
    PELEM_CIRCULAR_ARC_CLOSE3,
    PELEM_CIRCULAR_ARC_CLOSE,
    PELEM_FILL_ELLIPSE3,
    PELEM_FILL_ELLIPSE,
    PELEM_ELLIPTICAL_ARC_CLOSE3,
    PELEM_ELLIPTICAL_ARC_CLOSE,
    PELEM_HIGHL_IND,
    PELEM_LINETYPE_ADAPT,
    PELEM_LINETYPE_CONT,
    PELEM_LINETYPE_OFFSET,
    PELEM_HIGHL_METHOD,
    PELEM_ACTIVE_TEXTURES,
    PELEM_BACK_ACTIVE_TEXTURES,
    PELEM_TEXTURE_PERSPECT_CORR,
    PELEM_TEXTURE_SAMPLING_FREQ,
    PELEM_TEXTURE_RES_OPT_HEUR,
    PELEM_TRANSPARENCY,
    PELEM_BACK_TRANSPARENCY,
    PELEM_ALPHA_SRC_SEL,
    PELEM_ALPHA_DATA_SEL_IND,
    PELEM_INST_STRUCT,
    PELEM_COND_EXEC_STRUCT,
    PELEM_COND_INST_STRUCT,
    PELEM_COND_RETURN,
    PELEM_COND_SKIP_ELEMS,
    PELEM_COND_SKIP_TO_LABEL,
    PELEM_PUSH_ST,
    PELEM_POP_ST
    /* end of Full PHIGS enumerations */

 } Pelem_type;
```

`Pin_class` **INPUT CLASS**

```
 typedef enum {

    /* start of Basic PHIGS enumerations */
    PIN_NONE,
    PIN_LOC,
    PIN_STROKE,
    PIN_VAL,
    PIN_CHOICE,
    PIN_PICK,
    PIN_STRING,
    /* end of Basic PHIGS enumerations */
    /* start of Full PHIGS enumerations */
    PIN_SET,
    PIN_COMPOSITE
    /* end of Full PHIGS enumerations */

 } Pin_class;
```

Pint_style **INTERIOR STYLE**

```
typedef enum {

    /* start of Basic PHIGS enumerations */
    ...
    /* end of Basic PHIGS enumerations */
    /* start of Full PHIGS enumerations */
    PSTYLE_TEXTURE
    /* end of Full PHIGS PLUS enumerations */

} Pint_style;
```

Popen_struct_status **OPEN STRUCTURE STATUS**

```
typedef enum {

    /* start of Basic PHIGS enumerations */
    PSTRUCT_NONE,
    PSTRUCT_OPEN,
    /* end of Basic PHIGS enumerations */
    /* start of Full PHIGS enumerations */
    PSTRUCT_DI_OPEN
    /* end of Full PHIGS enumerations */

} Popen_struct_status;
```

11.3 Modifications to *PHIGS PLUS profile* type definitions

`Pcolr_rep_ptr` **COLOUR REPRESENTATION POINTER**

```
 typedef union {
    /* start of PHIGS PLUS union members */
    Prgb        *rgb;          /* pointer to RGB colour values */
    Pcieluv     *cieluv;       /* pointer to CIELUV pointer values */
    Phls        *hls;          /* pointer to HLS pointer values */
    Phsv        *hsv;          /* pointer to HSV pointer values */
    /* end of PHIGS PLUS union members */
    /* start of Full PHIGS union members */
    Prgba       *rgba;         /* pointer to RGBA colour values */
    Pcieluva    *cieluva;      /* pointer to CIELUVA pointer values */
    Phlsa       *hlsa;         /* pointer to HLSA pointer values */
    Phsva       *hsva;         /* pointer to HSVA pointer values */
    /* end of Full PHIGS union members */
    ...                        /* implementation defined */
 } Pcolr_rep_ptr;
```

`Ppat_rep_plus` **PATTERN REPRESENTATION PLUS**

```
 typedef Pgcolr_array Ppat_rep_plus;
```

11.4 Implementation dependent *Full PHIGS profile* type definitions

---

`Pbackg_method` **POSTING GROUP BACKGROUND METHOD**

```
 typedef struct {
    Pint               method;        /* background method         */
    union Pbackg_method_data {
       struct Pcolr_table_0 {
          Pint      unused;           /* unused                    */
       } colr_table_0;
       struct Pcolr_ind {
          Pint      colr_ind;         /* colour index              */
       } colr_ind;
       struct Pcolr {
          Pcolr_rep colr_rep;         /* target address            */
       } colr;
```

```
        struct Pimage_resource {
           Pint      image_resource_id; /* image resource identifier   */
        } image_resource;
        ...                             /* implementation defined      */
     } data;
  } Pbackg_method;
```

`Pclamp_method` **BOUNDARY CLAMP METHOD DATA RECORD**

```
  typedef struct {
     Pint            method;          /* clamp method                  */
     union Pclamp_method_data {
        Pint         unused;          /* used by method 1              */
        Pgcolr       clamp_colr;      /* used by method 2              */
        ...                           /* implementation defined        */
     } data;
  } Pclamp_method;
```

`Pcoord_src` **COOORDINATE SOURCE DATA RECORD**

```
  typedef struct {
     Pint            coord_src;       /* coordinate source             */
     union Pcoord_src_data {          /* coordinate source data record */
        Pint         unused;          /* used by coord sources 1 - 5   */
        Pint         coord_ind;       /* used by coord source 6        */
        Pmatrix3     refl_matrix;     /* used by coord sources 7 and 8 */
        ...                           /* implementation defined        */
     } data;
  } Pcoord_src;
```

`Pdi_pick_data` **DIRECT INTERPRETATION PICK DATA RECORD**

```
  typedef struct {
     union Pdi_pick_pets {
        struct Pdi_pick_pet_other {
           Pint      unused;
        } pet_other;  /* When no echo-specific data is required */
        struct Pdi_pick_pet_r1 {
           . . .                          /* impl. dependent             */
        } pet_r1;  /* For PET 1 */
        ...                               /* implementation defined PET's*/
     } pets;
     union Pdi_pick_measure_data {
        struct Pdi_pick_meas_other {
           Pint      unused;
        } meas_other; /* When no measure-specific data is required */
        . . .        /* data for impl. defined measure processes */
     } measure_data;
     union Pdi_pick_trigger_data {
        struct Pdi_pick_trig_other {
           Pint      unused;
        } trig_other; /* When no trigger-specific data is required */
        . . .        /* data for impl. defined di_pick processes */
     } trigger_data;
     union Pdi_pick_ack_data {
        struct Pdi_pick_ack_other {
           Pint      unused;
        } ack_other; /* When no ack.-specific data is required */
        . . .        /* data for impl. defined ack. processes */
     } acknowledgement_data;
  } Pdi_pick_data;
```

Pdi_pick_data3 **DIRECT INTERPRETATION PICK DATA RECORD 3**

```
  typedef struct {
     union Pdi_pick3_pets {
        struct Pdi_pick3_pet_other {
           Pint      unused;
        } pet_other;  /* When no echo-specific data is required */
        struct Pdi_pick3_pet_r1 {
           . . .                          /* impl. dependent             */
        } pet_r1;  /* For PET 1 */
        . . .        /* data for impl. defined pets */
     } pets;
     union Pdi_pick3_measure_data {
        struct Pdi_pick3_meas_other {
           Pint      unused;
        } meas_other; /* When no measure-specific data is required */
```

```
      . . .       /* data for impl. defined measure processes */
    } measure_data;
    union Pdi_pick3_trigger_data {
      struct Pdi_pick3_trig_other {
        Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      . . .       /* data for impl. defined di_pick processes */
    } trigger_data;
    union Pdi_pick3_ack_data {
      struct Pdi_pick3_ack_other {
        Pint      unused;
      } ack_other; /* When no ack.-specific data is required */
      . . .       /* data for impl. defined ack. processes */
    } acknowledgement_data;
  } Pdi_pick_data3;
```

`Phighl_method` **HIGHLIGHTING METHOD**

```
  typedef struct {
    Pint                method;
    union _Phighl_data {
      Pint          unused;
      Pfloat        blink_rate;
      Pint          colr_ind;
      Pgcolr        colr;
      struct _Phighl_blink_colr {
        Pfloat    blink_rate;
        Pgcolr    blink_colr;
      } highl_blink_colr;
      ...         /* implementation defined methods */
    } highl_data;
  } Phighl_method;
```

Pimage_res **IMAGE RESOURCE**

```
typedef struct {
   Pint image_spec_method;
   union _Pimage_data {
      struct Puncomp_colr_ind_array {
         Pint       *colr_ind_array;    /* uncompressed array of      */
                                        /* colour indices             */
      } uncomp_colr_ind_array;
      struct Puncomp_colr_array {
         Pgcolr_array  *colr_array;     /* uncompressed array of      */
                                        /*  colours                   */
      } uncomp_colr_array;
      struct Pwindow_sys_bitmap {
         Pint          handle;          /* bitmap handle              */
      } window_sys_bitmap;
      struct Pvideo_sig_chan_id {
         Pint          chan_id;         /* video signal channel id    */
      } video_sig_chan_id;
      struct Pluminance {
         Pfloat_array  *lum_values;     /* uncompressed array of*/
                                        /* luminance values*/
      } luminance;
      struct Pluminance_alpha {
         Plum_alpha_array  *lum_values;    /* uncompressed array of   */
                                           /* luminance values        */
      } luminance_alpha;
      struct Pmipmap_colrs {
         Pgcolr_array_set  colr_arrays;    /* set of uncompressed     */
                                           /* colour arrays           */
      } mipmap_colrs;
      struct Pmipmap_luminance {
         Pfloat_array_set  lum_values;     /* set of uncompressed     */
                                           /* luminance arrays        */
      } mipmap_luminance;
      struct Pmipmap_luminance_alpha {
         Plum_alpha_array_set lum_values;     /* set of uncompressed   */
                                              /* luminance/alpha arrays*/
      } mipmap_luminance_alpha;
      ...          /* implementation defined methods */
   } image_data;
} Pimage_res;
```

`Pin_class_data` **INPUT CLASS DATA RECORD**

```
 typedef union _Pin_class_data {

    Pint                 unused;
    Ploc_data            loc;
    Ploc_data3           loc3;
    Pstroke_data         stroke;
    Pstroke_data3        stroke3;
    Pval_data            val;
    Pval_data3           val3;
    Pchoice_data         choice;
    Pchoice_data3        choice3;
    Ppick_data           pick;
    Ppick_data3          pick3;
    Pstring_data         string;
    Pstring_data3        string3;
    Pset_data            set;
    Pcomposite_data      composite;
    ...                                /* other classes of input data */

 } Pin_class_data;
```

`Pin_class_measure` **INPUT CLASS MEASURE**

```
 typedef union _Pin_class_measure {

    struct {
       Pint         view_ind;
       Ppoint       loc_pos;
    } loc;

    struct {
       Pint         view_ind;
       Ppoint3      loc_pos;
    } loc3;

    struct {
       Pint         view_ind;
       Ppoint_list  points;
    } stroke;
    struct {
       Pint          view_ind;
       Ppoint_list3  points;
    } stroke3;

    Pfloat              val;

    struct {
       Pin_status    istat;
       Pint          selection;
    } choice;

    struct {
       Pin_status    istat;
       Ppick_path    *path;
    } pick;

    char                *string;

    Pset_measure                *set;

    Pcomposite_measure          *composite;

    ...                 /* other classes which may be added */

 } Pin_class_measure;
```

Pmarker_data **MARKER DATA**

```
 typedef struct {

    Pint                shape_type;     /* shape type                  */

    union _Pshape_data {
       Ppoint_list  polyline;           /* for PSHAPE_POLYLINE         */
       Ppoint_list  fill_area;          /* for PSHAPE_FILL_AREA        */
       Ppoint_list  convex_fill_area;   /* for PSHAPE_CONVEX_FILL_AREA */
       ...                              /* implementation defined      */
    } shape_data;                       /* marker shape data           */

 } Pmarker_data;
```

Ppattern **PATTERN**

```
typedef struct {
   Pint                 type;           /* pattern type */
   union _Ppattern_data {
      struct Ppat_colr_ind_array {
         Ppat_rep  pat_rep;             /* pattern representation      */
      } pat_colr_ind_array;
      struct Ppat_colr_array {
         Ppat_rep_pluspat_rep_plus;     /* pattern representation plus */
      } pat_colr_array;
      struct Ppat_image_res {
         Pint      image_res_id;        /* pattern image resource      */
      } pat_image_res;
      ...                               /* implementation defined      */
   } data;                              /* pattern data */
} Ppattern;
```

Ptarg_op **TARGET OPERATION**

```
typedef struct {
   Pint                 type;           /* target operation type       */
   union _Ptarget_op_data {
      struct Prend_targ {
         Ptarg_addrtarg_addr;           /* target address              */
      } rend_targ;
      struct Pdisp_targ {
         Ptarg_addrtarg_addr;           /* target address              */
      } disp_targ;
      struct Pclear_targ {
         Ptarg_addrtarg_addr;           /* target address              */
      } clear_targ;
      struct Pcopy_targ {
         Ptarg_addrsrc_targ_addr;       /* source target address       */
         Ptarg_addrdest_targ_addr;      /* destination target address  */
      } copy_targ;
      ...                               /* implementation defined      */
   } data;                              /* target operation data       */
} Ptarg_op;
```

Ptest **TEST**

```
 typedef struct {
    Pint                method;        /* test method                */
    union _Ptest_data {
       struct Ptest_always_fail {
          Pint                 unused;
       } test_always_fail;
       struct Ptest_always_succeed {
          Pint                 unused;
       } test_always_succeed;
       struct Ptest_extent_3 {
          Plimit3              rect;          /* extent rectangle        */
          Ptest_comp           relation;      /* test relation           */
          Pfloat               threshhold;    /* threshhold value        */
       } test_extent_3;
       struct Ptest_extent {
          Plimit               rect;          /* extent rectangle        */
          Ptest_comp           relation;      /* test relation           */
          Pfloat               threshhold;    /* threshhold value        */
       } test_extent;
       struct Ptest_bounds_3 {
          Ppoint_list_list3 point_lists;   /* test object             */
          Pclip_cond           cond;          /* test condition          */
       } test_bounds_3;
       struct Ptest_bounds {
          Ppoint_list_list  point_lists;   /* test object             */
          Pclip_cond           cond;          /* test condition          */
       } test_bounds;
       struct Ptest_nameset {
          Pint                 filter_type;   /* type of filter          */
          Pint                 filter_id;     /* filter identifier       */
          Pfilter_op           cond;          /* test condition          */
       } test_nameset;
       struct Ptest_appl_int {
          Pint                 appl_id;       /* appl. integer selector  */
          Ptest_comp_rel       comparison;    /* test comparison         */
          Pint                 data;          /* appl integer data value */
       } test_appl_int;
       struct Ptest_appl_real {
          Pint                 appl_id;       /* appl. real selector     */
          Ptest_comp_rel       comparison;    /* test comparison         */
          Pfloat               data;          /* appl real data value    */
       } test_appl_real;
```

```
        struct Ptest_cond_flags {
           Pcond_flags_eval_op  op;      /* condition flags eval. op.   */
        } test_cond_flags;
        struct Ptest_appl_int_as_logical {
           Pint                 appl_id;    /* appl. integer selector   */
           Ptest_comp_logical comparison;   /* test comparison          */
           Pint                 data;       /* appl integer data value  */
        } test_appl_int;
        ...                              /* implementation defined      */
     } test_data;        /*              test data record*/
  } Ptest;
```

Ptexture_compos **TEXTURE COMPOSITION DATA RECORD**

```
  typedef struct {
     Pint           method;         /* composition method              */
     union {
        Pint        unused;         /* used by methods 1, 2, and 4     */
        struct {                    /* used by method 3                */
          Pgcolr    environ;        /* environment colour              */
          Pint      gamma_chan_sel; /* gamma channel selector          */
        } blend_env;
        Pgcolr      background_colr;/* used by method 5                */
        ...                         /* implementation defined          */
     } data;
  } Ptexture_compos;
```

Ptrav_res_op **TRAVERSAL RESOURCE OPERATION**

```
  typedef struct {
     Pint              type;        /* trav. res. operation type   */
     union Ptrav_res_op_data {
        struct Ptrav_res_clear {
           Pint     res_id;         /* traversal resource id.      */
        } trav_res_clear;
        struct Ptrav_res_copy {
           Pint     src_res_id;     /* source traversal res. id.   */
           Pint     dest_res_id;    /* destination trav. res. id.  */
        } trav_res_copy;
        ...                              /* implementation defined      */
     } data;                             /* trav. res. operation data   */

  } Ptrav_res_op;
```

11.5 Implementation independent type definitions

`Paccess_flag` **ACCESS FLAG**

```
  typedef enum {

     PACCESS_REFERENCED = 0,
     PACCESS_COPIED = 1

  } Paccess_flag;
```

`Pack_process` **ACKNOWLEDGEMENT PROCESS**

```
  typedef struct {

     Pack_type           type;
     Pint                process_id;

  } Pack_process;
```

`Pack_process_list` **ACKNOWLEDGEMENT PROCESS LIST**

```
  typedef struct {

     Pint                count;
     Pack_process        *procs;

  } Pack_process_list;
```

`Pack_type` **ACKNOWLEDGEMENT TYPE**

```
  typedef enum {

     P_ACK_ACCEPTANCE = 0,
     P_ACK_NONACCEPTANCE = 1

  } Packnowledgement_type;
```

Palpha_facs **ALPHA FACILITIES**

```
typedef struct {

    Pint_list  alpha_srcs;      /* list available alpha sources      */
    Pint_list  transp_modes;    /* list transparency modes supported */

} Palpha_facs;
```

Passoc_flag **ASSOCIATION FlAG**

```
typedef enum {

    PNOTASSOCIATED = 0,
    PASSOCIATED = 1

} Passoc_flag;
```

Patomic_lid_facs **ATOMIC LOGICAL INPUT DEVICE FACILITIES**

```
typedef struct {

    Pint      max_simul_trig,   /* max simul. trigger procs          */
    Pint      max_simul_echo,   /* max simul. echo procs             */
    Pint      max_simul_ack,    /* max simul. acknowledgement procs  */
    Pint_list meas_proc_ids,    /* available measure process ids     */
    Pint_list trig_proc_ids,    /* available trigger process ids     */
    Pint_list echo_proc_ids,    /* available echo process ids        */
    Pint_list ack_proc_ids      /* available acknowledgement process ids
                                   */

} Patomic_lid_facs;
```

Pbackg_redisplay **BACKGROUND REDISPLAY FLAG**

```
typedef enum {

    PBACKG_REDISPLAY_OFF = 0,
    PBACKG_REDISPLAY_ON = 1

} Pbackg_redisplay;
```

Pbackg_style **BACKGROUND STYLE**

```
typedef enum {

    PBACKG_STYLE_TRANSPARENT = 0,
    PBACKG_STYLE_OPAQUE = 1

} Pbackg_style;
```

Pboolean **BOOLEAN**

```
typedef enum {

    PFALSE = 0,
    PTRUE = 1

} Pboolean;
```

Pborder_indic **BORDER INDICATOR**

```
typedef enum {

    PBORDER_OFF = 0,
    PBORDER_ON = 1

} Pborder_indic;
```

Pcieluva **CIE L* U* V* WITH ALPHA**

```
typedef struct {

    Pfloat     cieluv_x;         /* x coefficient                                 */
    Pfloat     cieluv_y;         /* y coefficient                                 */
    Pfloat     cieluv_y_lum;     /* y luminance                                   */
    Pfloat     alpha;            /* alpha                                         */

} Pcieluva;
```

`Pclip_cond` **CLIP CONDITION**

```
typedef enum {

  PCLIPCOND_IS_NOT_CLIPPED = 0,
  PCLIPCOND_IS_PARTIALLY_CLIPPED = 1,
  PCLIPCOND_IS_FULLY_CLIPPED = 2

} Pclip_cond;
```

`Pclosure` **CLOSURE TYPE**

```
typedef enum {

    PCLOSURE_PIE = 0,
    PCLOSURE_SEGMENT = 1

} Pclosure;
```

`Pcomposite_data` **COMPOSITE DATA**

```
typedef struct {

    Pint                   count;
    Pcomposite_individual *individual;

} Pcomposite_data;
```

`Pcomposite_facs` **COMPOSITE FACILITIES**

```
 typedef struct {

    Pint      max_simul_meas,        /* max simul. measure procs       */
    Pint      max_simul_trig,        /* max simul. trigger procs       */
    Pint      max_simul_echo,        /* max simul. echo procs          */
    Pint      max_simul_ack,         /* max simul. acknowledgement procs
                                        */
    Pint_list loc_meas_proc_ids,     /* avail. locator measure proc. ids
                                        */
    Pint_list stroke_meas_proc_ids,  /* avail. stroke measure proc. ids
                                        */
    Pint_list val_meas_proc_ids,     /* avail. valuator measure proc. ids
                                        */
    Pint_list choice_meas_proc_ids,  /* avail. choice measure proc. ids
                                        */
    Pint_list pick_meas_proc_ids,    /* avail. pick measure proc. ids  */
    Pint_list string_meas_proc_ids,  /* avail. string measure proc. ids
                                        */
    Pint_list set_meas_proc_ids,     /* avail. set measure proc. ids   */
    Pint_list composite_meas_proc_ids,/*avail. composite meas. proc. ids
                                        */
    Pint_list trig_proc_ids,         /* available trigger process ids  */
    Pint_list loc_echo_proc_ids,     /* available locator echo proc. ids
                                        */
    Pint_list stroke_echo_proc_ids,  /* available stroke echo proc. ids
                                        */
    Pint_list val_echo_proc_ids,     /* available valuator echo proc. ids
                                        */
    Pint_list choice_echo_proc_ids,  /* available choice echo proc. ids
                                        */
    Pint_list pick_echo_proc_ids,    /* available pick echo proc. ids  */
    Pint_list string_echo_proc_ids,  /* available string echo proc. ids
                                        */
    Pint_list set_echo_proc_ids,     /* available set echo proc. ids   */
    Pint_list comp_echo_proc_ids,    /* available composite echo proc. ids
                                        */
    Pint_list ack_proc_ids           /* available ack. proc. ids       */

 } Pcomposite_facs;
```

Pcomposite_individual **COMPOSITE INDIVIDUAL**

```
typedef struct _Pcomposite_individual {

    Pin_class        class;
    _Pin_class_data  *data;

} Pcomposite_individual;
```

Pcomposite_measure **COMPOSITE MEASURE**

```
typedef struct _Pcomposite_measure {

    Pint         count;                  /* number of measure components*/
    _Pmeasure    *measures;              /* list of measure components  */

} Pcomposite_measure;
```

Pcond_trav_facs **CONDITIONAL TRAVERSAL FACILITIES**

```
typedef struct {

   Pint      num_invis_filters;/*number of avail. invisibility filters
                                  */
   Pint      num_highl_filters;/*number of avail. highlighting filters
                                  */
   Pint      num_pick_filters; /*number of avail. pick filters      */
   Pint      num_appl_filters; /*number of avail. application filters
                                  */
   Pint      num_appl_integers;/*number of application integer      */
                              /*  values supported                  */
   Pint      num_appl_reals;  /*  number of application real values  */
                              /*  supported                         */
   Pint_list test_methods;    /*  list of test methods available     */

} Pcond_trav_facs;
```

`Pcond_flags_eval_op` **CONDITION FLAGS EVALUATION OPERATOR**

```
typedef enum {

   PCOND_ALL_ENABLED,
   PCOND_ALL_DISABLED,
   PCOND_NOT_ALL_DISABLED,
   PCOND_NOT_ALL_ENABLED,
   PCOND_ALWAYS_PASS,
   PCOND_NEVER_PASS

} Pcond_flags_eval_op;
```

Pcond_flag_mask **CONDITION FLAG MASK**

```
typedef union {

    unsigned int mask;
    struct _Pcond_flag_set {
      unsigned int flag0:1;
      unsigned int flag1:1;
      unsigned int flag2:1;
      unsigned int flag3:1;
      unsigned int flag4:1;
      unsigned int flag5:1;
      unsigned int flag6:1;
      unsigned int flag7:1;
      unsigned int flag8:1;
      unsigned int flag9:1;
      unsigned int flag10:1;
      unsigned int flag11:1;
      unsigned int flag12:1;
      unsigned int flag13:1;
      unsigned int flag14:1;
      unsigned int flag15:1;
      unsigned int flag16:1;
      unsigned int flag17:1;
      unsigned int flag18:1;
      unsigned int flag19:1;
      unsigned int flag20:1;
      unsigned int flag21:1;
      unsigned int flag22:1;
      unsigned int flag23:1;
      unsigned int flag24:1;
      unsigned int flag25:1;
      unsigned int flag26:1;
      unsigned int flag27:1;
      unsigned int flag28:1;
      unsigned int flag29:1;
      unsigned int flag30:1;
      unsigned int flag31:1;
    } flags;

} Pcond_flag_mask;
```

Pcond_flags **CONDITION FLAGS**

```
typedef struct {

   Pcond_flag_mask    enable_mask;   /* condition flag enable mask  */
   Pcond_flag_mask    disable_mask;  /* condition flag disable mask */

} Pcond_flags;
```

Pcond_flag_test **CONDITION FLAG TEST**

```
typedef struct {

   Ptest              test;          /* test operation              */
   Pcond_flags        mask;          /* condition test mask         */

} Pcond_flag_test;
```

Pcond_test_list **CONDITION TEST LIST**

```
typedef struct {

   Pint               num_tests;    /*  number of condition test    */
   Pcond_flag_test    *tests;       /*  list of condition flag tests*/

} Pcond_test_list;
```

Pconfig_name **CONFIGURATION NAME**

```
typedef struct {

   Pint            name_size;       /* size of name                 */
   char            *name;           /* configuration setting name   */

} Pconfig_name;
```

Pconfig_name_list **CONFIGURATION NAME LIST**

```
typedef struct {

   Pint            list_size;       /* number of names              */
   Pconfig_name    *name;           /* configuration name list      */

} Pconfig_name_list;
```

`Pconfig_setting_facs` **CONFIGURATION SETTING FACILITIES**

```
 typedef struct {
    Pint          num_supported; /* number of simultaneously supported
                                          */
                                 /* device coordinate clip regions    */
 } Pconfig_setting_facs;
```

`Pconfig_settings` **CONFIGURATION SETTINGS**

```
 typedef struct {
    char             *name;          /* configuration setting name    */
    Pconfig_value    *value;         /* configuration setting value   */
 } Pconfig_settings;
```

`Pconfig_value` **CONFIGURATION VALUE**

```
 typedef union {
    Pint         int_value;            /* integer configuration value   */
    Pfloat       real_value;           /* real configuration value      */
    char         *string_value;        /* string configuration value    */
 } Pconfig_value;
```

`Pdc_clip_facs` **DEVICE COORDINATE CLIP REGION FACILITIES**

```
 typedef struct {
    Pint              num_supported;   /* number of simultaneously sup-
                                          ported*/
                                       /* device coordinate clip regions */
 } Pdc_clip_facs;
```

Pdepth_cue_mask **DEPTH CUE MASK**

```
typedef struct {

    unsigned int mode:1;
    unsigned int ref_planes:1;
    unsigned int scale:1;
    unsigned int gcolr:1;
    unsigned int :28;

} Pdepth_cue_mask;
```

Pdi_mode **DIRECT INTERPRETATION MODE**

```
typedef enum {

    PDIM_INTERPRET_NONE = 0,
    PDIM_INTERPRET_STATE = 1,
    PDIM_INTERPRET_ALL = 2

} Pdi_mode;
```

Pdi_trav_facs **DIRECT INTERPRETATION TRAVERSAL FACILITIES**

```
typedef struct {

    Pint     num_trav_procs;         /* number of simultaneously posted
                                        */
                                     /* traversal processes            */
    Pint     num_rend_algorithms;    /* number of multi-pass algorithms
                                        */
                                     /* supported for rendering        */
    Pint     num_pick_algorithms;    /* number of multi-pass algortihms
                                        */
                                     /* supported for picking          */

} Pdi_trav_facs;
```

Pdyns_post_grps **DYNAMICS OF POSTING GROUPS**

```
typedef struct {

    Pdyn_mod           posting_status;/* posting status modification */

} Pdyns_post_grps;
```

`Pdyns_ws_attrs_texture` **DYNAMICS OF WORKSTATION ATTRIBUTES TEXTURE**

```
typedef struct {
    Pdyn_mod   texture_rep;         /* texture representation        */
} Pdyns_ws_attrs_texture;
```

`Pecho_measure_proc` **ECHO MEASURE PROCESS**

```
typedef struct {
    Pin_class           dev_class;   /* input class                   */
    Pint                instance;    /* i = ith of class in LID def.  */
} Pecho_measure_proc;
```

`Pecho_process` **ECHO PROCESS**

```
typedef struct {
    Pint                pet;
    Pecho_measure_proc *measures_echoed;
    Pint                process_id;
} Pecho_process;
```

`Pecho_process_list` **ECHO PROCESS LIST**

```
typedef struct {
    Pint                count;
    Pecho_process      *procs;
} Pecho_process_list;
```

Pedge_bundle_full **EDGE BUNDLE FULL**

```
typedef struct {

   Pedge_flag          flag;          /* edge flag                    */
   Pint                type;          /* edgetype                     */
   Pfloat              width;         /* edgewidth scale factor       */
   Pgolr               colr;          /* edge colour                  */
   Plinetype_adapt     adapt;         /* edgetype adaptability        */
   Plinetype_cont      cont;          /* edgetype continuity          */
   Pfloat              offset;        /* edgetype offset              */
   Plinecap            cap;           /* edgecap                      */
   Plinejoin           join;          /* edgejoin                     */
   Pfloat              limit;         /* edgemitre limit              */

} Pedge_bundle_full;
```

Pedge_mask **EDGE MASK**

```
typedef struct {

   unsigned int flag:1;
   unsigned int type:1;
   unsigned int width:1;
   unsigned int colr:1;
   unsigned int adapt:1;
   unsigned int cont:1;
   unsigned int offset:1;
   unsigned int cap:1;
   unsigned int join:1;
   unsigned int limit:1;
   unsigned int :22;

} Pedge_mask;
```

Pedgetype_adapt **EDGETYPE ADAPTABILITY**

```
typedef enum {

   PEDTA_EXACT = 0,
   PEDTA_ADAPT = 1

} Pedgetype_adapt;
```

Pedgecap **EDGECAP**

```
  typedef enum {

     PEDC_BUTT = 0,
     PEDC_ROUND = 1,
     PEDC_SQUARE = 2

  } Pedgecap;
```

Pedgetype_cont **EDGETYPE CONTINUITY**

```
  typedef enum {

     PEDTC_CONTINUOUS = 0,
     PEDTC_RESTART = 1

  } Pedgetype_cont;
```

Pedgejoin **EDGEJOIN**

```
  typedef enum {

     PEDJ_FLAT = 0,
     PEDJ_MITRE = 1,
     PEDJ_ROUND = 2,
     PEDJ_BEVEL = 3

  } Pedgejoin;
```

Pext_pat_facs **EXTENDED PATTERN FACILITIES**

```
  typedef struct {

     Pint_list pattern_types;      /* list of supported pattern types   */
     Pint_list image_spec_methods;/* list of supported image specifcation
                                     */
                                  /* methods                            */

  } Pext_pat_facs;
```

`Pfilter_op` **FILTER RELATIONS**

```
typedef enum {

    PFILTER_FAILS = 0,
    PFILTER_PASSES = 1

} Pfilter_op;
```

`Pfloat_array` **FLOAT ARRAY**

```
typedef struct {

    Pint_size      dims;              /* float array dimensions      */
    Pfloat         *floats;           /* float array                 */

} Pfloat_array;
```

`Pfloat_array_set` **FLOAT ARRAY SET**

```
typedef struct {

    Pint           num_arrays;      /* number of float arrays in set */
    Pfloat_array   *floats;         /* float arrays                  */

} Pfloat_array_set;
```

`Pgcolr_array` **GENERAL COLOUR ARRAY**

```
typedef struct {

    Pint           colr_type;         /* colour type                 */
    Pcolrv_array   colrs;             /* colour array                */

} Pgcolr_array;
```

`Pgcolr_array_set` **GENERAL COLOUR ARRAY SET**

```
typedef struct {

    Pint           colr_type;         /* colour type                 */
    Pint           num_arrays;        /* number of arrays in set     */
    Pcolrv_array   *colrs;            /* set of colour arrays        */

} Pgcolr_array_set;
```

Phighl_facs **HIGHLIGHTING FACILITIES**

```
typedef struct {

    Pint            num_pred;       /* number of predefined methods   */
    Pint_list       methods;        /* number of highl. methods supp. */
    Pint            max_highl_ind;  /* max highl. table indices       */

} Phighl_facs;
```

Phlsa **HUE LIGHTNESS SATURATION WITH ALPHA**

```
typedef struct {

    Pfloat     hue;              /* hue                                */
    Pfloat     lightness;        /* lightness                          */
    Pfloat     saturation;       /* saturation                         */
    Pfloat     alpha;            /* alpha                              */

} Phlsa;
```

Phsva **HUE SATURATION VALUE WITH ALPHA**

```
typedef struct {

    Pfloat     hue;              /* hue                                */
    Pfloat     saturation;       /* saturation                         */
    Pfloat     value;            /* value                              */
    Pfloat     alpha;            /* alpha                              */

} Phsva;
```

Pimage_res_facs **IMAGE RESOURCE FACILITIES**

```
typedef struct {

    Pint_list image_spec_methods;/* list of supported image          */
                                 /* specification methods            */
    Pint      num_def_ids;       /* number of definable image resource
                                    */
                                 /* identifiers                      */
    Pint      num_predef_ids;    /* number of predefined image resources
                                    */

} Pimage_res_facs;
```

`_Pin_class_data` **INPUT CLASS DATA RECORD**

```
union _Pin_class_data;   /* used for defining Pset_data and
                            Pcomposite_data; */
```

`Pin_class_measure` **INPUT CLASS MEASURE**

```
union _Pin_class_measure;   /* used for defining Pmeasure and
                               Pset_measure */
```

`Pinterior_mask` **INTERIOR MASK**

```
typedef struct {

   unsigned int style:1;
   unsigned int style_ind:1;
   unsigned int colr:1;
   unsigned int shad_method:1;
   unsigned int :28;

} Pinterior_mask;
```

`Plight_attach` **LIGHT SOURCE ATTACHMENT**

```
typedef struct {

   Pint     light_ws_id;             /* workstation id for view     */
   Pint     light_src_ind;           /* view index                  */
   Pint     light_action_type;       /* view action type            */
   Pint     transfer_type;           /* measure transfer type       */

} Plight_attach;
```

`Plimit_list` **LIMIT LIST**

```
typedef struct {

   Pint                 num_limits;    /* number of limits            */
   Plimit               *limits;       /* list of limits              */

} Plimit_list;
```

Plimit3_list **LIMIT 3 LIST**

```
typedef struct {

    Pint                num_limits;     /* number of limits            */
    Plimit3             *limits;        /* list of limits              */

} Plimit3_list;
```

Pline_bundle_full **POLYLINE BUNDLE FULL**

```
typedef struct {

    Pint                type;           /* linetype                    */
    Pfloat              width;          /* linewidth scale factor      */
    Pgolr               colr;           /* polyline colour             */
    Pint                shad_method;    /* polyline shading method     */
    Pcurve_approx_crit_data
                        curve_approx_crit_data;
                                        /* curve approx. criteria      */
    Plinetype_adapt     adapt;          /* linetype adaptability       */
    Plinetype_cont      cont;           /* linetype continuity         */
    Pfloat              offset;         /* linetype offset             */
    Plinecap            cap;            /* linecap                     */
    Plinejoin           join;           /* linejoin                    */
    Pfloat              limit;          /* linemitre limit             */

} Pline_bundle_full;
```

Plinecap **LINECAP**

```
typedef enum {

    PLNC_BUTT = 0,
    PLNC_ROUND = 1,
    PLNC_SQUARE = 2

} Plinecap;
```

`Plinejoin` **LINEJOIN**

```
typedef enum {

    PLNJ_FLAT = 0,
    PLNJ_MITRE = 1,
    PLNJ_ROUND = 2,
    PLNJ_BEVEL = 3

} Plinejoin;
```

`Plinetype_adapt` **LINETYPE ADAPTABILITY**

```
typedef enum {

    PLNTA_EXACT = 0,
    PLNTA_ADAPT = 1

} Plinetype_adapt;
```

`Plinetype_cont` **LINETYPE CONTINUITY**

```
typedef enum {

    PLNTC_CONTINUOUS = 0,
    PLNTC_RESTART = 1

} Plinetype_cont;
```

`Plinetype_def_facs` **LINETYPE DEFINITION FACILITIES**

```
typedef struct {

    Pint      num_definable;       /* number definable linetypes       */
    Pint      num_predefined;      /* number predefined linetypes      */
    Pint      max_dash_segments;   /* maximum dash segments supported */
    Pfloat    offset;              /* offset to definable linetypes   */
    Pint_list predef_def_linetypes /* list of predef. definable linetypes
                                      */

} Plinetype_def_facs;
```

Plum_alpha_array **LUMINANCE ALPHA ARRAY**

```
 typedef struct {

    Pint_size        dims;        /* array dimensions                  */
    Plum_alpha_pair  *values;     /* 2D array of luminance/alpha pairs */

 } Plum_alpha_array;
```

Plum_alpha_array_set **LUMINANCE ALPHA ARRAY SET**

```
 typedef struct {

    Pint             num_arrays;  /* number of arrays in set           */
    Plum_alpha_array *sets;       /* set of 2D luminance/alpha arrays  */

 } Plum_alpha_array_set;
```

Plum_alpha_pair **LUMINANCE ALPHA PAIR**

```
 typedef struct {

    Pfloat          lum_value;           /* luminance_value            */
    Pfloat          alpha_value;         /* alpha value                */

 } Plum_alpha_pair;
```

Pmarker_desc **MARKER DESCRIPTOR**

```
 typedef struct {

    Pint             number_shapes;      /* number of shapes           */
    Pmarker_data     *marker;            /* array of shape descriptors */

 } Pmarker_desc;
```

`Pmarker_type_def_facs` **MARKER TYPE DEFINITION FACILITIES**

```
 typedef struct {

    Pint       num_definable;       /* number definable marker types  */
    Pint       num_predefined;      /* number predefined marker types */
    Pfloat     offset;              /* offset to definable linetypes  */
    Pint       max_allowable_pts;   /* maximum allowable points       */
    Pint_list  shape_types;         /* list of available shape types  */
    Pint_list  predef_marker_types  /* list of predefined marker types
                                       */

 } Pmarker_type_def_facs;
```

`Pmeasure` **MEASURE**

```
 typedef struct _Pmeasure {

    Pin_class           class;
    _Pin_class_measure *data;

 } Pmeasure;
```

`Pmeas_process` **MEASURE PROCESS**

```
 typedef struct {

    Pin_class           dev_class;
    Pint                process_id;

 } Pmeas_process;
```

`Pmeas_process_set` **MEASURE PROCESS SET**

```
 typedef struct {

    Pint                set_maxsize;   /* maximum set size            */
    Pmeas_process       process_id;    /* measure process identifier  */

 } Pmeas_process_set;
```

Pmeas_process_list **MEASURE PROCESS LIST**

```
typedef struct _Pmeas_process_list {

    Pint               count;          /* number of process in list   */
    Pmeas_process     *measure_procs; /* list of measure processes   */

} Pmeas_process_list;
```

Pmipmap_facs **MIPMAP FACILITIES**

```
typedef struct {

    Pint      max_dims[3];      /* max dimensions mipmap base level   */
    Pboolean  equal_dims_req;   /* equal mipmap dimensions required   */
    Pboolean  power_2_dims_req; /* power of 2 mipmap dimensions req.  */

} Pmipmap_facs;
```

Pna_in_status **NON-ATOMIC INPUT STATUS**

```
typedef enum {

    PNA_IN_STATUS_NONE = 0,
    PNA_IN_STATUS_INCOMPLETE = 1,
    PNA_IN_STATUS_OK = 2

} Pna_in_status;
```

Pnum_na_in **NUMBER NON-ATOMIC INPUT DEVICES**

```
typedef struct {

    Pint         set;            /* number available set LIDs        */
    Pint         composite;      /* number available composite LIDs  */

} Pnum_na_in;
```

Ppattern_mask **PATTERN MASK**

```
 typedef struct {

    unsigned int pat_bundle:1;
    unsigned int pat_rep_plus:1;
    unsigned int ext_pat_rep:1;
    unsigned int :29

 } Ppattern_mask;
```

Pperspect_corr **PERSPECTIVE CORRECTION**

```
 typedef enum {

    PPERSP_CORR_NONE = 0,
    PPERSP_CORR_AT_VERTICES = 1,
    PPERSP_CORR_INTERIOR = 2

 } Pperspect_corr;
```

Ppick_mapping_facs **PICK MAPPING FACILITIES**

```
 typedef struct {

    Pint_list  avail_pick_types;     /* list available pick types    */
    Pint_list  avail_echo_types;     /* list of available echo types */

 } Ppick_mapping_facs;
```

Ppick_path_list **PICK PATH LIST**

```
 typedef struct _Ppick_path_list {

    Pint        num_paths;       /* number paths in list             */
    Ppick_path  *paths;          /* array of pick paths              */

 } Ppick_path_list;
```

Ppick_stat **PICK STATUS**

```
typedef enum {
    PPICK_STAT_FALSE = 0,
    PPICK_STAT_TRUE = 1
} Ppick_stat;
```

Ppict_stat **PICTURE STATUS**

```
typedef enum {
    PPICT_STAT_COMP = 0,
    PPICT_STAT_INCOMP = 1
} Ppict_stat;
```

Ppolyline_mask **POLYLINE MASK**

```
typedef struct {
    unsigned int type:1;
    unsigned int width:1;
    unsigned int colr:1;
    unsigned int shad_method:1;
    unsigned int curve_approx_crit_data:1;
    unsigned int adapt:1;
    unsigned int cont:1;
    unsigned int offset:1;
    unsigned int cap:1;
    unsigned int join:1;
    unsigned int limit:1;
    unsigned int :21;
} Ppolyline_mask;
```

Ppolymarker_mask **POLYMARKER MASK**

```
typedef struct {
    unsigned int type:1;
    unsigned int size:1;
    unsigned int colr:1;
    unsigned int :29;
} Ppolymarker_mask;
```

`Pposted_ind` **POSTED INDICATOR**

```
 typedef enum {
    PPOSTED_FALSE = 0,
    PPOSTED_TRUE = 1
 } Pposted_ind;
```

`Ppost_grp_facs` **POSTING GROUP FACILITIES**

```
 typedef struct {
    Pint        num_def_grps;       /* number of definable posting groups
                                       */
    Pint        num_predef_grps;    /* number of predefined posting groups
                                       */
    Pint_list   predef_grps;        /* list of predefined posting groups
                                       */
    Pint_list   backg_methods;      /* list of supported posting group */
                                    /* background methods              */
 } Ppost_grp_facs;
```

`Ppost_grp_status` **POSTING GROUP STATUS**

```
 typedef enum {
    PPGSTAT_INACTIVE = 0,
    PPGSTAT_ACTIVE = 1
 } Ppost_grp_status;
```

`Pref_type` **REFERENCE TYPE**

```
 typedef enum {
    PREF_EXECUTE = 0,
    PREF_INSTANCE = 1
 } Pref_type;
```

`Prefl_mask` **REFLECTANCE MASK**

```
typedef struct {

    unsigned int refl_model:1;
    unsigned int data:1;
    unsigned int :30;

} Prefl_mask;
```

`Prgba` **RED GREEN BLUE WITH ALPHA**

```
typedef struct {

    Pfloat     red;            /* red intensity                    */
    Pfloat     green;          /* green intensity                  */
    Pfloat     blue;           /* blue intensity                   */
    Pfloat     alpha;          /* alpha                            */

} Prgba;
```

`Pset_data` **SET DATA RECORD**

```
typedef struct _Pset_data {

    Pin_class        dev_class;
    Pint             num_in_set;
    _Pin_class_data  *data;

} Pset_data;
```

Pset_facs **SET FACILITIES**

```
typedef struct {

    Pint      max_measures,            /* max instances                */
    Pint      max_simul_trig,          /* max simul. trigger procs     */
    Pint      max_simul_echo,          /* max simul. echo procs        */
    Pint      max_simul_ack,           /* max simul. ack. procs        */
    Pint_list loc_meas_proc_ids,       /* avail locator meas proc ids */
    Pint_list stroke_meas_proc_ids,    /* avail stroke meas proc ids  */
    Pint_list val_meas_proc_ids,       /* avail valuator meas proc ids*/
    Pint_list choice_meas_proc_ids,    /* avail choice meas proc ids  */
    Pint_list pick_meas_proc_ids,      /* avail pick meas proc ids    */
    Pint_list string_meas_proc_ids,    /* avail string meas proc ids  */
    Pint_list set_meas_proc_ids,       /* avail set meas proc ids     */
    Pint_list composite_meas_proc_ids, /* avail compos meas proc ids  */
    Pint_list trig_proc_ids,           /* avail trigger proc ids      */
    Pint_list loc_echo_proc_ids,       /* avail locator echo proc ids */
    Pint_list stroke_echo_proc_ids,    /* avail stroke echo proc ids  */
    Pint_list val_echo_proc_ids,       /* avail valuator echo proc ids*/
    Pint_list choice_echo_proc_ids,    /* avail choice echo proc ids  */
    Pint_list pick_echo_proc_ids,      /* avail pick echo proc ids    */
    Pint_list string_echo_proc_ids,    /* avail string echo proc ids  */
    Pint_list set_echo_proc_ids,       /* avail set echo proc ids     */
    Pint_list comp_echo_proc_ids,      /* avail compos echo proc ids  */
    Pint_list ack_proc_ids             /* avail ack. proc ids         */

} Pset_facs;
```

Pset_measure **SET MEASURE**

```
typedef struct _Pset_measure {

    Pin_class           dev_class;     /* input class of set           */
    Pint                set_size;      /* size of measure set          */
    _Pin_class_measure  *measure_set;  /* array of measures of same    */
                                       /* class                        */

} Pset_measure;
```

Psupport_indication **SUPPORT INDICATION**

```
typedef enum {

    PSI_NOT_SUPPORTED = 0,
    PSI_SUPPORTED = 1

} Psupport_indication;
```

`Ptarg_addr` **TARGET ADDRESS**

```
typedef struct {
   Ptarg_type          ref_targ;      /* reference target          */
   Pint                offset;        /* offset from reference target*/
} Ptarg_addr;
```

`Ptarg_data_st` **TARGET DATA STATE**

```
typedef enum {
   PTARG_DATA_ST_ABSENT = 0,
   PTARG_DATA_ST_PRESENT = 1
} Ptarg_data_st;
```

`Ptarg_empty_st` **TARGET EMPTY**

```
typedef enum {
   PTARG_EMPTY_ST_EMPTY = 0,
   PTARG_EMPTY_ST_NOT_EMPTY = 1
} Ptarg_empty_st;
```

`Ptarget_facs` **TARGET FACILITIES**

```
typedef struct {
   Pint            num_avail_targets; /* number of available targets */
} Ptarget_facs;
```

`Ptarg_op_list` **TARGET OPERATION LIST**

```
typedef struct {
   Pint                num_ops;       /* number of target operations */
   Ptarg_op            *targ_ops;     /* list of target operations   */
} Ptarg_op_list;
```

`Ptarg_type` **TARGET TYPE**

```
typedef enum {

    PBASE_TARG = 0,
    PREND_TARG = 1,
    PDISP_TARG = 2

} Ptarg_type;
```

`Ptarg_type_list` **TARGET TYPE LIST**

```
typedef struct {

    Pint                num_targ_types;/* number of target types       */
    Ptarg_type          *targ_types;   /* list of target types         */

} Ptarg_type_list;
```

`Ptest_comp_rel` **TEST COMPARISON RELATIONAL**

```
typedef enum {

    PTRL_EQUAL = 0,
    PTRL_NOT_EQUAL = 1,
    PTRL_GREATER = 2,
    PTRL_LESS = 3,
    PTRL_GREATER_OR_EQUAL = 4,
    PTRL_LESS_OR_EQUAL = 5

} Ptest_comp_rel;
```

`Ptest_comp_logical` **TEST COMPARISON LOGICAL**

```
typedef enum {

    PTLG_BITWISE_AND = 1,
    PTLG_BITWISE_OR = 2,
    PTLG_BITWISE_XOR = 3

} Ptest_comp_logical;
```

Ptext_mask **TEXT MASK**

```
typedef struct {

   unsigned int font:1;
   unsigned int precision:1;
   unsigned int char_expan:1;
   unsigned int char_space:1;
   unsigned int colr:1;
   unsigned int :27;

} Ptext_mask;
```

Ptexture_binding **TEXTURE BINDING**

```
typedef struct {

   Pint     image_res_id;        /* texture image resource identifier */
   Pint     rendering_order;     /* rendering order                   */

} Ptexture_binding;
```

Ptexture_facs **TEXTURE FACILITIES**

```
typedef struct {

   Pint_list  coord_srcs;      /* list avail. coordinate sources     */
   Pint_list  compos_methods;  /* list avail. composition methods    */
   Pint_list  min_methods;     /* list avail. minification methods   */
   Pint_list  mag_methods;     /* list avail. magnification methods  */
   Pint_list  bound_cond;      /* list avail. boundary conditions    */
   Pint_list  clamp_methods;   /* list avail. boundary clamp methods */
   Pint_list  render_orders;   /* list avail. rendering orders       */
   Pint       num_pred_inds;   /* number predefined texture indices  */

} Ptexture_facs;
```

Ptexture_map_facs **TEXTURE MAPPING FACILITIES**

```
typedef struct {

   Pint      max_simul;          /* max number simultaneously          */
                                 /* appliable textures                 */
   Pint_list perspect_corr;      /* list of available perspective      */
                                 /* correction methods                 */
   Pint_list sampling_freqs;     /* list avail. sampling frequencies   */
   Pint_list opt_hints;          /* list of available resource         */
                                 /* optimization hints                 */
   Ptup_util usage;              /* utilization of texture usage       */
                                 /* priorities                         */
   Pint_list image_res_types;    /* list of image resource types       */
                                 /* available for texture mapping      */

} Ptexture_map_facs;
```

Ptexture_mask **TEXTURE MASK**

```
typedef struct {

   unsigned int coord_src:1;
   unsigned int coord_src_data:1;
   unsigned int orientation:1;
   unsigned int compos_method:1;
   unsigned int compos_data:1;
   unsigned int min_method:1;
   unsigned int max_method:1;
   unsigned int bound_conds:1;
   unsigned int clamp_method:1
   unsigned int clamp_data:1
   unsigned int depth_sampl_hint:1
   unsigned int freq_wt_hints:1
   unsigned int image_res_id:1
   unsigned int rendering_order:1
   unsigned int :18

} Ptexture_mask;
```

Ptexture_param **TEXTURE PARAMETRIZATION**

```
typedef struct {

   Pcoord_src      coord_src;         /* coordinate source and data   */
   Pmatrix3        orientation;       /* orientation matrix           */

} Ptexture_param;
```

Ptexture_rep **TEXTURE REPRESENTATION**

```
typedef struct {

   Ptexture_param    param;            /* texture parametrization     */
   Ptexture_compos   compos;           /* texture composition         */
   Ptexture_sampling sampling;         /* texture sampling            */
   Ptexture_binding  binding;          /* texture binding             */

} Ptexture_rep;
```

Ptexture_rep **TEXTURE RESOURCE OPTIMIZATION HEURISTICS**

```
typedef struct {

   Pint              opt_hint;           /* optimization hint         */
   Pint_list         usage_priorities;   /* texture usage priorities */

} Ptexture_res_opt_heur;
```

Ptexture_sampling **TEXTURE SAMPLING**

```
typedef struct {

   Pint          min_method;          /* minification method          */
   Pint          mag_method;          /* magnification method         */
   Pint          bound_conds[3];      /* boundary conditions          */
   Pclamp_method clamp;               /* boundary clamp method & data*/
   Pfloat        depth_sampl_hint;    /* depth sampling hint          */
   Pfloat        freq_wt_hints[3];    /* frequency weight hints       */

} Ptexture_sampling;
```

Ptrav_res **TRAVERSAL RESOURCE**

```
typedef struct {

   Pint                 id;          /* traversal resource identifier */
   Pint                 type;        /* traversal resource type       */

} Ptrav_res;
```

`Ptrav_res_facs` **TRAVERSAL RESOURCE FACILITIES**

```
 typedef struct {

    Ptrav_res_list avail_trav_res;  /* list of avail. traversal resources
                                       */

 } Ptrav_res_facs;
```

`Ptrav_res_list` **TRAVERSAL RESOURCE LIST**

```
 typedef struct {

    Pint             num_trav_res;   /* number of traversal resources  */
    Ptrav_res        *trav_res;      /* list of traversal resources    */

 } Ptrav_res_list;
```

`Ptrav_type` **TRAVERSAL TYPE**

```
 typedef enum {

    PTRAV_RENDER = 0,
    PTRAV_PICK = 1

 } Ptrav_type;
```

`Ptrig_process` **TRIGGER PROCESS**

```
 typedef struct {

    Ptrig_type            type;
    Pint                  process_id;

 } Ptrig_process;
```

`Ptrig_process_list` **TRIGGER PROCESS LIST**

```
 typedef struct {

    Pint                  count;
    Ptrig_process         *procs;

 } Ptrig_process_list;
```

`Ptrig_type` **TRIGGER TYPE**

```
typedef enum {

    PTRIGGER_BREAK = 0,
    PTRIGGER_SELECT = 1,
    PTRIGGER_EVENT = 2,
    PTRIGGER_DELETE = 3

} Ptrig_type;
```

`Ptup_util` **TEXTURE USAGE PRIORITIES UTILIZATION**

```
typedef enum {

    PTUP_UTILIZED_NO = 0,
    PTUP_UTILIZED_YES = 1

} Ptup_util;
```

`Pval_facs` **VALUATOR FACILITIES**

```
typedef struct {

    Pint      max_simul_trig,   /* max simul. trigger procs          */
    Pint      max_simul_echo,   /* max simul. echo procs             */
    Pint      max_simul_ack,    /* max simul. acknowledgement procs  */
    Pint_list meas_proc_ids,    /* available measure process ids     */
    Pint_list trig_proc_ids,    /* available trigger process ids     */
    Pint_list echo_proc_ids,    /* available echo process ids        */
    Pint_list ack_proc_ids      /* available acknowledgement process ids
                                   */

} Pval_facs;
```

`Pview_attach` **VIEW ATTACHMENT**

```
typedef struct {

    Pint    view_ws_id;             /* workstation id for view     */
    Pint    view_ind;               /* view index                  */
    Pint    view_action_type;       /* view action type            */
    Pint    transfer_type;          /* measure transfer type       */
    Pint    view_upd_method;        /* view update method          */

} Pview_attach;
```

`Pview_status` **VIEW_STATUS**

```
 typedef enum {
    PVIEW_ST_CORRESPOND = 0,
    PVIEW_ST_DIFFERENT = 1
 } Pview_status;
```

`Pwatch_enable` **WATCH ENABLE FLAG**

```
 typedef enum {
    PWATCH_OFF = 0,
    PWATCH_ON = 1
 } Pwatch_enable;
```

`Pws_st_tables_highl` **LENGTHS OF WORKSTATION STATE TABLES HIGHLIGHTING**

```
 typedef struct {
    Pint   highl_rep;       /* max. # of highlighting table entries  */
 } Pws_st_tables_highl;
```

`Pws_st_tables_texture` **LENGTHS OF WORKSTATION STATE TABLES TEXTURE**

```
 typedef struct {
    Pint   texture_rep;        /* max. # of texture table entries    */
 } Pws_st_tables_texture;
```

12 C *Full PHIGS profile* macro definitions

12.1 Function identifiers

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

Function Name Macro	Function Number
`#define Pfn_ws_type_create`	`(268)`
`#define Pfn_ws_type_set`	`(269)`
`#define Pfn_ws_type_get`	`(270)`
`#define Pfn_ws_type_destroy`	`(271)`
`#define Pfn_redraw_all_structs_from_grp`	`(272)`
`#define Pfn_redraw_all_structs_on_targ`	`(273)`
`#define Pfn_redraw_all_structs_from_grp_on_targ`	`(274)`
`#define Pfn_upd_targ`	`(275)`
`#define Pfn_define_post_grp`	`(276)`
`#define Pfn_undefine_post_grp`	`(277)`
`#define Pfn_set_post_grp_status`	`(278)`
`#define Pfn_set_post_grp_priority`	`(279)`
`#define Pfn_set_post_grp_backg_style`	`(280)`
`#define Pfn_set_post_grp_backg_method`	`(281)`
`#define Pfn_set_post_grp_border_indic`	`(282)`
`#define Pfn_set_post_grp_border_ind`	`(283)`
`#define Pfn_assoc_image_res`	`(284)`
`#define Pfn_disassoc_image_res`	`(285)`
`#define Pfn_set_dc_clip_regions3`	`(286)`
`#define Pfn_set_dc_clip_regions`	`(287)`
`#define Pfn_set_targ_manip_mode`	`(288)`
`#define Pfn_set_targ_dispos`	`(289)`
`#define Pfn_set_disp_targ`	`(290)`
`#define Pfn_set_rend_targ`	`(291)`
`#define Pfn_clear_targ`	`(292)`
`#define Pfn_copy_targ`	`(293)`
`#define Pfn_create_targ`	`(294)`
`#define Pfn_destroy_targ`	`(295)`
`#define Pfn_set_st_visual_rep`	`(296)`
`#define Pfn_set_targ_st_visual_rep`	`(297)`
`#define Pfn_assoc_trav_res`	`(298)`
`#define Pfn_disassoc_trav_res`	`(299)`
`#define Pfn_manip_trav_res`	`(300)`
`#define Pfn_reset_all_trav_res`	`(301)`
`#define Pfn_ret_window_system_colr`	`(302)`
`#define Pfn_set_appl_int`	`(303)`
`#define Pfn_set_appl_real`	`(304)`
`#define Pfn_circle3`	`(305)`
`#define Pfn_circle`	`(306)`
`#define Pfn_circular_arc3`	`(307)`

```
#define Pfn_circular_arc                      (308)
#define Pfn_ellipse3                          (309)
#define Pfn_ellipse                           (310)
#define Pfn_elliptical_arc3                   (311)
#define Pfn_elliptical_arc                    (312)
#define Pfn_fill_circle3                      (313)
#define Pfn_fill_circle                       (314)
#define Pfn_circular_arc_close3               (315)
#define Pfn_circular_arc_close                (316)
#define Pfn_fill_ellipse3                     (317)
#define Pfn_fill_ellipse                      (318)
#define Pfn_elliptical_arc_close3             (319)
#define Pfn_elliptical_arc_close              (320)
#define Pfn_set_highl_ind                     (321)
#define Pfn_set_linetype_adapt                (322)
#define Pfn_set_linetype_cont                 (323)
#define Pfn_set_linetype_offset               (324)
#define Pfn_set_linecap                       (325)
#define Pfn_set_linejoin                      (326)
#define Pfn_set_linemitre_limit               (327)
#define Pfn_set_edgetype_adapt                (328)
#define Pfn_set_edgetype_cont                 (329)
#define Pfn_set_edgetype_offset               (330)
#define Pfn_set_edgecap                       (331)
#define Pfn_set_edgejoin                      (332)
#define Pfn_set_edgemitre_limit               (333)
#define Pfn_set_highl_method                  (334)
#define Pfn_set_transparency                  (335)
#define Pfn_set_back_transparency             (336)
#define Pfn_set_alpha_src_sel                 (337)
#define Pfn_set_alpha_data_sel_ind            (338)
#define Pfn_set_active_textures               (339)
#define Pfn_set_back_active_textures          (340)
#define Pfn_set_texture_perspect_corr         (341)
#define Pfn_set_texture_sampling_freq         (342)
#define Pfn_set_texture_res_opt_heur          (343)
#define Pfn_set_cond_flags                    (344)
#define Pfn_set_cond_flags_from_tests         (345)
#define Pfn_set_line_rep_full                 (346)
#define Pfn_set_line_rep_mask                 (347)
#define Pfn_set_marker_rep_mask               (348)
#define Pfn_set_text_rep_mask                 (349)
#define Pfn_set_int_rep_mask                  (350)
#define Pfn_set_edge_rep_full                 (351)
#define Pfn_set_edge_rep_mask                 (352)
#define Pfn_set_ext_pat_rep                   (353)
#define Pfn_set_pat_rep_mask                  (354)
#define Pfn_set_texture_rep                   (355)
#define Pfn_set_texture_rep_mask              (356)
#define Pfn_set_texture_param                 (357)
```

```
#define Pfn_set_texture_composition              (358)
#define Pfn_set_texture_sampling                 (359)
#define Pfn_set_texture_binding                  (360)
#define Pfn_set_highl_rep                        (361)
#define Pfn_set_transparency_mode                (362)
#define Pfn_set_transparency_thresholds          (363)
#define Pfn_set_refl_rep_mask                    (364)
#define Pfn_set_appl_filter                      (365)
#define Pfn_create_mipmap_texture                (366)
#define Pfn_define_linetype                      (367)
#define Pfn_define_marker_type                   (368)
#define Pfn_set_view_ref_point3                  (369)
#define Pfn_set_view_ref_point                   (370)
#define Pfn_set_view_plane_norm                  (371)
#define Pfn_set_view_up_vec3                     (372)
#define Pfn_set_view_up_vec                      (373)
#define Pfn_set_view_win_limits                  (374)
#define Pfn_set_proj_vp3                         (375)
#define Pfn_set_proj_vp                          (376)
#define Pfn_set_proj_type                        (377)
#define Pfn_set_proj_ref_point                   (378)
#define Pfn_set_view_plane_dist                  (379)
#define Pfn_set_front_plane_dist                 (380)
#define Pfn_set_back_plane_dist                  (381)
#define Pfn_set_xy_clip_indicator                (382)
#define Pfn_set_front_clip_indicator             (383)
#define Pfn_set_back_clip_indicator              (384)
#define Pfn_upd_view_rep                         (385)
#define Pfn_map_dc_to_wsc                        (386)
#define Pfn_map_wsc_to_dc                        (387)
#define Pfn_map_dc_to_wc                         (388)
#define Pfn_open_di_struct                       (389)
#define Pfn_close_di_struct                      (390)
#define Pfn_inst_struct                          (391)
#define Pfn_cond_exec_struct                     (392)
#define Pfn_cond_inst_struct                     (393)
#define Pfn_cond_return                          (394)
#define Pfn_cond_skip_elements                   (395)
#define Pfn_cond_skip_to_label                   (396)
#define Pfn_push_st                              (397)
#define Pfn_pop_st                               (398)
#define Pfn_copy_elem_struct                     (399)
#define Pfn_copy_elem_range_struct               (400)
#define Pfn_copy_elems_between_labels_struct     (401)
#define Pfn_move_elem                            (402)
#define Pfn_move_elem_range                      (403)
#define Pfn_move_elems_between_labels            (404)
#define Pfn_set_watch_on_elem_range              (405)
#define Pfn_end_watch_on_elem_range              (406)
#define Pfn_post_struct_to_grp                   (407)
```

```
#define Pfn_unpost_structs_from_grps         (408)
#define Pfn_unpost_all_structs_from_grp      (409)
#define Pfn_post_di_struct                   (410)
#define Pfn_post_di_struct_to_grp            (411)
#define Pfn_unpost_di_struct                 (412)
#define Pfn_renew_di_state                   (413)
#define Pfn_set_di_mode                      (414)
#define Pfn_mark_multi_pass_start            (415)
#define Pfn_mark_multi_pass_compl            (416)
#define Pfn_mark_pass_start                  (417)
#define Pfn_mark_pass_compl                  (418)
#define Pfn_ret_num_passes_req               (419)
#define Pfn_set_set_pick_filter              (420)
#define Pfn_set_composite_pick_filter        (421)
#define Pfn_set_di_pick_filter               (422)
#define Pfn_set_di_pick_corr_point3          (423)
#define Pfn_set_di_pick_corr_point           (424)
#define Pfn_set_pick_mapping_data            (425)
#define Pfn_map_dc_point_to_pick_paths       (426)
#define Pfn_define_locator                   (427)
#define Pfn_define_stroke                    (428)
#define Pfn_define_valuator                  (429)
#define Pfn_define_choice                    (430)
#define Pfn_define_pick                      (431)
#define Pfn_define_string                    (432)
#define Pfn_create_set_measure               (433)
#define Pfn_define_set                       (434)
#define Pfn_create_composite_measure         (435)
#define Pfn_define_composite                 (436)
#define Pfn_undefine_locator                 (437)
#define Pfn_undefine_stroke                  (438)
#define Pfn_undefine_valuator                (439)
#define Pfn_undefine_choice                  (440)
#define Pfn_undefine_pick                    (441)
#define Pfn_undefine_string                  (442)
#define Pfn_destroy_set_measure              (443)
#define Pfn_undefine_set                     (444)
#define Pfn_destroy_composite_measure        (445)
#define Pfn_undefine_composite               (446)
#define Pfn_init_set3                        (447)
#define Pfn_init_set                         (448)
#define Pfn_init_composite3                  (449)
#define Pfn_init_composite                   (450)
#define Pfn_set_set_mode                     (451)
#define Pfn_set_composite_mode               (452)
#define Pfn_req_set3                         (453)
#define Pfn_req_set                          (454)
#define Pfn_req_composite3                   (455)
#define Pfn_req_composite                    (456)
#define Pfn_sample_set3                      (457)
```

```
#define Pfn_sample_set                              (458)
#define Pfn_sample_composite3                       (459)
#define Pfn_sample_composite                        (460)
#define Pfn_get_set3                                (461)
#define Pfn_get_set                                 (462)
#define Pfn_get_composite3                          (463)
#define Pfn_get_composite                           (464)
#define Pfn_attach_lid_to_view                      (465)
#define Pfn_detach_lid_from_view                    (466)
#define Pfn_attach_lid_to_light_src                 (467)
#define Pfn_detach_lid_from_light_src               (468)
#define Pfn_init_di_pick3                           (469)
#define Pfn_init_di_pick                            (470)
#define Pfn_enable_di_pick                          (471)
#define Pfn_disable_di_pick                         (472)
#define Pfn_trans_di_pick_set                       (473)
```
”

Pages 76

12.2 Error codes

```
/* State Errors */
#define PE_NOT_DISO                    (8)   /* Ignoring function, func-
                                                tion requires state
                                                (PHOP,*,DISO,*) */
#define PE_NOT_RETAINED_STOP_ONLY      (9)   /* Ignoring function, func-
                                                tion requires state
                                                (PHOP,*,STOP,*) */
#define PE_NOT_WSOP_AND_DISO           (10)  /* Ignoring function, func-
                                                tion requires state
                                                (PHOP,WSOP,DISO,*) */

/* Workstation Errors */
#define PE_BAD_TARG_OP                 (65)  /* Ignoring function, one of
                                                the target operations is
                                                not supported */
#define PE_INVALID_TRAV_RES            (66)  /* Ignoring function, the
                                                resource is not valid */
#define PE_TRAV_RES_ASSOC_TARG         (67)  /* Ignoring function, the
                                                traversal resource is
                                                already associated with a
                                                target */
#define PE_TRAV_RES_NO_DISASSOC        (68)  /* Ignoring function, the
                                                resource cannot be disas-
                                                sociated */
#define PE_INCOMPAT_TRAV_RES           (69)  /* Ignoring function, source
                                                and destination traversal
                                                resources are incompatible
                                                */
```

```
#define PE_DI_STRUCT_NOT_POSTED             (71)  /* Ignoring function, the
                                                     direct interpretation
                                                     structure is not posted on
                                                     the specified workstation
                                                     */
#define PE_DI_REND_TRAV_PROC_ACTIVE         (72)  /* Ignoring function, the
                                                     direct interpretation ren-
                                                     der traversal process is
                                                     already active on the
                                                     specified workstation */
#define PE_DI_REND_TRAV_PROC_NOT_ACTIVE
                                            (73)  /* Ignoring function, the
                                                     direct interpretation tra-
                                                     versal process is not
                                                     active on the specified
                                                     workstation */
#define PE_CANNOT_CREATE_TARGET             (74)  /* Ignoring function, an
                                                     additional target could
                                                     not be created */
#define PE_NUM_TARGETS_NOT_REDUCED          (75)  /* Ignoring function, the
                                                     number of targets cannot
                                                     be reduced below the
                                                     default number available
                                                     for the respective work-
                                                     station type */
#define PE_TARG_NOT_DESTROYED               (76)  /* Ignoring function, the
                                                     target could not be
                                                     destroyed */
#define PE_DISP_REND_TARG_NOT_DESTROYED     (77)  /* Ignoring function, nei-
                                                     ther the display target
                                                     nor the rendering target
                                                     can be destroyed */
#define PE_MULTIPASS_IN_PROGRESS            (78)  /* Ignoring function, a
                                                     multi-pass operation is
                                                     already in progress */
#define PE_MULTI_PASS_NOT_IN_PROGRESS       (79)  /* Ignoring function, a
                                                     multi-pass operation is
                                                     not in progress */
#define PE_PASS_DEF_IN_PROGRESS             (80)  /* Ignoring function, a pass
                                                     definition is already in
                                                     progress */
#define PE_PASS_DEF_NOT_IN_PROGRESS         (81)  /* Ignoring function, a pass
                                                     definition is not in
                                                     progress */
#define PE_LT_SRC_WS_NO_OUT_CAPABILITY      (82)  /* Ignoring function, the
                                                     specified light source
                                                     workstation does not have
                                                     output capability (i.e,
                                                     the workstation category
                                                     is neither OUTPUT, OUTIN,
                                                     nor MO) */
#define PE_POST_GRP_DEF                     (83)  /* Ignoring function, the
                                                     posting group has already
                                                     been defined */
```

```
#define PE_POST_GRP_NOT_DEF              (84)  /* Ignoring function, the
                                                  posting group has not been
                                                  defined */
#define PE_POST_GRP_ID_LESS_THAN_ONE     (85)  /* Ignoring function, the
                                                  posting group identifier
                                                  is less than one */
#define PE_MAX_POST_GRP_EXCEEDED         (86)  /* Ignoring function, defin-
                                                  ing this posting group
                                                  would exceed the maximum
                                                  number of posting groups
                                                  supported on the worksta-
                                                  tion */
#define PE_IMAGE_RES_NOT_DEF             (87)  /* Ignoring function, image
                                                  resource not defined */
#define PE_BAD_IMAGE_SPEC_METHOD         (88)  /* Ignoring function, the
                                                  image resource image spec-
                                                  ification method is not
                                                  supported for this opera-
                                                  tion */
#define PE_IMAGE_RES_CAP_NOT_SUP         (89)  /* Ignoring function, image
                                                  resource capability is not
                                                  supported on the worksta-
                                                  tion */
#define PE_SPEC_IMAGE_RES_NOT_DEF        (90)  /* Ignoring function, the
                                                  specified image resource
                                                  has not been defined */
#define PE_MAX_IMAGE_RES_EXCEEDED        (91)  /* Ignoring function, associ-
                                                  ating this image resource
                                                  would exceed the maximum
                                                  number of entries allowed
                                                  in the workstation image
                                                  resource table */
#define PE_IMAGE_RES_EXCEEDS_CAP_WS      (92)  /* Ignoring function, the
                                                  image resource exceeds the
                                                  capabilities of the speci-
                                                  fied workstation */
#define PE_VW_WS_OF_CATEGORY_MI          (93)  /* Ignoring function, the
                                                  specified view worksta-
                                                  tion is of category MI */
#define PE_LID_WS_NOT_IN_NOR_OUTIN       (94)  /* Ignoring function, the
                                                  specified LID workstation
                                                  is neither of category
                                                  INPUT nor of category
                                                  OUTIN */
#define PE_NEW_WS_TYPE_NOT_REALIZED      (95)  /* Warning, Not all
                                                  attributes were fully
                                                  realized during the cre-
                                                  ation of the new worksta-
                                                  tion type */
#define PE_WS_TYPE_CANNOT_BE_MODIFIED    (96)  /* Ignoring function, work-
                                                  station type is a default
                                                  type or bound to a work-
                                                  station and cannot be mod-
                                                  ified */
```

```
/* Output Attribute Errors */
#define PE_NO_STATE_SAVED                   (141) /* Ignoring function, no state
                                                     has been saved */
#define PE_SPEC_PAT_TYPE_NOT_SUP            (142) /* Ignoring function, the
                                                     specified pattern type is
                                                     not supported on the spec-
                                                     ified workstation */
#define PE_PAT_TYPE_NOT_SUP_ON_INQ          (143  /* Ignoring function, pattern
                                                     type of requested pattern
                                                     table entry is not sup-
                                                     ported by this inquiry */
#define PE_MAX_VIEWS_EXCEEDED               (150) /* Ignoring function, setting
                                                     this view table entry
                                                     would exceed the maximum
                                                     number of entries allowed
                                                     */
#define PE_INVALID_LIMITS                   (151) /* Ignoring function, XMIN ≥
                                                     XMAX, YMIN ≥ YMAX, ZMIN >
                                                     ZMAX, UMIN ≥ UMAX or VMIN ≥
                                                     VMAX */
#define PE_INVALID_VIEWPORT                 (152) /* Ignoring function, invalid
                                                     viewport: XMIN ≥ XMAX, YMIN
                                                     ≥ YMAX, ZMIN > ZMAX */
#define PE_INVALID_VIEW_CLIP_LIM            (153) /* Ignoring function, invalid
                                                     view clipping limits: XMIN
                                                     ≥ XMAX, YMIN ≥ YMAX, ZMIN >
                                                     ZMAX */
#define PE_VIEW_CLIP_LIM_NOT_NPC            (154) /* Ignoring function, the view
                                                     clipping limits are not
                                                     within NPC range */
#define PE_PROJ_VP_LIMITS_NOT_NPC           (155) /* Ignoring function, the pro-
                                                     jection viewport limits
                                                     are not within NPC range
                                                     */
#define PE_WS_WIN_LIM_NOT_NPC               (156) /* Ignoring function, the
                                                     workstation window limits
                                                     are not within NPC range
                                                     */
#define PE_WS_VP_NOT_IN_DISPLAY_SPACE       (157) /* Ignoring function, the
                                                     workstation viewport is
                                                     not within display space
                                                     */
#define PE_INVALID_VIEW_PLANES              (158) /* Ignoring function, front
                                                     plane and back plane dis-
                                                     tances same and z-extent
                                                     of projection viewport
                                                     non-zero */
#define PE_VPN_LEN_ZERO                     (159) /* Ignoring function, the view
                                                     plane normal has length
                                                     zero */
#define PE_VUP_LEN_ZERO                     (160) /* Ignoring function, the view
                                                     up vector has lenght zero
                                                     */
```

```
#define PE_VUP_VPN_PARALLEL                 (161) /* Ignoring function, view up
                                                     and view plane normal vec-
                                                     tors parallel */
#define PE_PRP_BETW_FRONT_BACK              (162) /* Ignoring function, the
                                                     projection reference point
                                                     is between the front and
                                                     back planes */
#define PE_PRP_ON_VIEW_PLANE                (163) /* Ignoring function, the
                                                     projection referece point
                                                     cannot be positioned on
                                                     the view plane */
#define PE_BACK_PLN_BEFORE_FRONT_PLN        (164) /* Ignoring function, the
                                                     back plane is in front of
                                                     the front plane */
#define PE_HIGHL_IND_LT_ONE                 (165) /* Ignoring function, the
                                                     highlighting index is less
                                                     than one */
#define PE_HIGHL_METHOD_NOT_AVAIL           (166) /* Ignoring function, the
                                                     specified highlighting
                                                     method is not available on
                                                     the specified workstation
                                                     */
#define PE_LINETYPE_NOT_DEFINABLE           (167) /* Ignoring function, the
                                                     specified linetype does
                                                     not reference a definable
                                                     linetype */
#define PE_MAX_DEF_LINETYPES_EXCEEDED       (168) /* Ignoring function, set-
                                                     ting this table entry
                                                     would exceed the maximum
                                                     number of entries allowed
                                                     in the definable linetype
                                                     table */
#define PE_LINETYPE_NOT_DEFINED             (169) /* Ignoring function, the
                                                     specified linetype has not
                                                     been defined */
#define PE_LINETYPE_NOT_SUP                 (170) /* Warning, this linetype
                                                     definition is not sup-
                                                     ported on at least one of
                                                     the available workstation
                                                     types */
#define PE_LINETYPE_DEF_NOT_AVAIL           (171) /* Ignoring function, the
                                                     specified standard  or
                                                     implementation-defined
                                                     linetype definition is not
                                                     available */
#define PE_MARKER_TYPE_NOT_DEFINABLE        (172) /* Ignoring function, the
                                                     specified marker type does
                                                     not reference a definable
                                                     marker type */
#define PE_MAX_DEF_MARKER_TYPES_EXCEEDED    (173) /* Ignoring function, set-
                                                     ting this table entry
                                                     would exceed the maximum
                                                     number of entries allowed
```

```
                                                 in the definable marker
                                                 type table */
#define PE_MARKER_TYPE_NOT_DEFINED       (174) /* Ignoring function, the
                                                 specified marker type has
                                                 not been defined */
#define PE_MARKER_SHAPE_INVALID          (175) /* Ignoring function, the
                                                 marker shape descriptor is
                                                 invalid */
#define PE_MARKER_SHAPE_TYPE_NOT_SUP     (176) /* Ignoring function, the
                                                 specified marker shape
                                                 type is not supported */
#define PE_MARKER_TYPE_DEF_NOT_AVAIL     (177) /* Ignoring function, the
                                                 specified standard or
                                                 implementation defined
                                                 marker type definition is
                                                 not available */
#define PE_FILTER_SEL_INVALID            (178) /* Ignoring function, the
                                                 filter selection is
                                                 invalid */
#define PE_FILTER_NOT_DEFINED            (179) /* Ignoring function, the
                                                 specified application fil-
                                                 ter has not been defined
                                                 */

/* Input Errors */
#define PE_DI_PICK_TRAV_PROC_ACTIVE      (264) /* Ignoring function, the
                                                 direct interpretation pick
                                                 traversal process is
                                                 already active on the
                                                 specified workstation */
#define PE_DI_PICK_TRAV_PROC_NOT_ACTIVE  (265) /* Ignoring function, the
                                                 direct interpretation pick
                                                 traversal process is not
                                                 active on the specified
                                                 workstation */
#define PE_BAD_PICK_TYPE                 (266) /* Ignoring function, the
                                                 specified pick type is not
                                                 available on the speci-
                                                 fied workstation */
#define PE_LISTED_MEAS_PROC_INVALID      (267) /* Ignoring function, one of
                                                 the listed measure pro-
                                                 cesses is invalid */
#define PE_LISTED_TRIG_PROC_INVALID      (268) /* Ignoring function, one of
                                                 the listed trigger pro-
                                                 cesses is invalid */
#define PE_LISTED_ECHO_PROC_INVALID      (269) /* Ignoring function, one of
                                                 the listed echo processes
                                                 is invalid */
#define PE_LISTED_ACK_PROC_INVALID       (270) /* Ignoring function, one of
                                                 the listed acknowledge-
                                                 ment processes is invalid
                                                 */
#define PE_LID_NR_IN_USE                 (271) /* Ignoring function, device
                                                 number already in use */
```

```
#define PE_NR_MEAS_INVALID                (272) /* Ignoring function, number
                                                   of measures is invalid */
#define PE_MEAS_ID_NOT_AVAIL              (273) /* Ignoring function, the
                                                   specified measure identi-
                                                   fier is not available on
                                                   the specified workstation
                                                   */
#define PE_MEAS_ID_IN_USE                 (274) /* Ignoring function, the
                                                   specified measure identi-
                                                   fier is currently part of
                                                   a set or composite logi-
                                                   cal input device defini-
                                                   tion */
#define PE_NO_PICK_MEASURE                (275) /* Ignoring function, the
                                                   specified set or compos-
                                                   ite device does not have a
                                                   pick measure */
#define PE_LID_INAPPROP_FOR_VW_ACTION     (276) /* Ignoring function, the
                                                   logical input device class
                                                   specified is inappropri-
                                                   ate for the view action
                                                   specified */
#define PE_LID_INAPPROP_FOR_LT_ACT        (277) /* Ignoring function, the
                                                   input device class speci-
                                                   fied is inappropriate for
                                                   the light action speci-
                                                   fied */
#define PE_LT_ACT_INAPPROP_FOR_LT_SRC     (278) /* Ignoring function, the
                                                   light action is inappro-
                                                   priate for the light
                                                   source specified */

/* Miscellaneous Errors */
#define PE_TEST_OP_INVALID                (451) /* Ignoring function, invalid
                                                   test operator for test type
                                                   */
#define PE_NO_WINDOW_SYSTEM               (452) /* Ignoring function, the
                                                   specified workstation is
                                                   not operating in a window
                                                   management system envrion-
                                                   ment */
#define PE_WIN_SYS_COLR_NOT_DETER         (453) /* Ignoring function, the
                                                   window system colour can-
                                                   not be determined */

/* Additional Output Attribute Errors */
#define PE_TEXTR_COORD_SRC_NOT_AVAIL      (600) /* Ignoring function, the
                                                   specified coordinate
                                                   source is not available on
                                                   the specified workstation
                                                   */
#define PE_DATA_VALUE_IND_LESS_1          (601) /* Ignoring function, the
                                                   specified data value index
                                                   is less than one */
```

```
#define PE_TEXTR_COMPOS_METHOD_NOT_AVAIL
                                          (602) /* Ignoring function, the
                                                   specified composition
                                                   method is not available on
                                                   the specified workstation
                                                   */
#define PE_TEXTR_MIN_METHOD_NOT_AVAIL     (603) /* Ignoring function, the
                                                   specified minification
                                                   method is not available on
                                                   the specified workstation
                                                   */
#define PE_TEXTR_MAG_METHOD_NOT_AVAIL     (604) /* Ignoring function, the
                                                   specified magnification
                                                   method is not available on
                                                   the specified workstation
                                                   */
#define PE_BOUND_COND_NOT_AVAIL           (605) /* Ignoring function, the
                                                   specified boundary condi-
                                                   tion is not available on
                                                   the specified workstation
                                                   */
#define PE_BOUND_CLAMP_METHOD_NOT_AVAIL
                                          (606) /* Ignoring function, the
                                                   specified boundary clamp
                                                   method is not available on
                                                   the specified workstation
                                                   */
#define PE_DEPTH_SAMPL_BIAS_OUT_OF_RANGE
                                          (607) /* Ignoring function, the
                                                   specified depth sampling
                                                   bias hint is out of range
                                                   */
#define PE_TEXTR_IMAGE_RES_NOT_DEFINED    (608) /* Ignoring function, the
                                                   specified texture image
                                                   resource identifier is not
                                                   defined on the specified
                                                   workstation */
#define PE_RENDERING_ORDER_NOT_AVAIL      (612) /* Ignoring function, the
                                                   specified rendering order
                                                   is not available on the
                                                   specified workstation */
```

12.3 Miscellaneous

12.3.1 Hatch styles

```
#define PHATCH_HORIZ                          (1)
#define PHATCH_VERT                           (2)
#define PHATCH_POS_SLOPE                      (3)
#define PHATCH_NEG_SLOPE                      (4)
#define PHATCH_HORIZ_VERT_CROSS               (5)
#define PHATCH_POS_NEG_SLOPE_CROSS            (6)
```

12.3.2 Prompt and echo types (additional items)

```
#define PPICK_HIGHL_PRIM                    (4)
#define PPICK_HIGHL_PRIM_GROUP              (5)
#define PPICK_HIGHL_STRUCT_NETWORK          (6)
```

12.3.3 Colour types (additional items)

```
#define PCOLR_RGBA                          (5)
#define PCOLR_CIELUVA                       (6)
#define PCOLR_HSVA                          (7)
#define PCOLR_HLSA                          (8)
```

12.3.4 Direct interpretation pick prompt and echo types

```
#define PDI_PICK_HIGHL                      (1)
```

12.3.5 Pick types

```
#define PPICK_TYPE_WS_DEP                   (1)
#define PPICK_TYPE_PICK_FIRST               (2)
#define PPICK_TYPE_PICK_LAST                (3)
#define PPICK_TYPE_PICK_CLOSEST             (4)
```

12.3.6 Target manipulation modes

```
#define PTARG_MANIP_AUTO                    (1)
#define PTARG_MANIP_MANUAL                  (2)
```

12.3.7 Target operation types

```
#define PSET_REND_TARG                      (1)
#define PSET_DISP_TARG                      (2)
#define PCLEAR_TARG                         (3)
#define PCOPY_TARG                          (4)
```

12.3.8 Traversal resource operation types

```
#define PCLEAR_TRAV_RES                     (0)
#define PCOPY_TRAV_RES                      (1)
```

12.3.9 Traversal resource types

```
#define PTRAV_RES_Z_BUF                     (1)
#define PTRAV_RES_PAINT_BUF                 (2)
```

12.3.10 View update method

```
#define PVUM_APPLY_ALL                      (1)
#define PVUM_APPLY_ROTATION                 (2)
```

12.3.11 View action types

```
#define PVWA_ROT_ABOUT_U_VRC_AXIS                  (1)
#define PVWA_ROT_ABOUT_V_VRC_AXIS                  (2)
#define PVWA_ROT_ABOUT_N_VRC_AXIS                  (3)
#define PVWA_TRANS_U_VRC_AXIS_CLIP                 (4)
#define PVWA_TRANS_U_VRC_AXIS_NOCLIP               (5)
#define PVWA_TRANS_V_VRC_AXIS_CLIP                 (6)
#define PVWA_TRANS_V_VRC_AXIS_NOCLIP               (7)
#define PVWA_TRANS_N_VRC_AXIS                      (8)
#define PVWA_SCALE_WIN_UNIFORMLY                   (9)
#define PVWA_SCALE_VP_UNIFORMLY                    (10)
#define PVWA_SCALE_WIN_WIDTH                       (11)
#define PVWA_SCALE_VP_WIDTH                        (12)
#define PVWA_SCALE_WIN_HEIGHT                      (13)
#define PVWA_SCALE_VP_HEIGHT                       (14)
#define PVWA_TRANS_FRONT_CLIP_PLANE                (15)
#define PVWA_TRANS_BACK_CLIP_PLANE                 (16)
#define PVWA_ROT_AROUND_VRP_ALONG_CYL_LAT          (17)
#define PVWA_TRANS_EYE_FROM_VRP_OUT                (18)
#define PVWA_TRANS_EYE_FROM_VRP_UP                 (19)
#define PVWA_ROT_EYE_FROM_VRP_UP                   (20)
#define PVWA_CHG_EYE_POS_WRT_VRP                   (21)
#define PVWA_CHG_VRP_POS_ROT_UP_WRT_PRP            (22)
#define PVWA_CHG_VRP_POS_ROT_RL_WRT_PRP            (23)
```

12.3.12 Measure transfer types

```
#define PMTT_MEASURE_CHANGE                   (1)
#define PMTT_EVENT_WITH_DISCARD               (2)
#define PMTT_EVENT_WITH_PROPAGATION           (3)
```

12.3.13 Background methods

```
#define PBACKGM_COLOUR_TABLE_0                (1)
#define PBACKGM_COLOUR_INDEX                  (2)
#define PBACKGM_COLOUR                        (3)
#define PBACKGM_IMAGE_RESOURCE                (4)
```

12.3.14 Image specification methods

```
#define PIMGSM_UNCOMP_COLR_INDEX_ARRAY        (1)
#define PIMGSM_UNCOMP_COLR_ARRAY              (2)
#define PIMGSM_WINDOW_SYS_BITMAP              (3)
#define PIMGSM_VIDEO_SIGNAL_CHAN_ID           (4)
#define PIMGSM_LUMINANCE                      (5)
#define PIMGSM_LUMINANCE_ALPHA                (6)
#define PIMGSM_MIPMAP_COLRS                   (7)
#define PIMGSM_MIPMAP_LUMINANCE               (8)
#define PIMGSM_MIPMAP_LUMINANCE_ALPHA         (9)
```

12.3.15 Pattern types

```
#define PPATT_COLOUR_INDEX                             (1)
#define PPATT_COLOUR                                   (2)
#define PPATT_IMAGE_RESOURCE                           (3)
```

12.3.16 Marker Shape Types

```
#define PSHAPE_POLYLINE                                (1)
#define PSHAPE_FILL_AREA                               (2)
#define PSHAPE_CONVEX_FILL_AREA                        (3)
```

12.3.17 Test Methods

```
#define PTEST_ALWAYS_FAIL                              (0)
#define PTEST_ALWAYS_SUCCEED                           (1)
#define PTEST_EXTENT3                                  (2)
#define PTEST_EXTENT                                   (3)
#define PTEST_BOUNDS3                                  (4)
#define PTEST_BOUNDS                                   (5)
#define PTEST_NAMESET                                  (6)
#define PTEST_APPL_INTEGER                             (7)
#define PTEST_APPL_REAL                                (8)
```

12.3.18 Filter Types

```
#define PFILTER_TYPE_INVIS                             (1)
#define PFILTER_TYPE_HIGHL                             (2)
#define PFILTER_TYPE_PICK                              (3)
#define PFILTER_TYPE_APPL                              (4)
```

12.3.12 Texture sampling frequency types

```
#define PTEXTRSF_PER_PIXEL                             (1)
#define PTEXTRSF_INTERP_DEP                            (2)
```

12.3.13 Texture optimization hints

```
#define PTEXTR_OPT_NO_HINT                             (1)
#define PTEXTR_OPT_SPEED                               (2)
#define PTEXTR_OPT_SPACE                               (3)
```

12.3.14 Alpha sources

```
#define PALPHASRC_TRANSPARENCY                         (1)
#define PALPHASRC_COLR_ASPECT                          (2)
#define PALPHASRC_FACET_COLR                           (3)
#define PALPHASRC_FACET_DATA                           (4)
#define PALPHASRC_VERTEX_COLR                          (5)
#define PALPHASRC_VERTEX_DATA                          (6)
```

12.3.15 Texture coordinate sources

```
#define PCOORDSRC_VERTEX_COORD_MC           (1)
#define PCOORDSRC_VERTEX_COORD_WC           (2)
#define PCOORDSRC_VERTEX_NORMAL_MC          (3)
#define PCOORDSRC_VERTEX_NORMAL_WC          (4)
#define PCOORDSRC_NURB_SURF_UV              (5)
#define PCOORDSRC_DATA                      (6)
#define PCOORDSRC_REFL_SPHERE_VRC           (7)
#define PCOORDSRC_REFL_SPHERE_WC            (8)
```

12.3.16 Gamma channel selectors

```
#define PGCS_GAMMA_ONE                      (1)
#define PGCS_GAMMA_RED                      (1)
#define PGCS_GAMMA_TWO                      (2)
#define PGCS_GAMMA_GREEN                    (2)
#define PGCS_GAMMA_THREE                    (3)
#define PGCS_GAMMA_BLUE                     (3)
```

12.3.17 Texture composition methods

```
#define PCOMPOS_REPLACE                     (1)
#define PCOMPOS_MODULATE                    (2)
#define PCOMPOS_BLEND_ENVIRONMENT_COLR      (3)
#define PCOMPOS_DECAL                       (4)
#define PCOMPOS_DECAL_BACKGROUND            (5)
```

12.3.18 Minification methods

```
#define PMINM_SINGLE_BASE                   (1)
#define PMINM_LINEAR_BASE                   (2)
#define PMINM_SINGLE_IN_MIPMAP              (3)
#define PMINM_LINEAR_IN_MIPMAP              (4)
#define PMINM_SINGLE_BETWEEN_MIPMAPS        (5)
#define PMINM_LINEAR_BETWEEN_MIPMAPS        (6)
```

12.3.19 Magnificiation methods

```
#define PMAGM_SINGLE_BASE                   (1)
#define PMAGM_LINEAR_BASE                   (2)
#define PMAGM_SINGLE_IN_MIPMAP              (3)
#define PMAGM_LINEAR_IN_MIPMAP              (4)
#define PMAGM_SINGLE_BETWEEN_MIPMAPS        (5)
#define PMAGM_LINEAR_BETWEEN_MIPMAPS        (6)
```

12.3.20 Boundary conditions

```
#define PBOUND_COND_CLAMP_EXPLICIT_METHOD          (1)
#define PBOUND_COND_CLAMP_BORDER                   (2)
#define PBOUND_COND_WRAP                           (3)
#define PBOUND_COND_MIRROR                         (4)
```

12.3.21 Boundary clamp methods

```
#define PBOUND_CLAMP_DISCONTINUE              (1)
#define PBOUND_CLAMP_COLR                     (2)
```

12.3.22 Rendering orders

```
#define PRENDORDR_PRELIGHT                    (1)
#define PRENDORDR_POST_LIGHT                  (2)
```

12.3.23 Transparency modes

```
#define PTRANSPMODE_NONE                      (1)
#define PTRANSPMODE_SCREEN_DOOR               (2)
#define PTRANSPMODE_ALPHA_BLEND               (3)
#define PTRANSPMODE_TWO_PASS                  (4)
#define PTRANSPMODE_MULTIPASS                 (5)
```

12.3.24 Light source action types

```
#define PLTA_ADJ_LT_COLR_COMP_1               (1)
#define PLTA_ADJ_LT_COLR_COMP_2               (2)
#define PLTA_ADJ_LT_COLR_COMP_3               (3)
#define PLTA_ADJ_BRIGHT                       (4)
#define PLTA_ROT_AROUND_X_DIR_VEC             (5)
#define PLTA_ROT_AROUND_Y_DIR_VEC             (6)
#define PLTA_ROT_AROUND_Z_DIR_VEC             (7)
#define PLTA_TRANS_X_LT_POS                   (8)
#define PLTA_TRANS_Y_LT_POS                   (9)
#define PLTA_TRANS_Z_LT_POS                   (10)
#define PLTA_ADJ_CONCENTRATION_EXP            (11)
#define PLTA_ADJ_LT_SPREAD_ANGLE              (12)
```

13 C *Full PHIGS profile* functions

13.1 Control functions

WORKSTATION TYPE CREATE

```
void pws_type_create (

   Pint                      ws_type,        /* workstation type         */
   const Pconfig_setting     *settings_list,/* array of config. settings*/
   Pint                      *new_ws_type    /* OUT new workstation type */

);
```

WORKSTATION TYPE SET

```
void pws_type_set (

   Pint                      ws_type,        /* workstation type         */
   const Pconfig_setting     *settings_list /* array of config. settings*/

);
```

WORKSTATION TYPE GET

```
void pws_type_get (

   Pint           ws_type,          /* workstation type                 */
   const char     *setting_name,    /* configuration setting name       */
   Pstore         store,            /* store object for returned value */
   Pconfig_value  **setting_value   /* OUT configuration setting value */

);
```

WORKSTATION TYPE DESTROY

```
void pws_type_destroy (

   Pint         ws_type                    /* workstation type         */

);
```

REDRAW ALL STRUCTURES FROM POSTING GROUP

```
void predraw_all_structs_from_grp (

   Pint                ws_id,            /* workstation identifier      */
   Pint                grp_id,           /* posting group               */
   Pbackg_redisplay    redisplay_flag    /* background redisplay flag   */

);
```

REDRAW ALL STRUCTURES ON TARGET

```
void predraw_all_structs_on_targ (

   Pint                ws_id,            /* workstation identifier      */
   const Ptarg_addr    *targ_addr,       /* target address              */
   Pctrl_flag          ctrl_flag         /* control flag                */

);
```

REDRAW ALL STRUCTURES FROM POSTING GROUP ON TARGET

```
void predraw_all_from_grp_on_targ (

   Pint                ws_id,            /* workstation identifier      */
   Pint                grp_id,           /* posting group               */
   Pbackg_redisplay    redisplay_flag    /* background redisplay flag   */
   const Ptarg_addr    *targ_addr,       /* target address              */
   Pctrl_flag          ctrl_flag         /* control flag                */

);
```

UPDATE TARGET

```
void pupd_targ (

   Pint                ws_id,            /* workstation identifier      */
   const Ptarg_addr    *targ_addr,       /* target address              */
   Pregen_flag         regen_flag        /* update regeneration flag    */

);
```

DEFINE POSTING GROUP

```
void pdefine_post_grp (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 grp_id,         /* posting group identifier    */
   Pint                 ref_grp_id,     /* reference posting group id  */
   Prel_pri             rel_pri,        /* relative priority           */
   Pint                 def_view_ind,   /* default view index          */
   Ppost_grp_status     post_grp_status,/* posting group status        */
   Pbackg_style         backg_style,    /* posting group backg. style  */
   const Pbackg_method*backg_method,    /* posting group backg. method */
   Pborder_indic        border_indic,   /* posting group border indic. */
   Pint                 border_ind      /* posting group border index  */

);
```

UNDEFINE POSTING GROUP

```
void pundefine_post_grp (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 grp_id          /* posting group identifier    */

);
```

SET POSTING GROUP STATUS

```
void pset_post_grp_status (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 grp_id,         /* posting group identifier    */
   Ppost_grp_status     post_grp_status /* posting group status        */

);
```

SET POSTING GROUP PRIORITY

```
void pset_post_grp_priority (

   Pint        ws_id,                   /* workstation identifier      */
   Pint        grp_id,                  /* posting group identifier    */
   Pint        ref_grp_id,              /* reference posting group id  */
   Prel_pri    rel_pri                  /* relative priority           */

);
```

SET POSTING GROUP BACKGROUND STYLE

```
void pset_post_grp_backg_style (
   Pint            ws_id,          /* workstation identifier         */
   Pint            grp_id,         /* posting group identifier       */
   Pbackg_style    backg_style     /* posting group background style */
);
```

SET POSTING GROUP BACKGROUND METHOD

```
void pset_post_grp_backg_method (
   Pint                  ws_id,          /* workstation identifier     */
   Pint                  grp_id,         /* posting group identifier   */
   const Pbackg_method   *backg_method   /* posting group background   */
                                         /* method                     */
);
```

SET POSTING GROUP BORDER INDICATOR

```
void pset_post_grp_border_indicator (
   Pint           ws_id,           /* workstation identifier         */
   Pint           grp_id,          /* posting group identifier       */
   Pborder_indic  border_indic     /* posting group border indicator */
);
```

SET POSTING GROUP BORDER INDEX

```
void pset_post_grp_border_ind (
   Pint       ws_id,              /* workstation identifier          */
   Pint       grp_id,             /* posting group identifier        */
   Pint       border_ind          /* posting group border index      */
);
```

ASSOCIATE IMAGE RESOURCE

```
void passoc_image_res (
   Pint                ws_id,          /* workstation identifier      */
   Pint                image_res_id,   /* image resource identifier   */
   const Pint_size     *dims,          /* m,n dimensions              */
   const Pimage_res    *image_res,     /* image resource              */
   Paccess_flag        *access_flag    /* OUT access flag             */
);
```

DISASSOCIATE IMAGE RESOURCE

```
void pdisassoc_image_res (
   Pint                ws_id,          /* workstation identifier      */
   Pint                image_res_id    /* image resource identifier   */
);
```

SET DEVICE COORDINATE CLIP REGIONS 3

```
void pset_dc_clip_regions3 (
   Pint                  ws_id,        /* workstation identifier      */
   const Plimit3_list    *limit_list   /* list of DC clip regions     */
);
```

SET DEVICE COORDINATE CLIP REGIONS

```
void pset_dc_clip_regions (
   Pint                ws_id,          /* workstation identifier      */
   const Plimit_list   *limit_list     /* list of DC clip regions     */
);
```

SET TARGET MANIPULATION MODE

```
void pset_targ_manip_mode (
   Pint              ws_id,           /* workstation identifier      */
   Pint              targ_manip_mode  /* target manipulation mode    */
);
```

SET TARGET DISPOSITION

```
void pset_targ_dispos (
  Pint                  ws_id,        /* workstation identifier      */
  const Ptarg_op_list   *targ_ops     /* target operations list      */
);
```

SET DISPLAY TARGET

```
void pset_disp_targ (
   Pint               ws_id,          /* workstation identifier      */
   const Ptarg_addr   *targ_addr      /* target address              */
);
```

SET RENDERING TARGET

```
void pset_rend_targ (
   Pint               ws_id,          /* workstation identifier      */
   const Ptarg_addr   *targ_addr      /* target address              */
);
```

CLEAR TARGET

```
void pclear_targ (
   Pint               ws_id,          /* workstation identifier      */
   const Ptarg_addr   *targ_addr      /* target address              */
);
```

COPY TARGET

```
void pcopy_targ (

   Pint                ws_id,            /* workstation identifier      */
   const Ptarg_addr    *src_targ_addr,   /* source target address       */
   const Ptarg_addr    *dest_targ_addr   /* destination target address  */

);
```

CREATE TARGET

```
void pcreate_targ (

   Pint                ws_id,            /* workstation identifier      */
   const Ptarg_addr    *targ_addr        /* target address              */

);
```

DESTROY TARGET

```
void pdestroy_targ (

   Pint                ws_id,            /* workstation identifier      */
   const Ptarg_addr    *targ_addr        /* target address              */

);
```

SET STATE OF VISUAL REPRESENTATION

```
void pset_st_visual_rep (

   Pint             ws_id,         /* workstation identifier          */
   Pvisual_st       visual_st      /* state of visual representation  */

);
```

SET TARGET STATE OF VISUAL REPRESENTATION

```
void pset_targ_st_visual_rep (

   Pint                ws_id,         /* workstation identifier          */
   const Ptarg_addr    *targ_addr,    /* target address                  */
   Pvisual_st          visual_st      /* state of visual representation */

);
```

ASSOCIATE TRAVERSAL RESOURCE

```
void passoc_trav_res (

   Pint                ws_id,        /* workstation identifier          */
   Pint                res_id,       /* traversal resource identifier   */
   const Ptarg_addr    *targ_addr    /* source target address           */

);
```

DISASSOCIATE TRAVERSAL RESOURCE

```
void pdisassoc_trav_res (

   Pint                ws_id,        /* workstation identifier          */
   Pint                res_id        /* traversal resource identifier   */

);
```

MANIPULATE TRAVERSAL RESOURCE

```
void pmanip_trav_res (

  Pint                   ws_id,        /* workstation identifier        */
  Pint                   res_id,       /* traversal resource identifier*/
  const Ptrav_res_op     *manip_data   /* traversal resource operation */

);
```

RESET ALL TRAVERSAL RESOURCES

```
void preset_all_trav_res (

  Pint                   ws_id           /* workstation identifier      */

);
```

RETRIEVE WINDOW SYSTEM COLOUR

```
void pret_window_system_colr (
 Pint            ws_id,          /* workstation identifier          */
 const Pgcolr    *colr,          /* colour sought                   */
 Pstore          store,          /* store object for returned value */
 Pdata           **winsys_colr   /* OUT window system colour        */
);
```

SET APPLICATION INTEGER

```
void pset_appl_int (
   Pint     integer_id,    /* application integer identifier     */
   Pint     value          /* value of application integer       */
);
```

SET APPLICATION REAL

```
void pset_appl_real (
   Pint     real_id,       /* application real identifier        */
   Pfloat   value          /* value of application real          */
);
```

13.2 Output primitive functions

CIRCLE 3

```
void pcircle3 (
   const Ppoint3      *ctr_point,     /* centre point             */
   Pfloat             radius,         /* radius                   */
   const Pvec3        ref_vecs[2]     /* reference vectors        */
);
```

CIRCLE

```
void pcircle (
   const Ppoint         *ctr_point,      /* centre point                      */
   Pfloat               radius           /* radius                            */
);
```

CIRCULAR ARC 3

```
void pcircular_arc3 (
   const Ppoint3        *ctr_point,      /* centre point                      */
   Pfloat               radius,          /* radius                            */
   const Pvec3          ref_vecs[2],     /* reference vectors                 */
   Pfloat               start,           /* start angle                       */
   Pfloat               end              /* end angle                         */
);
```

CIRCULAR ARC

```
void pcircular_arc (
   const Ppoint         *ctr_point,      /* centre point                      */
   Pfloat               radius,          /* radius                            */
   Pfloat               start,           /* start angle                       */
   Pfloat               end              /* end angle                         */
);
```

ELLIPSE 3

```
void pellipse3 (
   const Ppoint3     *ctr_point,         /* centre point                      */
   const Pvec3       *major_ref_vec,     /* major axis reference vector */
   const Pvec3       *minor_ref_vec      /* minor axis reference vector */
);
```

ELLIPSE

```
void pellipse (
   const Ppoint     *ctr_point,          /* centre point                */
   const Pvec       *major_ref_vec,      /* major axis reference vector */
   const Pvec       *minor_ref_vec       /* minor axis reference vector */
);
```

ELLIPTICAL ARC 3

```
void pelliptical_arc3 (
   const Ppoint3    *ctr_point,          /* centre point                */
   const Pvec3      *major_ref_vec,      /* major axis reference vector */
   const Pvec3      *minor_ref_vec,      /* minor axis reference vector */
   Pfloat           start,               /* start angle                 */
   Pfloat           end                  /* end angle                   */
);
```

ELLIPTICAL ARC

```
void pelliptical_arc (
   const Ppoint     *ctr_point,          /* centre point                */
   const Pvec       *major_ref_vec,      /* major axis reference vector */
   const Pvec       *minor_ref_vec,      /* minor axis reference vector */
   Pfloat           start,               /* start angle                 */
   Pfloat           end                  /* end angle                   */
);
```

FILL CIRCLE 3

```
void pfill_circle3 (
   const Ppoint3       *ctr_point,       /* centre point                */
   Pfloat              radius,           /* radius                      */
   const Pvec3         ref_vecs[2]       /* reference vectors           */
);
```

FILL CIRCLE

```
void pfill_circle (

   const Ppoint        *ctr_point,      /* centre point                 */
   Pfloat              radius           /* radius                       */

);
```

CIRCULAR ARC CLOSE 3

```
void pcircular_arc_close3 (

   const Ppoint3       *ctr_point,      /* centre point                 */
   Pfloat              radius,          /* radius                       */
   const Pvec3         ref_vecs[2],     /* reference vectors            */
   Pfloat              start,           /* start angle                  */
   Pfloat              end,             /* end angle                    */
   Pclosure            type             /* closure type                 */

);
```

CIRCULAR ARC CLOSE

```
void pcircular_arc_close (

   const Ppoint        *ctr_point,      /* centre point                 */
   Pfloat              radius,          /* radius                       */
   Pfloat              start,           /* start angle                  */
   Pfloat              end,             /* end angle                    */
   Pclosure            type             /* closure type                 */

);
```

FILL ELLIPSE 3

```
void pfill_ellipse3 (

   const Ppoint3     *ctr_point,        /* centre point                 */
   const Pvec3       *major_ref_vec,    /* major axis reference vector */
   const Pvec3       *minor_ref_vec     /* minor axis reference vector */

);
```

FILL ELLIPSE

```
void pfill_ellipse (

   const Ppoint     *ctr_point,         /* centre point                */
   const Pvec       *major_ref_vec,     /* major axis reference vector */
   const Pvec       *minor_ref_vec      /* minor axis reference vector */

);
```

ELLIPTICAL ARC CLOSE 3

```
void pelliptical_arc_close3 (

   const Ppoint3    *ctr_point,         /* centre point                */
   const Pvec3      *major_ref_vec,     /* major axis reference vector */
   const Pvec3      *minor_ref_vec,     /* minor axis reference vector */
   Pfloat           start,              /* start angle                 */
   Pfloat           end,                /* end angle                   */
   Pclosure         type                /* closure type                */

);
```

ELLIPTICAL ARC CLOSE

```
void pelliptical_arc_close (

   const Ppoint     *ctr_point,         /* centre point                */
   const Pvec       *major_ref_vec,     /* major axis reference vector */
   const Pvec       *minor_ref_vec,     /* minor axis reference vector */
   Pfloat           start,              /* start angle                 */
   Pfloat           end,                /* end angle                   */
   Pclosure         type                /* closure type                */

);
```

13.3 Attribute specification functions

13.3.1 Bundled attribute selection

SET HIGHLIGHTING INDEX

```
void pset_highl_ind (
   Pint        highl_ind                /* highlighting index          */
);
```

13.3.2 Individually selected bundled attributes

SET LINETYPE ADAPTABILITY

```
void pset_linetype_adapt (
   Plinetype_adapt    adaptability     /* linetype adaptability       */
);
```

SET LINETYPE CONTINUITY

```
void pset_linetype_cont (
   Plinetype_cont     continuity       /* linetype continuity         */
);
```

SET LINETYPE OFFSET

```
void pset_linetype_offset (
   Pfloat             offset           /* linetype offset             */
);
```

SET LINECAP

```
void pset_linecap (
   Plinecap           linecap          /* linecap                     */
);
```

SET LINEJOIN

```
void pset_linejoin (
   Plinejoin           linejoin          /* linejoin                   */
);
```

SET LINEMITRE LIMIT

```
void pset_linemitre_limit (
   Pfloat              limit             /* linemitre limit            */
);
```

SET EDGETYPE ADAPTABILITY

```
void pset_edgetype_adapt (
   Pedgetype_adapt     adaptability      /* edgetype adaptability      */
);
```

SET EDGETYPE CONTINUITY

```
void pset_edgetype_cont (
   Pedgetype_cont      continuity        /* edgetype continuity        */
);
```

SET EDGETYPE OFFSET

```
void pset_edgetype_offset (
   Pfloat              offset            /* edgetype offset            */
);
```

SET EDGECAP

```
void pset_edgecap (
   Pedgecap            edgecap          /* edgecap*/
);
```

SET EDGEJOIN

```
void pset_edgejoin (
   Pedgejoin           edgejoin         /* edgejoin                    */
);
```

SET EDGEMITRE LIMIT

```
void pset_edgemitre_limit (
   Pfloat              limit            /* edgemitre limit             */
);
```

SET HIGHLIGHTING METHOD

```
void pset_highl_method (
   const Phighl_method  *method         /* highlighting method         */
);
```

SET TRANSPARENCY

```
void pset_transparency (
   Pfloat    transparency               /* transparency                */
);
```

SET BACK TRANSPARENCY

```
void pset_back_transparency (
   Pfloat    back_transparency         /* back transparency              */
);
```

SET ALPHA SOURCE SELECTOR

```
void pset_alpha_src_sel (
   const Pint_list  *alpha_src_sel     /* alpha source selector list     */
);
```

SET ALPHA DATA SELECTION INDEX

```
void pset_alpha_data_sel_ind (
   Pint       alpha_data_sel_ind       /* alpha data selection index     */
);
```

SET ACTIVE TEXTURES

```
void pset_active_textures (
   const Pint_list  *active_textures   /* list of active textures        */
);
```

SET BACK ACTIVE TEXTURES

```
void pset_back_active_textures (
   const Pint_list  *back_active_textures /* list of active textures     */
);
```

SET TEXTURE PERSPECTIVE CORRECTION

```
void pset_texture_perspect_corr (
   Pperspect_corr   perspect_corr    /* texture perspective correction */
);
```

SET TEXTURE SAMPLING FREQUENCY

```
void pset_texture_sampling_freq (
   Pint           sampling_freq         /* texture sampling frequency  */
);
```

SET TEXTURE RESOURCE OPTIMIZATION HEURISTICS

```
void pset_texture_res_opt_heur (
   const Ptexture_res_opt_heur *heuristics   /* heuristics             */
);
```

13.3.3 Individual attributes

SET CONDITION FLAGS

```
void pset_cond_flags (
   const Pcond_flags     *flags        /* condition flag masks        */
);
```

SET CONDITION FLAGS FROM TESTS

```
void pset_cond_flags_from_tests (
   const Pcond_test_list   *tests      /* list of condition flag tests*/
);
```

13.3.4 Workstation attribute table definitions

SET POLYLINE REPRESENTATION FULL

```
void pset_line_rep_full (

   Pint                     ws_id,     /* workstation identifier      */
   Pint                     index,     /* polyline bundle index       */
   const Pline_bundle_full  *rep       /* polyline representation full*/

);
```

SET POLYLINE REPRESENTATION MASK

```
void pset_line_rep_mask (

   Pint                     ws_id,     /* workstation identifier      */
   Pint                     index,     /* polyline bundle index       */
   Ppolyline_mask           mask,      /* polyline validity mask      */
   const Pline_bundle_full  rep        /* polyline representation     */

);
```

SET POLYMARKER REPRESENTATION MASK

```
void pset_marker_rep_mask (

   Pint                       ws_id,   /* workstation identifier      */
   Pint                       index,   /* polymarker bundle index     */
   Ppolymarker_mask           mask,    /* polymarker validity mask    */
   const Pmarker_bundle_plus  rep      /* polymarker representation   */

);
```

SET TEXT REPRESENTATION MASK

```
void pset_text_rep_mask (

   Pint                       ws_id,   /* workstation identifier      */
   Pint                       index,   /* text bundle index           */
   Ptext_mask                 mask,    /* text validity mask          */
   const Ptext_bundle_plus    rep      /* text representation         */

);
```

SET INTERIOR REPRESENTATION MASK

```
void pset_int_rep_mask (
   Pint                     ws_id,    /* workstation identifier      */
   Pint                     index,    /* interior bundle index       */
   Pinterior_mask           mask,     /* interior validity mask      */
   const Pint_bundle_plus   rep       /* interior representation     */
);
```

SET EDGE REPRESENTATION FULL

```
void pset_edge_rep_full (
   Pint                    ws_id,     /* workstation identifier      */
   Pint                    index,     /* edge bundle index           */
   const Pedge_bundle_full *rep       /* edge representation full    */
);
```

SET EDGE REPRESENTATION MASK

```
void pset_edge_rep_mask (
   Pint                    ws_id,     /* workstation identifier      */
   Pint                    index,     /* edge bundle index           */
   Pedge_mask              mask,      /* edge validity mask          */
   const Pedge_bundle_full *rep       /* edge representation full    */
);
```

SET EXTENDED PATTERN REPRESENTATION

```
void pset_ext_pat_rep (
   Pint              ws_id,        /* workstation identifier           */
   Pint              pat_ind,      /* pattern bundle index             */
   const Ppattern    *pattern      /* extended pattern representation */
);
```

SET PATTERN REPRESENTATION MASK

```
void pset_pat_rep_mask (

   Pint                  ws_id,          /* workstation identifier      */
   Pint                  pat_ind,        /* pattern bundle index        */
   Ppattern_mask         mask,           /* pattern validity mask       */
   const Ppat_rep        *pat_bundle,    /* pattern representation      */
   const Ppat_rep_plus   *pat_rep_plus,  /* pattern representation plus */
   const Ppattern        *pattern        /* ext. pattern representation */

);
```

NOTE — Only the parameters indicated by the mask need be provided. Other representation parameters may reference null structures or unused representation structures;

SET TEXTURE REPRESENTATION

```
void pset_texture_rep (

   Pint                     ws_id,      /* workstation identifier        */
   Pint                     index,      /* texture index                 */
   const Ptexture_rep       *rep        /* texture representation        */

);
```

SET TEXTURE REPRESENTATION MASK

```
void pset_texture_rep_mask (

   Pint                     ws_id,      /* workstation identifier        */
   Pint                     index,      /* texture index                 */
   Ptexture_mask            mask,       /* texture validity mask         */
   const Ptexture_rep       *rep        /* texture representation        */

);
```

NOTE — Only those fields of the structures within the texture representation that are specified by the mask are referenced.

SET TEXTURE PARAMETRIZATION

```
void pset_texture_param (

   Pint                    ws_id,   /* workstation identifier          */
   Pint                    index,   /* texture index                   */
   const Ptexture_param    *param   /* texture parametrization         */

);
```

SET TEXTURE COMPOSITION

```
void pset_texture_composition (

   Pint                     ws_id,       /* workstation identifier      */
   Pint                     index,       /* texture index               */
   const Ptexture_compos    *compos      /* texture composition         */

);
```

SET TEXTURE SAMPLING

```
void pset_texture_sampling (

   Pint                      ws_id,     /* workstation identifier      */
   Pint                      index,     /* texture index               */
   const Ptexture_sampling   *sampling  /* texture sampling            */

);
```

SET TEXTURE BINDING

```
void pset_texture_binding (

   Pint                     ws_id,       /* workstation identifier      */
   Pint                     index,       /* texture index               */
   const Ptexture_binding   *binding     /* texture binding             */

);
```

SET HIGHLIGHTING REPRESENTATION

```
void pset_highl_rep (

   Pint                 ws_id,         /* workstation identifier       */
   Pint                 highl_ind,     /* highlighting bundle index    */
   const Phighl_method  *method        /* highlighting method          */

);
```

SET TRANSPARENCY MODE

```
void pset_transparency_mode (

   Pint                    ws_id,      /* workstation identifier       */
   Pint                    mode        /* transparency mode            */

);

);
```

SET DEPTH CUE REPRESENTATION MASK

```
void pset_depth_cue_rep_mask (

   Pint                    ws_id,      /* workstation identifier       */
   Pint                    index,      /* depth cue bundle index       */
   Pdepth_cue_mask         mask,       /* depth cue validity mask      */
   const Pdepth_cue_rep    *rep        /* depth cue representation     */

);
```

SET TRANSPARENCY THRESHOLDS

```
void pset_transparency_thresholds (

   Pint               ws_id,       /* workstation identifier           */
   const Pfloat_list  *thresholds  /* list of transparency thresholds */

);
```

SET REFLECTANCE REPRESENTATION MASK

```
void pset_refl_rep_mask (

   Pint                     ws_id,     /* workstation identifier      */
   Pint                     index,     /* reflectance bundle index    */
   Prefl_mask               mask,      /* reflectance validity mask   */
   const Prefl_rep          *rep       /* reflectance representation  */

);
```

13.3.5 Workstation attribute filter definition

SET APPLICATION FILTER

```
void pset_appl_filter (

   Pint             ws_id,             /* workstation identifier        */
   Pint             filter_id,         /* application filter identifier */
   const Pfilter    *filter            /* application filter            */

);
```

13.3.6 Attribute utility functions

CREATE MIPMAP TEXTURE

```
void pcreate_mipmap_texture (

   Pint     ws_id,                  /* workstation identifier         */
   Pint     src_image_res_id,       /* source image resource id       */
   Pint     mipmap_image_res_id,    /* mipmap image resource id       */
   Pint     max_levels,             /* max mipmap levels              */
   Pint     creation_heuristic      /* mipmap creation heuristic      */

);
```

DEFINE LINETYPE

```
void pdefine_linetype (

   Pint                  linetype,          /* linetype                  */
   Pfloat                repeat_length,     /* repeat length             */
   const Pfloat_list     *segment_lengths   /* segment length list       */

);
```

DEFINE MARKER TYPE

```
void pdefine_marker_type (
   Pint                marker_type,     /* marker type                */
   const Pmarker_desc *marker           /* marker descriptor          */
);
```

13.4 Transformation and clipping functions

13.4.1 View construction functions

SET VIEW REFERENCE POINT 3

```
void pset_view_ref_point3 (
   Pint            ws_id,               /* workstation id             */
   Pint            view_ind,            /* view index                 */
   const Ppoint3   *view_ref_point      /* view reference point       */
);
```

SET VIEW REFERENCE POINT

```
void pset_view_ref_point (
   Pint            ws_id,               /* workstation id             */
   Pint            view_ind,            /* view index                 */
   const Ppoint    *view_ref_point      /* view reference point       */
);
```

SET VIEW PLANE NORMAL

```
void pset_view_plane_norm (
   Pint                ws_id,           /* workstation id             */
   Pint                view_ind,        /* view index                 */
   const Pvec3         *view_norm_vec   /* view normal vector         */
);
```

SET VIEW UP VECTOR 3

```
void pset_view_up_vec3 (

   Pint                ws_id,          /* workstation id            */
   Pint                view_ind,       /* view index                */
   const Pvec3         *view_up_vec    /* view up vector            */

);
```

SET VIEW UP VECTOR

```
void pset_view_up_vec (

   Pint                ws_id,          /* workstation id            */
   Pint                view_ind,       /* view index                */
   const Pvec          *view_up_vec    /* view up vector            */

);
```

SET VIEW WINDOW LIMITS

```
void pset_view_win_limits (

   Pint                ws_id,          /* workstation id            */
   Pint                view_ind,       /* view index                */
   const Plimit        *win            /* view window limits        */

);
```

SET PROJECTION VIEWPORT 3

```
void pset_proj_vp3 (

   Pint                ws_id,          /* workstation id            */
   Pint                view_ind,       /* view index                */
   const Plimit3       *proj_vp        /* projection viewport limits */

);
```

SET PROJECTION VIEWPORT

```
void pset_proj_vp (
   Pint               ws_id,          /* workstation id            */
   Pint               view_ind,       /* view index                */
   const Plimit       *proj_vp        /* projection viewport limits */
);
```

SET PROJECTION TYPE

```
void pset_proj_type (
   Pint               ws_id,          /* workstation id            */
   Pint               view_ind,       /* view index                */
   Pproj_type         proj_type       /* projection type           */
);
```

SET PROJECTION REFERENCE POINT

```
void pset_proj_ref_point (
   Pint            ws_id,             /* workstation id            */
   Pint            view_ind,          /* view index                */
   const Ppoint3   *proj_ref_point    /* projection reference point */
);
```

SET VIEW PLANE DISTANCE

```
void pset_view_plane_dist (
   Pint               ws_id,          /* workstation id            */
   Pint               view_ind,       /* view index                */
   Pfloat             view_plane      /* view plane distance       */
);
```

SET FRONT PLANE DISTANCE

```
void pset_front_plane_dist (

   Pint                 ws_id,          /* workstation id            */
   Pint                 view_ind,       /* view index                */
   Pfloat               front_plane     /* front plane distance      */

);
```

SET BACK PLANE DISTANCE

```
void pset_back_plane_dist (

   Pint                 ws_id,          /* workstation id            */
   Pint                 view_ind,       /* view index                */
   Pfloat               back_plane      /* back plane distance       */

);
```

SET X-Y CLIPPING INDICATOR

```
void pset_xy_clip_indicator (

   Pint                 ws_id,          /* workstation id            */
   Pint                 view_ind,       /* view index                */
   Pclip_ind            xy_clip         /* xy clipping indicator     */

);
```

SET FRONT CLIPPING INDICATOR

```
void pset_front_clip_indicator (

   Pint                 ws_id,          /* workstation id            */
   Pint                 view_ind,       /* view index                */
   Pclip_ind            front_clip      /* front clipping indicator  */

);
```

SET BACK CLIPPING INDICATOR

```
void pset_back_clip_indicator (

   Pint               ws_id,          /* workstation id             */
   Pint               view_ind,       /* view index                 */
   Pclip_ind          back_clip       /* back clipping indicator    */

);
```

UPDATE VIEW REPRESENTATION

```
void pupd_view_rep (

   Pint             ws_id,            /* workstation id             */
   Pint             view_ind,         /* view index                 */
   Pint             view_upd_method   /* view update method         */

);
```

13.4.2 Workstation utility functions

MAP DEVICE COORDINATES TO WINDOW SYSTEM COORDINATES

```
void pmap_dc_to_wsc (

   Pint              ws_id,   /* workstation id                       */
   const Ppoint_list *dc,     /* device coordinate points             */
   Ppoint_list       *wsc     /* OUT window system coordinate points  */

);
```

MAP WINDOW SYSTEM COORDINATES TO DEVICE COORDINATES

```
void pmap_wsc_to_dc (

   Pint              ws_id,       /* workstation id                       */
   const Ppoint_list *wsc,        /* window system coordinate points      */
   Ppoint_list       *dc          /* OUT device coordinate points         */

);
```

MAP DEVICE COORDINATES TO WORLD COORDINATES

```
void pmap_dc_to_wc (

   Pint               ws_id,       /* workstation id                */
   Pint               view_ind,    /* view index                    */
   const Ppoint_list *dc,          /* device coordinate points      */
   Ppoint_list       *wc           /* OUT world coordinate points   */

);
```

13.5 Structure content functions

OPEN DIRECT INTERPRETATION STRUCTURE

```
void popen_di_struct (

   void

);
```

CLOSE DIRECT INTERPRETATION STRUCTURE

```
void pclose_di_struct (

   void

);
```

INSTANCE STRUCTURE

```
void pinst_struct (

   Pint         struct_id              /* structure identifier        */

);
```

CONDITIONAL EXECUTE STRUCTURE

```
void pcond_exec_struct (
   Pint                  struct_id,      /* structure identifier        */
   const Ptest           *test           /* condition test              */
);
```

CONDITIONAL INSTANCE STRUCTURE

```
void pcond_inst_struct (
   Pint                  struct_id,      /* structure identifier        */
   const Ptest           *test           /* condition test              */
);
```

CONDITIONAL RETURN

```
void pcond_return (
   const Ptest           *test           /* condition test              */
);
```

CONDITIONAL SKIP ELEMENTS

```
void pcond_skip_elements (
   Pint           skip_count,            /* number of elements to skip  */
   const Ptest    *test                  /* condition test              */
);
```

CONDITIONAL SKIP TO LABEL

```
void pcond_skip_to_label (
   Pint           label,                 /* label to which to skip      */
   const Ptest    *test                  /* condition test              */
);
```

PUSH STATE

```
void ppush_st (
   Pint                 struct_id,      /* structure identifier        */
   Pref_type            ref_type        /* reference type              */
);
```

POP STATE

```
void ppop_st (
   void
);
```

COPY ELEMENT FROM STRUCTURE

```
void pcopy_elem_struct (
   Pint           struct_id,            /* structure identifier        */
   Pint           elem_num              /* element number              */
);
```

COPY ELEMENT RANGE FROM STRUCTURE

```
void pcopy_elem_range_struct (
   Pint           struct_id,            /* structure identifier        */
   Pint           elem_ptr_1_value,     /* element position 1          */
   Pint           elem_ptr_2_value      /* element position 2          */
);
```

COPY ELEMENTS BETWEEN LABELS FROM STRUCTURE

```
void pcopy_elems_between_labels_struct (
   Pint           struct_id,            /* structure identifier        */
   Pint           label1_id,            /* label identifier 1          */
   Pint           label2_id             /* label identifier 2          */
);
```

MOVE ELEMENT FROM STRUCTURE

```
void pmove_elem_struct (

   Pint          struct_id,           /* structure identifier          */
   Pint          elem_num             /* element number                */

);
```

MOVE ELEMENT RANGE FROM STRUCTURE

```
void pmove_elem_range_struct (

   Pint          struct_id,           /* structure identifier          */
   Pint          elem_ptr_1_value,    /* element position 1            */
   Pint          elem_ptr_2_value     /* element position 2*/

);
```

MOVE ELEMENTS BETWEEN LABELS FROM STRUCTURE

```
void pmove_elems_between_labels_struct (

   Pint          struct_id,           /* structure identifier          */
   Pint          label1_id,           /* label identifier 1            */
   Pint          label2_id            /* label identifier 2            */

);
```

SET WATCH ON ELEMENT RANGE

```
void pset_watch_on_elem_range (

   Pint      struct_id,               /* watched structure identifier*/
   Pint      elem_pos1,               /* element position 1            */
   Pint      elem_pos2                /* element position 2            */

);
```

END WATCH ON ELEMENT RANGE

```
void pend_watch_on_elem_range (

   void

);
```

13.6 Structure display functions

POST STRUCTURE TO GROUP

```
void ppost_struct_to_grp (
   Pint                ws_id,          /* workstation identifier      */
   Pint                grp_id,         /* posting group identifier    */
   Pint                struct_id,      /* structure identifier        */
   Pfloat              pri             /* priority*/
);
```

UNPOST STRUCTURES FROM GROUPS

```
void punpost_structs_from_grps (
   Pint                ws_id,          /* workstation identifier      */
   Pint_list           grp_ids,        /* posting group identifiers   */
   Pint_list           struct_ids      /* structure identifiers       */
);
```

UNPOST ALL STRUCTURES FROM GROUP

```
void punpost_all_structs_from_grp (
   Pint                ws_id,          /* workstation identifier      */
   Pint                grp_id          /* posting group identifier    */
);
```

POST DIRECT INTERPRETATION STRUCTURE

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

POST DIRECT INTERPRETATION STRUCTURE TO POSTING GROUP

```
void ppost_di_struct_to_grp (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 grp_id,         /* group identifier            */
   Pint                 struct_id       /* structure identifier        */

);
```

UNPOST DIRECT INTERPRETATION STRUCTURE

```
void punpost_di_struct (

   Pint                 ws_id           /* workstation identifier      */

);
```

RENEW DIRECT INTERPRETATION STATE

```
void prenew_di_state (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 struct_id,      /* structure identifier        */
   Ptrav_type           trav_type       /* traversal type              */

);
```

SET DIRECT INTERPRETATION MODE

```
void pset_di_mode (

   Pint                 ws_id,          /* workstation identifier      */
   Ptrav_type           trav_type,      /* traversal type              */
   Pdi_mode             mode            /* direct interpretation mode  */

);
```

MARK MULTI-PASS START

```
void pmark_multi_pass_start (

   Pint                 ws_id           /* workstation identifier      */

);
```

MARK MULTI-PASS COMPLETION

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

MARK PASS START

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

MARK PASS COMPLETION

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

RETRIEVE NUMBER OF PASSES REQUIRED

```
void pret_num_passes_req (
   Pint          ws_id,              /* workstation identifier         */
   Ptrav_type    trav_type,          /* traversal type                 */
   Pint          *num_passes_req,    /* OUT picture passes required    */
   Pboolean      *delineation_req    /* OUT pass delineations required */
);
```

13.7 Input functions

13.7.1 Pick support functions

SET SET PICK FILTER

```
void pset_set_pick_filter (

   Pint                ws_id,           /* workstation identifier      */
   Pint                dev,             /* set dev. number             */
   const Pfilter       *filter          /* echo switch                 */

);
```

SET COMPOSITE PICK FILTER

```
void pset_composite_pick_filter (

   Pint                ws_id,           /* workstation identifier      */
   Pint                dev,             /* composite dev. number       */
   const Pfilter       *filter          /* echo switch                 */

);
```

SET DIRECT INTERPRETATION PICK FILTER

```
void pset_di_pick_filter (

   Pint                ws_id,           /* workstation identifier      */
   const Pfilter       *filter          /* pick filter                 */

);
```

SET DIRECT INTERPRETATION PICK CORRELATION POINT 3

```
void pset_di_pick_corr_point3 (

   Pint                ws_id,           /* workstation identifier      */
   const Ppoint3       *point           /* pick correlation point      */

);
```

SET DIRECT INTERPRETATION PICK CORRELATION POINT

```
void pset_di_pick_corr_point (

   Pint                ws_id,          /* workstation identifier      */
   const Ppoint        *point          /* pick correlation point      */

);
```

SET PICK MAPPING DATA

```
void pset_pick_mapping_data (

   Pint                   ws_id,       /* workstation identifier      */
   Pint                   pick_type,   /* pick type                   */
   Ppath_order            order,       /* pick path order             */
   const Pfilter          filter,      /* pick filter                 */
   Pecho_switch           echo,        /* echo switch                 */
   Pint                   pet,         /* prompt and echo type        */
   const Pdi_pick_data    *pick_data   /* pick data record            */

);
```

MAP DEVICE COORDINATE POINT TO PICK PATHS

```
void pmap_dc_point_to_pick_paths (

   Pint                ws_id,          /* workstation identifier      */
   const Ppoint        *point,         /* device coordinate point     */
   Pint                max_paths,      /* max paths to return         */
   Pint                depth,          /* pick path depth             */
   const Ppick_path    *start_path,    /* starting pick path          */
   Pstore              store,          /* handle to Store object      */
   Pint                *err_ind,       /* OUT error indicator         */
   Pin_status          *in_status,     /* OUT pick status             */
   Ppick_path_list     **pick          /* OUT list of pick paths      */

);
```

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

13.7.2 Logical input device definition functions

DEFINE LOCATOR

```
void pdefine_locator (
   Pint                        ws_id,            /* workstation id        */
   Pint                        dev,              /* locator dev. number   */
   Pint                        measure_proc_id,  /* locator measure proc. */
   const Ptrig_process_list    *triggers,        /* list of trigger proc. */
   const Pecho_process_list    *echos,           /* list of echo proc.    */
   const Pack_process_list     *acknowledgements /* list of ack. proc.    */
);
```

DEFINE STROKE

```
void pdefine_stroke (
   Pint                        ws_id,            /* workstation id        */
   Pint                        dev,              /* stroke dev. number    */
   Pint                        measure_proc_id,  /* stroke measure proc.  */
   const Ptrig_process_list    *triggers,        /* list of trigger proc. */
   const Pecho_process_list    *echos,           /* list of echo proc.    */
   const Pack_process_list     *acknowledgements /* list of ack. proc.    */
);
```

DEFINE VALUATOR

```
void pdefine_valuator (
   Pint                        ws_id,            /* workstation id.       */
   Pint                        dev,              /* valuator dev. number  */
   Pint                        measure_proc_id,  /* valuator measure proc*/
   const Ptrig_process_list    *triggers,        /* list of trigger proc. */
   const Pecho_process_list    *echos,           /* list of echo proc.    */
   const Pack_process_list     *acknowledgements /* list of ack. proc.    */
);
```

DEFINE CHOICE

```
void pdefine_choice (

   Pint                        ws_id,             /* workstation id.      */
   Pint                        dev,               /* choice dev. number   */
   Pint                        measure_proc_id,   /* choice measure proc. */
   const Ptrig_process_list *triggers,           /* list of trigger proc.*/
   const Pecho_process_list *echos,              /* list of echo proc.   */
   const Pack_process_list  *acknowledgements /* list of ack. proc.*/

);
```

DEFINE PICK

```
void pdefine_pick (

   Pint                        ws_id,             /* workstation id.      */
   Pint                        dev,               /* pick dev. number     */
   Pint                        measure_proc_id,   /* pick measure proc.   */
   const Ptrig_process_list *triggers,           /* list of trigger proc.*/
   const Pecho_process_list *echos,              /* list of echo proc.   */
   const Pack_process_list  *acknowledgements /* list of ack. proc.   */

);
```

DEFINE STRING

```
void pdefine_string (

   Pint                        ws_id,             /* workstation id.      */
   Pint                        dev,               /* string dev. number   */
   Pint                        measure_proc_id,   /* string measure proc. */
   const Ptrig_process_list *triggers,           /* list of trigger proc.*/
   const Pecho_process_list *echos,              /* list of echo proc.   */
   const Pack_process_list  *acknowledgements /* list of ack. proc.   */

);
```

CREATE SET MEASURE

```
void pcreate_set_measure (

 Pint                 ws_id,              /* workstation identifier   */
 const Pmeas_process  *base_measure_proc,/* measure process for set  */
 Pint                 max_measures,       /* maximum measures in set  */
 Pint                 *measure_proc_id    /* OUT set measure process  */
                                          /* identifier               */

);
```

DEFINE SET

```
void pdefine_set (

   Pint                    ws_id,            /* workstation id.       */
   Pint                    dev,              /* set dev. number       */
   Pint                    measure_proc_id,  /* set measure proc.     */
   const Ptrig_process_list *triggers,       /* list of trigger proc.*/
   const Pecho_process_list *echos,          /* list of echo proc.    */
   const Pack_process_list  *acknowledgements /* list of ack. proc.*/

);
```

CREATE COMPOSITE MEASURE

```
void pcreate_composite_measure (

   Pint                    ws_id,             /* workstation identifier */
   const Pmeas_process_list *measure_procs,   /* list of measure proc.  */
   Pint                    *measure_proc_id /* OUT composite measure  */
                                              /* process identifier     */

);
```

DEFINE COMPOSITE

```
void pdefine_composite (
   Pint                       ws_id,             /* workstation id.      */
   Pint                       dev,               /* composite dev. number*/
   Pint                       measure_proc_id,   /* composite meas. proc.*/
   const Ptrig_process_list  *triggers,         /* list of trigger proc.*/
   const Pecho_process_list  *echos,            /* list of echo proc.   */
   const Pack_process_list   *acknowledgements  /* list of ack. proc.   */
);
```

UNDEFINE LOCATOR

```
void pundefine_locator (
  Pint              ws_id,           /* workstation identifier         */
  Pint              dev              /* locator dev. number            */
);
```

UNDEFINE STROKE

```
void pundefine_stroke (
  Pint              ws_id,           /* workstation identifier         */
  Pint              dev              /* stroke dev. number             */
);
```

UNDEFINE VALUATOR

```
void pundefine_valuator (
  Pint              ws_id,           /* workstation identifier         */
  Pint              dev              /* locator dev. number            */
);
```

UNDEFINE CHOICE

```
void pundefine_choice (
 Pint             ws_id,          /* workstation identifier        */
 Pint             dev             /* choice dev. number            */
);
```

UNDEFINE PICK

```
void pundefine_pick (
 Pint             ws_id,          /* workstation identifier        */
 Pint             dev             /* pick dev. number              */
);
```

UNDEFINE STRING

```
void pundefine_string (
 Pint             ws_id,          /* workstation identifier        */
 Pint             dev             /* string dev. number            */
);
```

DESTROY SET MEASURE

```
void pdestroy_set_measure (
 Pint             ws_id,          /* workstation identifier        */
 Pint             measure_proc_id /* set measure process identifier */
);
```

UNDEFINE SET

```
void pundefine_set (
 Pint             ws_id,          /* workstation identifier        */
 Pint             dev             /* set dev. number               */
);
```

DESTROY COMPOSITE MEASURE

```
void pdestroy_composite_measure (
 Pint              ws_id,           /* workstation identifier        */
 Pint              measure_proc_id  /* composite measure process id.  */
);
```

UNDEFINE COMPOSITE

```
void pundefine_composite (
 Pint              ws_id,           /* workstation identifier        */
 Pint              dev              /* composite dev. number         */
);
```

13.7.3 Initialization of input devices

INITIALIZE SET 3

```
void pinit_set3 (
   Pint                   ws_id,           /* workstation identifier   */
   Pint                   dev,             /* set dev. number          */
   const Pset_measure     *init_measure,   /* initial measure          */
   Pint                   pet,             /* prompt and echo type     */
   const Plimit3          *echo_volume,    /* echo volume boundaries   */
   const Pset_data        *record          /* input data record        */
);
```

INITIALIZE SET

```
void pinit_set (
   Pint                   ws_id,           /* workstation identifier   */
   Pint                   dev,             /* set dev. number          */
   const Pset_measure     *init_measure,   /* initial measure          */
   Pint                   pet,             /* prompt and echo type     */
   const Plimit           *echo_area,      /* echo area boundaries     */
   const Pset_data        *record          /* input data record */
);
```

INITIALIZE COMPOSITE 3

```
void pinit_composite3 (

   Pint                     ws_id,          /* workstation identifier  */
   Pint                     dev,            /* composite dev. number   */
   const Pcomposite_measure *init_measure,  /* initial measure         */
   Pint                     pet,            /* prompt and echo type    */
   const Plimit3            *echo_volume,   /* echo volume boundaries  */
   const Pcomposite_data    *record         /* input data record       */

);
```

INITIALIZE COMPOSITE

```
void pinit_composite (

   Pint                     ws_id,          /* workstation identifier  */
   Pint                     dev,            /* composite dev. number   */
   const Pcomposite_measure *init_measure,  /* initial measure         */
   Pint                     pet,            /* prompt and echo type    */
   const Plimit             *echo_area,     /* echo area boundaries    */
   const Pcomposite_data    *record         /* input data record       */

);
```

13.7.4 Setting the mode of input devices

SET SET MODE

```
void pset_set_mode (

   Pint              ws_id,           /* workstation identifier      */
   Pint              dev,             /* set dev. number             */
   Pop_mode          mode,            /* operating mode              */
   Pecho_switch      echo             /* echo switch                 */

);
```

SET COMPOSITE MODE

```
void pset_composite_mode (
   Pint                ws_id,          /* workstation identifier      */
   Pint                dev,            /* composite dev. number       */
   Pop_mode            mode,           /* operating mode              */
   Pecho_switch        echo            /* echo switch                 */
);
```

13.7.5 Request input functions

REQUEST SET 3

```
void preq_set3 (
   Pint                    ws_id,      /* workstation identifier      */
   Pint                    dev,        /* set dev. number             */
   Pstore                  store,      /* store for measure data      */
   Pna_in_status           *status,    /* OUT non-atomic input status */
   Pset_measure            **measure   /* OUT pointer to measure data */
);
```

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

REQUEST SET

```
void preq_set (
   Pint                    ws_id,      /* workstation identifier      */
   Pint                    dev,        /* set dev. number             */
   Pstore                  store,      /* store for measure data      */
   Pna_in_status           *status,    /* OUT non-atomic input status */
   Pset_measure            **measure   /* OUT pointer to measure data */
);
```

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

REQUEST COMPOSITE 3

```
void preq_composite3 (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* composite dev. number       */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pcomposite_measure     **measure   /* OUT pointer to measure data */
);
```

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

REQUEST COMPOSITE

```
void preq_composite (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* composite dev. number       */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pcomposite_measure     **measure   /* OUT pointer to measure data */
);
```

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

13.7.6 Sample input functions

SAMPLE SET 3

```
void psample_set3 (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* set dev. number             */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pset_measure           **measure   /* OUT pointer to measure data */
);
```

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

SAMPLE SET

```
void psample_set (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* set dev. number             */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pset_measure           **measure   /* OUT pointer to measure data */
);
```

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

SAMPLE COMPOSITE 3

```
void psample_composite3 (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* composite dev. number       */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pcomposite_measure     **measure   /* OUT pointer to measure data */
);
```

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

SAMPLE COMPOSITE

```
void psample_composite (
   Pint                   ws_id,      /* workstation identifier      */
   Pint                   dev,        /* composite dev. number       */
   Pstore                 store,      /* store for measure data      */
   Pna_in_status          *status,    /* OUT non-atomic input status */
   Pcomposite_measure     **measure   /* OUT pointer to measure data */
);
```

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

13.7.7 Event input functions

GET SET 3

```
void pget_set3 (
   Pstore                 store,       /* store for measure data       */
   Pna_in_status          *status,     /* OUT non-atomic input status */
   Pset_measure           **measure    /* OUT pointer to measure data */
);
```

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

GET SET

```
void pget_set (
   Pstore                 store,       /* store for measure data       */
   Pna_in_status          *status,     /* OUT non-atomic input status */
   Pset_measure           **measure    /* OUT pointer to measure data */
);
```

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

GET COMPOSITE 3

```
void pget_composite3 (
   Pstore                 store,       /* store for measure data       */
   Pna_in_status          *status,     /* OUT non-atomic input status */
   Pcomposite_measure     **measure    /* OUT pointer to measure data */
);
```

NOTE — The memory referenced by `*measure` is managed by `store`

GET COMPOSITE

```
void pget_composite (
    Pstore                  store,       /* store for measure data      */
    Pna_in_status           *status,     /* OUT non-atomic input status */
    Pcomposite_measure      **measure    /* OUT pointer to measure data */
);
```

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

13.7.8 Local input operation functions

ATTACH LOGICAL INPUT DEVICE TO VIEW

```
void pattach_lid_to_view (
    Pint            lid_ws_id,          /* LID workstation id          */
    Pin_class       dev_class,          /* LID class                   */
    Pint            dev,                /* LID number                  */
    Pint            view_ws_id,         /* view workstation id         */
    Pint            view_ind,           /* view index                  */
    Pint            view_action_type,   /* action type                 */
    Pint            transfer_type,      /* measure transfer type       */
    Pint            view_upd_method     /* view update method          */
);
```

DETACH LOGICAL INPUT DEVICE FROM VIEW

```
void pdetach_lid_from_view (
    Pint            lid_ws_id,          /* LID workstation id          */
    Pin_class       dev_class,          /* LID class                   */
    Pint            dev,                /* LID number                  */
    Pint            view_ws_id,         /* view workstation id         */
    Pint            view_ind,           /* view index                  */
    Pint            view_action_type,   /* view action type            */
    Pint            view_upd_method     /* view update method          */
);
```

13.7.9 Local lighting functions

ATTACH LOGICAL INPUT DEVICE TO LIGHT SOURCE

```
void pattach_lid_to_light_src (

   Pint          lid_ws_id,            /* LID workstation id         */
   Pin_class     dev_class,            /* LID class                  */
   Pint          dev,                  /* LID number                 */
   Pint          light_ws_id,          /* light ws id                */
   Pint          light_src_ind,        /* light source index         */
   Pint          action,               /* light action type          */
   Pint          transfer_type         /* measure transfer type      */

);
```

DETACH LOGICAL INPUT DEVICE FROM LIGHT SOURCE

```
void pdetach_lid_from_light_src (

   Pint          LID_ws_id,            /* LID workstation id         */
   Pin_class     dev_class,            /* LID class                  */
   Pint          in_num,               /* LID number                 */
   Pint          light_ws_id,          /* light ws id                */
   Pint          light_src_ind,        /* light source index         */
   Pint          action                /* light action type          */

);
```

13.7.10 Direct interpretation picking functions

INITIALIZE DIRECT INTERPRETATION PICK 3

```
void pinit_di_pick3 (

   Pint                    ws_id,          /* workstation identifier     */
   Pint                    pick_type,      /* pick type                  */
   Pint                    max_sets,       /* max. pick sets to return   */
   Pecho_switch            echo_switch,    /* echo switch                */
   Pint                    pet,            /* prompt and echo type       */
   const Plimit3           *echo_vol,      /* echo volume                */
   const Pdi_pick_data3    *pick_data,     /* pick data record           */
   Ppath_order             order           /* pick path order            */

);
```

INITIALIZE DIRECT INTERPRETATION PICK

```
void pinit_di_pick (

   Pint                 ws_id,         /* workstation identifier      */
   Pint                 pick_type,     /* pick type                   */
   Pint                 max_sets,      /* max. pick sets to return    */
   Pecho_switch         echo_switch,   /* echo switch                 */
   Pint                 pet,           /* prompt and echo type        */
   const Plimit         *echo_area,    /* echo area                   */
   const Pdi_pick_data  *pick_data,    /* pick data record            */
   Ppath_order          order          /* pick path order             */

);
```

ENABLE DIRECT INTERPRETATION PICK

```
void penable_di_pick (

   Pint               ws_id,           /* workstation identifier      */
   Pint               struct_id        /* structure identifier        */

);
```

DISABLE DIRECT INTERPRETATION PICK

```
void pdisable_di_pick (

   Pint               ws_id            /* workstation identifier      */

);
```

TRANSFER DIRECT INTERPRETATION PICK SET

```
void ptrans_di_pick_set (

   Pint               ws_id,           /* workstation identifier      */
   Pstore             store,           /* handle to Store object      */
   Pin_status         *in_status,      /* OUT pick status             */
   Ppick_path_list    **pick           /* OUT direct interpretation   */
                                       /* pick sets                   */

);
```

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

13.8 Inquiry functions

13.8.1 Inquiry functions for PHIGS description table

INQUIRE LINETYPE DEFINITION FACILITIES

```
void pinq_linetype_def_facs (

   Pstore              store,         /* handle to store object        */
   Pint                *err_ind,      /* OUT error indicator           */
   Plinetype_def_facs  **facs         /* OUT linetype definition       */
                                      /* facilities                    */

);
```

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

INQUIRE PREDEFINED LINETYPE DEFINITION

```
void pinq_pred_linetype_def (

   Pint         linetype,              /* linetype                      */
   Pint         num_elems_appl_list,   /* # elems in appl list          */
   Pint         starting_ind,          /* starting element to return    */
   Pint         *err_ind,              /* OUT error indicator           */
   Pfloat       *repeat_length,        /* OUT dash cycle repeat length
                                          */
   Pfloat_list  *segment_lengths,      /* OUT list of segment lengths */
   Pint         *num_elems_impl_list   /* OUT # elems in impl list      */

);
```

INQUIRE MARKER TYPE DEFINITION FACILITIES

```
void pinq_marker_type_def_facs (

   Pstore                 store,     /* handle to store object         */
   Pint                   *err_ind,  /* OUT error indicator            */
   Pmarker_type_def_facs  **facs     /* OUT marker type definition     */
                                     /* facilities                     */

);
```

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

INQUIRE PREDEFINED MARKER TYPE DEFINITION

```
void pinq_pred_marker_type_def (

   Pint           marker_type,        /* marker type                  */
   Pstore         store,              /* handle to store object       */
   Pint           *err_ind,           /* OUT error indicator          */
   Pmarker_desc   **marker            /* OUT predef. marker descriptors
                                         */

);
```

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

13.8.2 Inquiry functions for PHIGS state list

INQUIRE SET OF WORKSTATIONS TO WHICH DIRECT INTERPRETATION STRUCTURE POSTED

```
void pinq_wss_di_struct_posted (

   Pint       num_elems_appl_list, /* # elems in appl. list          */
   Pint       starting_ind,        /* starting element to return     */
   Pint       *err_ind,            /* OUT error indicator            */
   Pint_list  *ws_id,              /* OUT list of ws identifiers     */
   Pint       *num_elems_impl_list /* OUT # elems in impl list       */

);
```

INQUIRE WATCH ON ELEMENT RANGE

```
void pinq_watch_elem_range (

   Pint     *err_ind,              /* OUT error indicator            */
   Poff_on  *watch_enable_flag,    /* OUT watch enable flag          */
   Pint     *struct_id,            /* OUT current watched structure  */
   Pint     *elem_pos1,            /* OUT element position 1         */
   Pint     *elem_pos2             /* OUT element position 2         */

);
```

INQUIRE WATCH RANGE STATE

```
void pinq_watch_range_st (

   Pint     *err_ind,                /* OUT error indicator                 */
   Poff_on  *primitive_erase_req,    /* OUT primitive erasures required */
   Poff_on  *primitive_redraw_req    /* OUT primitive redraws required  */

);
```

INQUIRE NUMBER OF DEFINED LINETYPES

```
void pinq_num_def_linetypes (

   Pint  *err_ind,               /* OUT error indicator                  */
   Pint  *num_defined            /* OUT number defined linetypes         */

);
```

INQUIRE NUMBER OF DEFINED MARKER TYPES

```
void pinq_num_def_marker_types (

   Pint  *err_ind,               /* OUT error indicator                  */
   Pint  *num_defined            /* OUT number defined marker types      */

);
```

INQUIRE LINETYPE DEFINITION

```
void pinq_linetype_def (

   Pint        linetype,              /* linetype                        */
   Pint        num_elems_appl_list,   /* # elems in appl list            */
   Pint        starting_ind,          /* starting element to return      */
   Pint        *err_ind,              /* OUT error indicator             */
   Pfloat      *repeat_length,        /* OUT dash cycle repeat length*/
   Pfloat_list *segment_lengths,      /* OUT list of segment lengths */
   Pint        *num_elems_impl_list   /* OUT # elems in impl list        */

);
```

INQUIRE MARKER TYPE DEFINITION

```
void pinq_marker_type_def (
   Pint           marker_type,        /* marker type                  */
   Pstore         store,              /* handle to store object       */
   Pint           *err_ind,           /* OUT error indicator          */
   Pmarker_desc   **marker            /* OUT list of marker descriptors
                                         */
);
```

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

13.8.3 Inquiry functions for workstation state list

INQUIRE LIST OF POSTING GROUPS

```
void pinq_list_post_grps (
   Pint       ws_id,                /* workstation identifier        */
   Pint       num_elems_appl_list,  /* # elems in appl. list         */
   Pint       starting_ind,         /* starting elem to return       */
   Pint       *err_ind,             /* OUT error indicator           */
   Pint_list  *grp_ids,             /* OUT list of posting groups    */
   Pint       *num_elems_impl_list  /* OUT # elems in impl. list     */
);
```

INQUIRE ASSOCIATED IMAGE RESOURCES

```
void pinq_assoc_image_res (
   Pint       ws_id,                /* workstation identifier        */
   Pint       num_elems_appl_list,  /* # elems in appl list          */
   Pint       starting_ind,         /* start index                   */
   Pint       *err_ind,             /* OUT error indicator           */
   Pint_list  **image_res_ids,      /* OUT list of image resource    */
                                    /* identifiers                   */
   Pint       *num_elems_impl_list  /* OUT # elems in impl list      */
);
```

INQUIRE DEVICE COORDINATE CLIP REGIONS 3

```
void pinq_dc_clip_regions3 (
   Pint                 ws_id,          /* workstation identifier      */
   Pstore               store,          /* handle to Store object      */
   Pint                 *err_ind,       /* OUT error indicator         */
   Plimit3_list         **limit_list    /* OUT list of clip regions    */
);
```

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

INQUIRE DEVICE COORDINATE CLIP REGIONS

```
void pinq_dc_clip_regions (
   Pint                 ws_id,          /* workstation identifier      */
   Pstore               store,          /* handle to Store object      */
   Pint                 *err_ind,       /* OUT error indicator         */
   Plimit_list          **limit_list    /* OUT list of clip regions    */
);
```

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

INQUIRE TARGET MANIPULATION MODE

```
void pinq_targ_manip_mode (
   Pint           ws_id,                /* workstation identifier      */
   Pint           *err_ind,             /* OUT error indicator         */
   Pint           *targ_manip_mode      /* OUT target manipulation mode*/
);
```

INQUIRE TARGET DISPOSITION

```
void pinq_targ_dispos (
   Pint                 ws_id,          /* workstation identifier      */
   Pstore               store,          /* handle to Store object      */
   Pint                 *err_ind,       /* OUT error indicator         */
   Ptarg_op_list        **targ_ops      /* OUT target operations list  */
);
```

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

INQUIRE DISPLAY TARGET

```
void pinq_disp_targ (

   Pint                 ws_id,           /* workstation identifier     */
   Pint                 *err_ind,        /* OUT error indicator        */
   Pint                 *offset          /* OUT offset from base target */

);
```

INQUIRE RENDERING TARGET

```
void pinq_rend_targ (

   Pint                 ws_id,           /* workstation identifier     */
   Pint                 *err_ind,        /* OUT error indicator        */
   Pint                 *offset          /* OUT offset from base target */

);
```

INQUIRE LIST OF HIGHLIGHTING INDICES

```
void pinq_list_highl_inds (

   Pint         ws_id,                   /* workstation identifier     */
   Pint         num_elems_appl_list,     /* # elems in appl list       */
   Pint         starting_ind,            /* starting element to return */
   Pint         *err_ind,                /* OUT error indicator        */
   Pint_list    *highl_inds,             /* OUT list of highl indices  */
   Pint         *num_elems_impl_list     /* OUT # elems in impl list   */

);
```

INQUIRE LIST OF DEFINED APPLICATION FILTERS

```
void pinq_list_def_appl_filters (

   Pint         ws_id,                   /* workstation id             */
   Pint         num_elems_appl_list,     /* num elems in appl list     */
   Pint         start_ind,               /* start elem index           */
   Pint         *err_ind,                /* OUT error indicator        */
   Pint_list    *filter_ids,             /* OUT list of appl. filters  */
   Pint         *num_elems_impl_list     /* OUT num elems in impl list */

);
```

INQUIRE LIST OF TEXTURE INDICES

```
void pinq_list_texture_inds (

   Pint          ws_id,                  /* workstation id             */
   Pint          num_elems_appl_list,    /* num elems in appl list     */
   Pint          start_ind,              /* start elem index           */
   Pint          *err_ind,               /* OUT error indicator        */
   Pint_list     *indices,               /* OUT list of defined texture */
                                         /* indices                    */
   Pint          *num_elems_impl_list    /* OUT num elems in impl list  */

);
```

INQUIRE TRANSPARENCY MODE

```
void pinq_transparency_mode (

   Pint                ws_id,            /* workstation id             */
   Pint                *err_ind,         /* OUT error indicator        */
   Pint                *mode             /* OUT transparency mode      */

);
```

INQUIRE TRANSPARENCY THRESHOLDS

```
void pinq_transparency_thresholds (

   Pint          ws_id,                  /* workstation id             */
   Pint          num_elems_appl_list,    /* num elems in appl list     */
   Pint          start_ind,              /* start elem index           */
   Pint          *err_ind,               /* OUT error indicator        */
   Pfloat_list   *thresholds,            /* OUT list of thresholds     */
   Pint          *num_elems_impl_list    /* OUT num elems in impl list  */

);
```

INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT 3

```
void pinq_di_pick_corr_point3 (

   Pint                ws_id,            /* workstation identifier     */
   Pint                *err_ind,         /* OUT error indicator        */
   Ppoint3             *point            /* OUT pick correlation point  */

);
```

INQUIRE DIRECT INTERPRETATION PICK CORRELATION POINT

```
void pinq_di_pick_corr_point (

   Pint                  ws_id,          /* workstation identifier      */
   Pint                  *err_ind,       /* OUT error indicator         */
   Ppoint                *point          /* OUT pick correlation point  */

);
```

INQUIRE DIRECT INTERPRETATION PICK SET STATUS

```
void pinq_di_pick_set_status (

   Pint                  ws_id,          /* workstation identifier      */
   Pint                  *err_ind,       /* OUT error indicator         */
   Ppick_stat            *pick_stat      /* OUT maximum number of pick  */
                                         /* reports reached             */

);
```

INQUIRE PICK MAPPING STATE

```
void pinq_pick_mapping_state (

   Pint                  ws_id,          /* workstation identifier      */
   Pstore                store,          /* handle to Store object      */
   Pint                  *err_ind,       /* OUT error indicator         */
   Pecho_switch          *echo,          /* OUT pick echo switch        */
   Pfilter               **filter,       /* OUT pick filter             */
   Pint                  *pick_type,     /* OUT pick type               */
   Pint                  *pet,           /* OUT pick mapping prompt and */
                                         /* echo type                   */
   Pdi_pick_data         **pick_data,    /* OUT pick data record        */
   Ppath_order           *order          /* pick path order             */

);
```

NOTE — The memory referenced by `*filter, *pick_type, and *pick_data` is managed by `store`.

INQUIRE LISTS OF AVAILABLE LOGICAL INPUT DEVICES

```
void pinq_lists_avail_lids (

   Pint       ws_id,               /* workstation identifier          */
   Pstore     store,               /* handle to Store object          */
   Pint       *err_ind,            /* OUT error indicator             */
   Pint_list  **avail_loc_dev,     /* OUT list available locator LIDs   */
   Pint_list  **avail_stroke_dev,/* OUT list available stroke LIDs    */
   Pint_list  **avail_val_dev,     /* OUT list available valuator LIDs */
   Pint_list  **avail_choice_dev,/* OUT list available choice LIDs    */
   Pint_list  **avail_pick_dev,    /* OUT list available pick LIDs      */
   Pint_list  **avail_string_dev,/* OUT list available string LIDs    */
   Pint_list  **avail_set_dev,     /* OUT list available set LIDs       */
   Pint_list  **avail_compos_dev /* OUT list available composite LIDs*/

);
```

INQUIRE POSTING GROUP

```
void pinq_post_grp (

   Pint              ws_id,            /* workstation identifier      */
   Pint              grp_id,           /* posting group identifier    */
   Pstore            store,            /* handle to Store object      */
   Pint              *err_ind,         /* OUT error indicator         */
   Pint              *view_ind,        /* OUT default view index      */
   Ppost_grp_status  *post_grp_status, /* OUT posting group status    */
   Pbackg_style      *backg_style,     /* OUT background style        */
   Pbackg_method     **backg_method,   /* OUT background method data  */
   Pborder_indic     *border_indic,    /* OUT border indicator        */
   Pint              *border_ind       /* OUT border index            */

);
```

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

INQUIRE POSTED STRUCTURES FROM POSTING GROUP

```
void pinq_posted_structs_from_post_grp (

   Pint       ws_id,                /* workstation id.                */
   Pint       num_elems_appl_list,  /* # elems in appl list           */
   Pint       start_ind,            /* starting index                 */
   Pint       *err_ind,             /* OUT error indicator            */
   Pint_list  *struct_ids,          /* OUT list of posted struct nets */
   Pint       *num_elems_impl_list  /* OUT # elems in impl list       */

);
```

INQUIRE DIRECT INTERPRETATION MODE

```
void pinq_di_mode (

   Pint              ws_id,        /* workstation identifier          */
   Ptrav_type        trav_type,    /* traversal type                  */
   Pint              *err_ind,     /* OUT error indicator             */
   Pdi_mode          *mode         /* OUT direct interpretation mode */

);
```

INQUIRE PICTURE STATUS

```
void pinq_pict_status (

   Pint              ws_id,          /* workstation identifier      */
   Ptrav_type        trav_type,      /* traversal type              */
   Pint              *err_ind,       /* OUT error indicator         */
   Ppict_stat        *pict_stat      /* OUT picture status          */

);
```

INQUIRE TARGET STATE

```
void pinq_targ_st (
   Pint                ws_id,          /* workstation identifier      */
   const Ptarg_addr    *targ_addr,     /* target address              */
   Pstore              store,          /* handle to Store object      */
   Pint                *err_ind,       /* OUT error indicator         */
   Pint                *offset,        /* OUT offset from base target */
   Ptarg_data_st       *st_nr_pict,    /* OUT state of non-retained   */
                                       /* picture                     */
   Ptarg_data_st       *st_ret_pict,   /* OUT state of retained       */
                                       /* picture                     */
   Pvisual_st          *visual_st,     /* OUT target state of         */
                                       /* visual rep.                 */
   Ptarg_empty_st      *empty_st,      /* OUT target empty            */
   Ptarg_type_list     **targ_types    /* OUT target usage list       */
);
```

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

INQUIRE TRAVERSAL RESOURCES ASSOCIATED WITH TARGET

```
void pinq_trav_res_assoc_targ (
   Pint               ws_id,               /* workstation identifier   */
   const Ptarg_addr   *targ_addr,          /* target address           */
   Pint               num_elems_appl_list, /* # elems in appl list     */
   Pint               starting_ind,        .* start index              */
   Pint               *err_ind,            /* OUT error indicator      */
   Pint_list          *res_ids,            /* OUT list of associated   */
                                           /* traversal resources      */
   Pint               *num_elems_impl_list /* OUT # elems in impl list */
);
```

INQUIRE TARGET ASSOCIATED WITH TRAVERSAL RESOURCE

```
void pinq_targ_assoc_trav_res (
   Pint              ws_id,          /* workstation identifier        */
   Pint              res_id,         /* traversal resource identifier */
   Pint              *err_ind,       /* OUT error indicator           */
   Passoc_flag       *assoc_flag,    /* OUT association flag          */
   Pint              *offset         /* OUT offset from base target   */
);
```

INQUIRE IMAGE RESOURCE

```
void pinq_image_res (

   Pint              ws_id,           /* workstation identifier      */
   Pint              image_res_id,    /* image resource identifier   */
   Pstore            store,           /* store                       */
   Pint              *err_ind,        /* OUT error indicator         */
   Pint_size         *dims,           /* OUT m,n dimensions          */
   Pimage_res        **image_res,     /* OUT image resource          */
   Paccess_flag      *access_flag     /* OUT access flag             */

);
```

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

INQUIRE POSTED DIRECT INTERPRETATION STRUCTURE

```
void pinq_posted_di_struct (

   Pint              ws_id,           /* workstation identifier      */
   Ptrav_type        trav_type,       /* traversal type              */
   Pint              *err_ind,        /* OUT error indicator         */
   Pposted_ind       *posted_ind,     /* OUT posted indicator        */
   Pint              *grp_id          /* OUT posting group identifier*/

);
```

INQUIRE POLYLINE REPRESENTATION FULL

```
void pinq_line_rep_full (

   Pint              ws_id,          /* workstation id              */
   Pint              index,          /* polyline bundle index       */
   Pint              *err_ind,       /* OUT error indicator         */
   Pline_bundle_full *rep            /* polyline representation full */

);
```

INQUIRE EDGE REPRESENTATION FULL

```
void pinq_edge_rep_full (

   Pint                ws_id,           /* workstation id             */
   Pint                index,           /* edge bundle index          */
   Pint                *err_ind,        /* OUT error indicator        */
   Pedge_bundle_full   *rep             /* edge representation full   */

);
```

INQUIRE EXTENDED PATTERN REPRESENTATION

```
void pinq_ext_pat_rep (

   Pint                ws_id,           /* workstation identifier     */
   Pint                index,           /* pattern index              */
   Pinq_type           type,            /* type of returned value     */
   Pstore              store,           /* handle to store object     */
   Pint                *err_ind,        /* OUT error indicator        */
   Ppattern            **pattern        /* OUT pattern data           */

);
```

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

INQUIRE VIEW STATUS

```
void pinq_view_status (

   Pint                ws_id,           /* workstation id             */
   Pint                view_ind,        /* view index                 */
   Pint                *err_ind,        /* OUT error indicator        */
   Pview_status        *view_status     /* OUT view status            */

);
```

INQUIRE VIEW REFERENCE POINT 3

```
void pinq_view_ref_point3 (

   Pint             ws_id,              /* workstation id             */
   Pint             view_ind,           /* view index                 */
   Pint             *err_ind,           /* OUT error indicator        */
   Ppoint3          *view_ref_point     /* OUT view reference point   */

);
```

INQUIRE VIEW REFERENCE POINT

```
void pinq_view_ref_point (

   Pint             ws_id,             /* workstation id            */
   Pint             view_ind,          /* view index                */
   Pint             *err_ind,          /* OUT error indicator       */
   Ppoint           *view_ref_point    /* OUT view reference point  */

);
```

INQUIRE VIEW PLANE NORMAL

```
void pinq_view_plane_norm (

   Pint             ws_id,             /* workstation id            */
   Pint             view_ind,          /* view index                */
   Pint             *err_ind,          /* OUT error indicator       */
   Pvec3            *view_norm_vec     /* OUT view plane normal     */

);
```

INQUIRE VIEW UP VECTOR 3

```
void pinq_view_up_vec3 (

   Pint             ws_id,             /* workstation id            */
   Pint             view_ind,          /* view index                */
   Pint             *err_ind,          /* OUT error indicator       */
   Pvec3            *view_up_vec       /* OUT view up vector        */

);
```

INQUIRE VIEW UP VECTOR

```
void pinq_view_up_vec (

   Pint             ws_id,             /* workstation id            */
   Pint             view_ind,          /* view index                */
   Pint             *err_ind,          /* OUT error indicator       */
   Pvec             *view_up_vec       /* OUT view up vector        */

);
```

INQUIRE VIEW WINDOW LIMITS

```
void pinq_view_win_limits (

   Pint                 ws_id,          /* workstation id          */
   Pint                 view_ind,       /* view index              */
   Pint                 *err_ind,       /* OUT error indicator     */
   Plimit               *win            /* OUT view window limits  */

);
```

INQUIRE PROJECTION VIEWPORT 3

```
void pinq_proj_vp3 (

   Pint             ws_id,            /* workstation id                 */
   Pint             view_ind,         /* view index                     */
   Pint             *err_ind,         /* OUT error indicator            */
   Plimit3          *proj_vp          /* OUT projection viewport limits */

);
```

INQUIRE PROJECTION VIEWPORT

```
void pinq_proj_vp (

   Pint              ws_id,           /* workstation id                 */
   Pint              view_ind,        /* view index                     */
   Pint              *err_ind,        /* OUT error indicator            */
   Plimit            *proj_vp         /* OUT projection viewport limits */

);
```

INQUIRE PROJECTION TYPE

```
void pinq_proj_type (

   Pint                 ws_id,          /* workstation id          */
   Pint                 view_ind,       /* view index              */
   Pint                 *err_ind,       /* OUT error indicator     */
   Pproj_type           *proj_type      /* OUT projection type     */

);
```

INQUIRE PROJECTION REFERENCE POINT

```
void pinq_proj_ref_point (

   Pint          ws_id,              /* workstation id                */
   Pint          view_ind,           /* view index                    */
   Pint          *err_ind,           /* OUT error indicator           */
   Ppoint3       *proj_ref_point     /* OUT projection reference point */

);
```

INQUIRE VIEW PLANE DISTANCE

```
void pinq_view_plane_dist (

   Pint               ws_id,          /* workstation id               */
   Pint               view_ind,       /* view index                   */
   Pint               *err_ind,       /* OUT error indicator          */
   Pfloat             *view_plane     /* OUT view plane distance      */

);
```

INQUIRE FRONT PLANE DISTANCE

```
void pinq_front_plane_dist (

   Pint               ws_id,          /* workstation id               */
   Pint               view_ind,       /* view index                   */
   Pint               *err_ind,       /* OUT error indicator          */
   Pfloat             *front_plane    /* OUT front plane distance     */

);
```

INQUIRE BACK PLANE DISTANCE

```
void pinq_back_plane_dist (

   Pint               ws_id,          /* workstation id               */
   Pint               view_ind,       /* view index                   */
   Pint               *err_ind,       /* OUT error indicator          */
   Pfloat             *back_plane     /* OUT back plane distance      */

);
```

INQUIRE X-Y CLIPPING INDICATOR

```
void pinq_xy_clip_indicator (

   Pint              ws_id,          /* workstation id           */
   Pint              view_ind,       /* view index               */
   Pint              *err_ind,       /* OUT error indicator      */
   Pclip_ind         *xy_clip        /* OUT x-y clipping indicator  */

);
```

INQUIRE FRONT CLIPPING INDICATOR

```
void pinq_front_clip_indicator (

   Pint              ws_id,          /* workstation id           */
   Pint              view_ind,       /* view index               */
   Pint              *err_ind,       /* OUT error indicator      */
   Pclip_ind         *front_clip     /* OUT front clipping indicator*/

);
```

INQUIRE BACK CLIPPING INDICATOR

```
void pinq_back_clip_indicator (

   Pint              ws_id,          /* workstation id           */
   Pint              view_ind,       /* view index               */
   Pint              *err_ind,       /* OUT error indicator      */
   Pclip_ind         *back_clip      /* OUT back clipping indicator */

);
```

INQUIRE HIGHLIGHTING REPRESENTATION

```
void pinq_highl_rep (

   Pint              ws_id,          /* workstation identifier      */
   Pint              index,          /* highlighting index          */
   Pinq_type         type,           /* type of returned value      */
   Pstore            store,          /* handle to store object      */
   Pint              *err_ind,       /* OUT error indicator         */
   Phighl_method     **method        /* OUT highlighting method data*/

);
```

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

INQUIRE TEXTURE REPRESENTATION

```
void pinq_texture_rep (

   Pint                ws_id,          /* workstation id               */
   Pint                index,          /* texture index                */
   Pinq_type           type,           /* type of returned values      */
   Pstore              store,          /* handle to Store object       */
   Pint                *err_ind,       /* OUT error indicator          */
   Ptexture_rep        **rep           /* OUT texture representation   */

);
```

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

INQUIRE DIRECT INTERPRETATION PICK STATE 3

```
void pinq_di_pick_st3 (

   Pint              ws_id,               /* workstation identifier     */
   Pinq_type         type,                /* type of returned value     */
   Pstore            store,               /* store for measure data     */
   Pint              *err_ind,            /* OUT error indicator        */
   Pecho_switch      *echo,               /* OUT echo switch            */
   Pfilter           **pick_filter,       /* OUT di pick filter         */
   Ppoint3           *correlation_pt,     /* OUT pick correlation point */
   Pint              *pick_type,          /* OUT pick type              */
   Pint              *max_sets,           /* OUT max sets to return     */
   Pin_status        *init_status,        /* OUT initial status         */
   Ppick_path_list   **init_pick_sets,    /* OUT initial di pick sets   */
   Pint              *pet,                /* OUT prompt and echo type   */
   Plimit3           *echo_vol,           /* OUT echo volume            */
   Ppick_data        **pick_data,         /* OUT input data record      */
   Ppath_order       *order               /* OUT pick path order        */

);
```

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

INQUIRE DIRECT INTERPRETATION PICK STATE

```
void pinq_di_pick_st (

   Pint             ws_id,            /* workstation identifier     */
   Pinq_type        type,             /* type of returned value     */
   Pstore           store,            /* store for measure data     */
   Pint             *err_ind,         /* OUT error indicator        */
   Pecho_switch     *echo_switch,     /* OUT echo switch            */
   Pfilter          **pick_filter,    /* OUT di pick filter         */
   Ppoint           *correlation_pt,  /* OUT pick correlation point */
   Pint             *pick_type,       /* OUT pick type              */
   Pint             *max_sets,        /* OUT max sets to return     */
   Pin_status       *init_status,     /* OUT initial status         */
   Ppick_path_list  **init_pick_sets, /* OUT initial di pick sets   */
   Pint             *pet,             /* OUT prompt and echo type   */
   Plimit           *echo_area,       /* OUT echo area              */
   Ppick_data       **pick_data,      /* OUT input data record      */
   Ppath_order      *order            /* OUT pick path order        */

);
```

NOTE — The memory referenced by *pick_filter, *pick_type, *init_pick_sets, and *pick_data is managed by store.

INQUIRE APPLICATION FILTER

```
void pinq_appl_filter (

   Pint              ws_id,           /* workstation identifier     */
   Pint              filter_sel,      /* filter selection           */
   Pstore            store,           /* handle to store object     */
   Pint              *err_ind,        /* OUT error indicator        */
   Pfilter           **appl_filter    /* OUT application filter     */

);
```

NOTE — The memory referenced by *appl_filter is managed by store.

NOTE — The memory referenced by *avail_loc_dev, *avail_stk_dev, *avail_val_dev, *avail_cho_dev, *avail_pik_dev, *avail_stg_dev, *avail_set_dev, and *avail_cmp_dev is managed by store.

INQUIRE LOGICAL INPUT DEVICE DEFINITION

```
void pinq_lid_def (

   Pint                   ws_id,            /* workstation identifier    */
   Pin_class              dev_class,        /* input class               */
   Pint                   device,           /* device number             */
   Pstore                 store,            /* handle to Store object    */
   Pint                   *err_ind,         /* OUT error indicator       */
   Pint                   *meas_proc,       /* OUT measure process       */
   Ptrig_process_list     **trig_procs,     /* OUT trigger processes     */
   Pecho_process_list     **echo_procs,     /* OUT echo processes        */
   Pack_process_list      **ack_procs       /* OUT ack. processes        */

);
```

NOTE — The memory referenced by `*trig_procs, *echo_procs, and *ack_procs` is managed by `store`.

INQUIRE SET MEASURE DEFINITION

```
void pinq_set_meas_def (

   Pint                     ws_id,          /* workstation identifier    */
   Pint                     meas_id,        /* set measure process id    */
   Pint                     *err_ind,       /* OUT error indicator       */
   Pint                     *max_meas,      /* OUT max measures in set   */
   Pmeas_process            *meas_proc      /* OUT base measure process  */

);
```

INQUIRE SET DEVICE STATE 3

```
void pinq_set_st3 (

   Pint                ws_id,           /* workstation identifier     */
   Pint                dev,             /* set device number          */
   Pstore              store,           /* store for measure data     */
   Pint                *err_ind,        /* OUT error indicator        */
   Pop_mode            *mode,           /* OUT operating mode         */
   Pecho_switch        *echo,           /* OUT echo switch            */
   Pfilter             **filter,        /* OUT set pick filter        */
   Pset_measure        **measure,       /* OUT initial measure        */
   Pint                *pet,            /* OUT prompt and echo type   */
   Plimit3             *echo_vol,       /* OUT echo volume            */
   Pset_data           **set_data       /* OUT input data record      */

);
```

NOTE — The memory referenced by `*filter, *measure, and *set_data` is managed by `store`.

INQUIRE SET DEVICE STATE

```
void pinq_set_st (

   Pint                ws_id,           /* workstation identifier     */
   Pint                dev,             /* set device number          */
   Pstore              store,           /* store for measure data     */
   Pint                *err_ind,        /* OUT error indicator        */
   Pop_mode            *mode,           /* OUT operating mode         */
   Pecho_switch        *echo,           /* OUT echo switch            */
   Pfilter             **filter,        /* OUT set pick filter        */
   Pset_measure        **measure,       /* OUT initial measure        */
   Pint                *pet,            /* OUT prompt and echo type   */
   Plimit              *echo_area,      /* OUT echo area              */
   Pset_data           **set_data       /* OUT input data record      */

);
```

NOTE — The memory referenced by `*filter, *measure, and *set_data` is managed by `store`.

INQUIRE COMPOSITE MEASURE DEFINITION

```
void pinq_composite_meas_def (

   Pint                   ws_id,         /* workstation identifier     */
   Pint                   meas_id,       /* composite measure proc id  */
   Pstore                 store,         /* store for measure procs    */
   Pint                   *err_ind,      /* OUT error indicator        */
   Pmeas_process_list     **meas_procs   /* OUT measure processes      */

);
```

NOTE — The memory referenced by `*meas_procs` is managed by `store`

INQUIRE COMPOSITE DEVICE STATE 3

```
void pinq_composite_st3 (

   Pint               ws_id,            /* workstation identifier     */
   Pint               dev,              /* composite dev. number      */
   Pstore             store,            /* store for measure data     */
   Pint               *err_ind,         /* OUT error indicator        */
   Pop_mode           *mode,            /* OUT operating mode         */
   Pecho_switch       *echo,            /* OUT echo switch            */
   Pfilter            **filter,         /* OUT composite pick filter  */
   Pcomposite_measure **measure,        /* OUT initial measure        */
   Pint               *pet,             /* OUT prompt and echo type   */
   Plimit3            *echo_vol,        /* OUT echo volume            */
   Pcomposite_data    **composite_data  /* OUT input data record      */

);
```

NOTE — The memory referenced by `*filter, *measure, and *composite_data` is managed by `store`.

INQUIRE COMPOSITE DEVICE STATE

```
void pinq_composite_st (

   Pint                ws_id,            /* workstation identifier     */
   Pint                dev,              /* composite dev. number      */
   Pstore              store,            /* store for measure data     */
   Pint                *err_ind,         /* OUT error indicator        */
   Pop_mode            *mode,            /* OUT operating mode         */
   Pecho_switch        *echo,            /* OUT echo switch            */
   Pfilter             **filter,         /* OUT composite pick filter  */
   Pcomposite_measure  **measure,        /* OUT initial measure        */
   Pint                *pet,             /* OUT prompt and echo type   */
   Plimit              *echo_area,       /* OUT echo area              */
   Pcomposite_data     **composite_data  /* OUT input data record      */

);
```

NOTE — The memory referenced by `*filter`, `*measure`, and `*composite_data` is managed by `store`.

INQUIRE LOGICAL INPUT DEVICE ATTACHED TO VIEW

```
void pinq_lid_attached_to_view (

   Pint         ws_id,          /* workstation identifier               */
   Pin_class    dev_class,      /* input class                          */
   Pint         dev,            /* logical input dev. number            */
   Pstore       store,          /* store for list of attachment data    */
   Pint         *err_ind,       /* OUT error indicator                  */
   Pint         *num_attached,  /* OUT num view attachments for LID     */
   Pview_attach **attached      /* OUT list of view attachments for LID*/

);
```

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

INQUIRE LOGICAL INPUT DEVICE ATTACHED TO LIGHT SOURCE

```
void pinq_lid_attached_to_light_src (

   Pint          ws_id,         /* workstation identifier          */
   Pin_class     dev_class,     /* input class                     */
   Pint          dev,           /* logical input dev. number       */
   Pstore        store,         /* store for list of attachment data */
   Pint          *err_ind,      /* OUT error indicator             */
   Pint          *num_attached, /* OUT num light source attachments */
                                /* for LID                         */
   Plight_attach **attached     /* OUT list of light source        */
                                /* attachments for LID*/

);
```

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

13.8.4 Inquiry functions for workstation description table

INQUIRE DEVICE COORDINATE CLIP REGIONS FACILITIES

```
void pinq_dc_clip_regions_facs(

   Pint          ws_type,             /* workstation type          */
   Pstore        store,               /* handle to Store object    */
   Pint          *err_ind,            /* OUT error indicator       */
   Pdc_clip_facs **facs               /* OUT DC clip region facilities
                                         */

);
```

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

INQUIRE CONFIGURATION SETTING FACILITIES

```
void pinq_config_setting_facs (

   Pint                 ws_type,       /* workstation type       */
   Pstore               store,         /* handle to Store object */
   Pint                 *err_ind,      /* OUT error indicator    */
   Pconfig_setting_facs **facs,        /* OUT configuration setting
                                          */
                                       /* facilities             */

);
```

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

INQUIRE TARGET FACILITIES

```
void pinq_targ_facs (

   Pint          ws_type,        /* workstation type                 */
   Pstore        store,          /* handle to Store object           */
   Pint          *err_ind,       /* OUT error indicator              */
   Ptarget_facs  *facs           /* OUT target facilities            */

);
```

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

INQUIRE DEFAULT TARGET DISPOSITION

```
void pinq_def_targ_disp (

   Pint          ws_type,           /* workstation type               */
   Pstore        store,             /* handle to Store object         */
   Pint          *err_ind,          /* OUT error indicator            */
   Ptarg_op_list **targ_ops         /* OUT list of target operations  */

);
```

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

INQUIRE TRAVERSAL RESOURCE FACILITIES

```
void pinq_trav_res_facs (

   Pint             ws_type,     /* workstation type                   */
   Pstore           store,       /* handle to Store object             */
   Pint             *err_ind,    /* OUT error indicator                */
   Ptrav_facs       **facs       /* OUT traversal resource facilities  */

);
```

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

INQUIRE CONDITIONAL TRAVERSAL FACILITIES

```
void pinq_cond_trav_facs (

   Pint       ws_type,          /* workstation type                 */
   Pstore     store,            /* handle to Store object           */
   Pint       *err_ind,         /* OUT error indicator              */
   Pcond_trav_facs  **facs,     /* OUT conditional traversal facilities
                                   */

);
```

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

INQUIRE DIRECT INTERPRETATION TRAVERSAL FACILITIES

```
void pinq_di_trav_facs (

   Pint             ws_type,        /* workstation type             */
   Pstore           store,          /* handle to Store object       */
   Pint             *err_ind,       /* OUT error indicator          */
   Pdi_trav_facs    **facs,         /* OUTconditionaltraversalfacilities
                                       */

);
```

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

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES TEXTURE

```
void pinq_dyns_ws_attrs_texture (

   Pint                     ws_type,     /* workstation type        */
   Pint                     *err_ind,    /* OUT error indicator     */
   Pdyn_ws_attrs_texture *dyns           /* OUT texture dynamics    */

);
```

INQUIRE DYNAMICS OF POSTING GROUPS

```
void pinq_dyns_post_grps (

   Pint                 ws_type,       /* workstation type          */
   Pint                 *err_ind,      /* OUT error indicator       */
   Pdyns_post_grps      *dyns          /* OUT posting dynamics      */

);
```

INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA 3

```
void pinq_def_di_pick_data3 (

   Pint                ws_type,         /* workstation type              */
   Pstore              store,           /* store for measure data        */
   Pint                *err_ind,        /* OUT error indicator           */
   Pint_list           **pet_list,      /* OUT list prompt/echo types    */
   Plimit3             *echo_vol,       /* OUT echo volume               */
   Pdi_pick_data       **pick_data      /* OUT default data record       */

);
```

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

INQUIRE DEFAULT DIRECT INTERPRETATION PICK DATA

```
void pinq_def_di_pick_data (

   Pint                ws_type,         /* workstation type              */
   Pstore              store,           /* store for measure data        */
   Pint                *err_ind,        /* OUT error indicator           */
   Pint_list           **pet_list,      /* OUT list prompt/echo types    */
   Plimit              *echo_area,      /* OUT echo area                 */
   Pdi_pick_data       **pick_data      /* OUT default data record       */

);
```

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

INQUIRE POSTING GROUP FACILITIES

```
void pinq_post_grp_facs (

   Pint             ws_type,      /* workstation type                   */
   Pstore           store,        /* handle to store object             */
   Pint             *err_ind,     /* OUT error indicator                */
   Ppost_grp_facs   **facs        /* OUT posting group facilities       */

);
```

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

INQUIRE IMAGE RESOURCE FACILITIES

```
void pinq_image_res_facs (

   Pint       ws_type,                /* workstation type              */
   Pstore     store,                  /* handle to store object        */
   Pint       *err_ind,               /* OUT error indicator           */
   Pint_list  **facs,                 /* OUT image resource facilities*/

);
```

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

INQUIRE LINETYPE DEFINITION SUPPORT

```
void pinq_linetype_def_support (

   Pint                ws_type,           /* workstation type          */
   Pfloat              repeat_length,     /* repeat length             */
   const Pfloat_list   *segment_lengths,  /* segment lengths           */
   Pint                *err_ind,          /* error indicator           */
   Psupport_indication*supported          /* support indication        */

);
```

INQUIRE EXTENDED PATTERN FACILITIES

```
void pinq_ext_pat_facs (

   Pint            ws_type,           /* workstation type              */
   Pstore          store,             /* handle to store object        */
   Pint            *err_ind,          /* OUT error indicator           */
   Pext_pat_facs   **facs,            /* OUT extended pattern facilities
                                         */

);
```

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

INQUIRE HIGHLIGHTING FACILITIES

```
void pinq_highl_facs (

   Pint         ws_type,              /* workstation type              */
   Pstore          store,                /* handle to Store object      */
   Pint         *err_ind,             /* OUT error indicator           */
   Phighl_facs  *facs,                /* OUT highlighting facilities   */

);
```

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

INQUIRE ALPHA FACILITIES

```
void pinq_alpha_facs (

   Pint            ws_type,              /* workstation type           */
   Pstore          store,                /* handle to Store object     */
   Pint            *err_ind,             /* OUT error indicator        */
   Palpha_facs     **facs                /* OUT alpha facilities       */

);
```

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

INQUIRE TEXTURE MAPPING FACILITIES

```
void pinq_texture_map_facs (

   Pint                ws_type,             /* workstation type        */
   Pstore              store,               /* handle to Store object  */
   Pint                *err_ind,            /* OUT error indicator     */
   Ptexture_map_facs   **texture_map_facs   /* OUT texture facilities  */

);
```

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

INQUIRE TEXTURE FACILITIES

```
void pinq_texture_facs (
   Pint                ws_type,        /* workstation type            */
   Pstore              store,          /* handle to Store object      */
   Pint                *err_ind,       /* OUT error indicator         */
   Ptexture_facs       **texture_facs  /* OUT texture facilities      */
);
```

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

INQUIRE MIPMAP FACILITIES

```
void pinq_mipmap_facs (
   Pint            ws_type,           /* workstation type              */
   Pint            *err_ind,          /* OUT error indicator           */
   Pmipmap_facs    *facs              /* OUT mipmap facilities         */
);
```

INQUIRE NUMBER OF PREDEFINED APPLICATION FILTERS

```
void pinq_num_pred_appl_filters (
   Pint  *err_ind,              /* OUT error indicator                */
   Pint  *num_predefined,       /* OUT number predefined appl. filters */
);
```

INQUIRE WORKSTATION STATE TABLE LENGTHS TEXTURE

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

INQUIRE WORKSTATION STATE TABLE LENGTHS HIGHLIGHTING

```
void pinq_ws_st_table_highl (

   Pint                    ws_type,    /* workstation type               */
   Pint                    *err_ind,   /* OUT error indicator            */
   Pws_st_tables_highl     *lengths    /* OUT lengths of workstation     */
                                       /* tables                         */
);
```

INQUIRE PICK MAPPING FACILITIES

```
void pinq_pick_mapping_facs (

   Pint                 ws_type,     /* workstation type                */
   Pstore               store,       /* store for pick mapping facs     */
   Pint                 *err_ind,    /* OUT error indicator             */
   Ppick_mapping_facs   *facs        /* OUT pick mapping facilities     */
);
```

INQUIRE NUMBER OF AVAILABLE NON-ATOMIC LOGICAL INPUT DEVICES

```
void pinq_num_avail_na_in (

   Pint        ws_type,            /* workstation type                */
   Pint        *err_ind,           /* OUT error indicator             */
   Pnum_na_in  *numin              /* OUT number of non-atomic LIDs   */
);
```

INQUIRE LOCATOR FACILITIES

```
void pinq_loc_facs (

   Pint               ws_type,     /* workstation type                */
   Pint               device,      /* locator device number           */
   Pstore             store,       /* store                           */
   Patomic_lid_facs   **facs       /* OUT locator facilities          */
);
```

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

INQUIRE STROKE FACILITIES

```
void pinq_stroke_facs (

   Pint               ws_type,     /* workstation type                 */
   Pint               device,      /* stroke device number             */
   Pstore             store,       /* store                            */
   Pint               *err_ind,    /* OUT error indicator              */
   Patomic_lid_facs   **facs       /* OUT stroke facilities            */

);
```

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

INQUIRE VALUATOR FACILITIES

```
void pinq_val_facs (

   Pint               ws_type,     /* workstation type                 */
   Pint               device,      /* valuator device number           */
   Pstore             store,       /* store                            */
   Pint               *err_ind,    /* OUT error indicator              */
   Patomic_lid_facs   **facs       /* OUT valuator facilities          */

);
```

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

INQUIRE CHOICE FACILITIES

```
void pinq_choice_facs (

   Pint               ws_type,     /* workstation type                 */
   Pint               device,      /* choice device number             */
   Pstore             store,       /* store                            */
   Pint               *err_ind,    /* OUT error indicator              */
   Patomic_lid_facs   **facs       /* OUT choice facilities            */

);
```

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

INQUIRE PICK FACILITIES

```
void pinq_pick_facs (

   Pint              ws_type,     /* workstation type          */
   Pint              device,      /* pick device number        */
   Pstore            store,       /* store                     */
   Pint              *err_ind,    /* OUT error indicator       */
   Patomic_lid_facs  **facs       /* OUT pick facilities       */

);
```

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

INQUIRE STRING FACILITIES

```
void pinq_string_facs (

   Pint              ws_type,     /* workstation type          */
   Pint              device,      /* string device number      */
   Pstore            store,       /* store                     */
   Pint              *err_ind,    /* OUT error indicator       */
   Patomic_lid_facs  **facs       /* OUT string facilities*/

);
```

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

INQUIRE SET FACILITIES

```
void pinq_set_facs (

   Pint       ws_type,                 /* workstation type          */
   Pstore     store,                   /* store                     */
   Pint       *err_ind,                /* OUT error indicator       */
   Pset_facs  **facs                   /* OUT set facilities        */

);
```

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

INQUIRE COMPOSITE FACILITIES

```
void pinq_composite_facs (

   Pint               ws_type,          /* workstation type          */
   Pint               device,           /* composite device number   */
   Pstore             store,            /* store                     */
   Pint               *err_ind,         /* OUT error indicator       */
   Pcomposite_facs    **facs            /* OUT composite facs        */

);
```

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

INQUIRE PREDEFINED POSTING GROUP

```
void pinq_pred_post_grp (

   Pint               ws_type,          /* workstation type          */
   Pint               grp_id,           /* posting group identifier  */
   Pint               *err_ind,         /* OUT error indicator       */
   Pint               *view_ind,        /* OUT default view index    */
   Ppost_grp_status   *post_grp_status, /* OUT posting group status  */
   Pbackg_style       *backg_style,     /* OUT background style      */
   Pbackg_method      *backg_method,    /* OUT background method     */
   Pborder_indic      *border_indic,    /* OUT border indicator      */
   Pint               *border_ind       /* OUT border index          */

);
```

INQUIRE PREDEFINED IMAGE RESOURCE

```
void pinq_pred_image_res (

   Pint                ws_type,         /* workstation type          */
   Pint                image_res_id,    /* image resource identifier */
   Pstore              store,           /* handle to store object    */
   Pint                *err_ind,        /* OUT error indicator       */
   Pint_size           *dims,           /* OUT m,n dimensions        */
   Pimage_res          **image_res,     /* OUT image resource        */
   Paccess_flag        *access_flag     /* OUT access flag           */

);
```

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

INQUIRE PREDEFINED ASSOCIATION OF TARGET WITH TRAVERSAL RESOURCES

```
void pinq_pred_assoc_targ_trav_res (

   Pint                ws_type,            /* workstation type       */
   const Ptarg_addr    *targ_addr,         /* target address         */
   Pint                num_elems_appl_list, /* # elems in appl. list  */
   Pint                start_ind,          /* start element to return */
   Pint                *err_ind,           /* OUT error indicator    */
   Pint_list           **res_ids,          /* OUT list of associated  */
                                           /* traversal resource ids  */
   Pint                *num_elems_impl_list /* # elems in impl. list  */

);
```

INQUIRE PREDEFINED ASSOCIATION OF TRAVERSAL RESOURCE WITH TARGET

```
void pinq_pred_assoc_trav_res_targ (

   Pint             ws_type,         /* workstation type               */
   Pint             res_id,          /* traversal resource identifier  */
   Pint             *err_ind,        /* OUT error indicator            */
   Passoc_flag      *assoc_flag,     /* OUT association flag           */
   Pint             *offset          /* OUT offset from base target    */

);
```

INQUIRE MARKER TYPE DEFINITION SUPPORT

```
void pinq_marker_type_def_support (

   Pint                ws_type,        /* workstation type          */
   const Pmarker_desc  *marker,        /* marker descriptor         */
   Pint                *err_ind,       /* error indicator           */
   Psupport_indication*supported       /* support indication        */

);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION FULL

```
void pinq_pred_line_rep_full (

   Pint                ws_type,     /* workstation type              */
   Pint                index,       /* predefined polyline bundle index*/
   Pint                *err_ind,    /* OUT error indicator           */
   Pline_bundle_full   *rep         /* polyline representation full  */

);
```

INQUIRE PREDEFINED EDGE REPRESENTATION FULL

```
void pinq_pred_edge_rep_full (

   Pint                ws_type,         /* workstation type          */
   Pint                index,           /* predefined edge bundle index*/
   Pint                *err_ind,        /* OUT error indicator       */
   Pedge_bundle_full   *rep             /* edge representation full  */

);
```

INQUIRE PREDEFINED EXTENDED PATTERN REPRESENTATION

```
void pinq_pred_ext_pat_rep (

   Pint                ws_type,         /* workstation type          */
   Pint                index,           /* predefined pattern index  */
   Pstore              store,           /* handle to store object    */
   Pint                *err_ind,        /* OUT error indicator       */
   Ppattern            **pattern        /* OUT pattern data          */

);
```

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

INQUIRE PREDEFINED HIGHLIGHTING REPRESENTATION

```
void pinq_pred_highl_rep (

   Pint             ws_type,         /* workstation type          */
   Pint             index,           /* predefined highl. index   */
   Pstore           store,           /* handle to store object    */
   Pint             *err_ind,        /* OUT error indicator       */
   Phighl_method    **method         /* OUT highl method data     */

);
```

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

INQUIRE PREDEFINED TEXTURE REPRESENTATION

```
void pinq_pred_texture_rep (

   Pint          ws_type,         /* workstation type                */
   Pint          index,           /* texture index                   */
   Pstore        store,           /* handle to Store object          */
   Pint          *err_ind,        /* OUT error indicator             */
   Ptexture_rep  **rep            /* OUT texture representation      */

);
```

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

INQUIRE PREDEFINED APPLICATION FILTER

```
void pinq_pred_appl_filter (

   Pint             ws_type,         /* workstation identifier    */
   Pint             filter_sel,      /* filter selection          */
   Pstore           store,           /* handle to store object    */
   Pint             *err_ind,        /* OUT error indicator       */
   Pfilter          **appl_filter    /* OUT application filter    */

);
```

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

INQUIRE PREDEFINED SET MEASURE DEFINITION

```
void pinq_pred_set_meas_def (

   Pint              ws_type,           /* workstation type          */
   Pint              meas_proc_id,      /* set measure proc id       */
   Pint              *err_ind,          /* OUT error indicator       */
   Pint              *max_measures,     /* OUT max measures in set   */
   Pmeas_process     *meas_proc         /* OUT base measure process  */

);
```

INQUIRE PREDEFINED COMPOSITE MEASURE DEFINITION

```
void pinq_pred_composite_meas_def (

   Pint                  ws_type,        /* workstation type          */
   Pint                  meas_proc_id,   /* composite measure proc id */
   Pstore                store,          /* store for measure data    */
   Pint                  *err_ind,       /* OUT error indicator       */
   Pmeas_process_list    **meas_procs    /* OUT measure processes     */

);
```

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

INQUIRE DEFAULT SET DEVICE DATA 3

```
void pinq_def_set_data3 (

   Pint              ws_type,          /* workstation type           */
   Pint              device,           /* set device number          */
   Pstore            store,            /* store for measure data     */
   Pint              *err_ind,         /* OUT error indicator        */
   Pint_list         **pet_list,       /* OUT list prompt/echo types */
   Plimit3           *echo_vol,        /* OUT echo volume            */
   Pset_data         **set_data        /* OUT default data record    */

);
```

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

INQUIRE DEFAULT SET DEVICE DATA

```
void pinq_def_set_data (

   Pint               ws_type,          /* workstation type              */
   Pint               device,           /* set device number             */
   Pstore             store,            /* store for measure data        */
   Pint               *err_ind,         /* OUT error indicator           */
   Pint_list          **pet_list,       /* OUT list prompt/echo types    */
   Plimit             *echo_area,       /* OUT echo area                 */
   Pset_data          **set_data        /* OUT default data record       */

);
```

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

INQUIRE DEFAULT COMPOSITE DEVICE DATA 3

```
void pinq_def_composite_data3 (

   Pint               ws_type,          /* workstation type              */
   Pint               device,           /* composite device number       */
   Pstore             store,            /* store for measure data        */
   Pint               *err_ind,         /* OUT error indicator           */
   Pint_list          **pet_list,       /* OUT list prompt/echo types    */
   Plimit3            *echo_vol,        /* OUT echo volume               */
   Pcomposite_data    **composite_data  /* OUT default data record       */

);
```

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

INQUIRE DEFAULT COMPOSITE DEVICE DATA

```
void pinq_def_composite_data (

   Pint               ws_type,          /* workstation type              */
   Pint               device,           /* composite device number       */
   Pstore             store,            /* store for measure data        */
   Pint               *err_ind,         /* OUT error indicator           */
   Pint_list          **pet_list,       /* OUT list prompt/echo types    */
   Plimit             *echo_area,       /* OUT echo area                 */
   Pcomposite_data    **composite_data  /* OUT default data record       */

);
```

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

13.8.5 Inquiry functions for structure state list

INQUIRE SET OF GROUPS TO WHICH POSTED

```
void pinq_grps_posted (

   Pint         ws_id,                   /* workstation identifier      */
   Pint         struct_id,               /* structure identifier        */
   Pint         num_elems_appl_list,     /* # elems in appl list        */
   Pint         start_ind,               /* starting index              */
   Pint         *err_ind,                /* OUT error indicator         */
   Pint_list    *grps,                   /* OUT list of posting groups  */
   Pint         *num_elems_impl_list     /* OUT # elems in impl list    */

);”
```

Annex A Data types in compilation order and external functions

Pages 155 to 161

A.1 Macro definitions

The following definition replaces the corresponding definition:

```
“
#define PE_NOT_STOP                     (5)    /* Ignoring function, func-
                                                  tion requires state
                                                  (PHOP,*,STOP | DISO,*) */”
```

The following definitions are added numerically by error number within the specified error macro definitions categories:

```
“/* State Errors */
#define PE_NOT_DISO                     (8)    /* Ignoring function, func-
                                                  tion requires state
                                                  (PHOP,*,DISO,*) */
#define PE_NOT_RETAINED_STOP_ONLY       (9)    /* Ignoring function, func-
                                                  tion requires state
                                                  (PHOP,*,STOP,*) */
#define PE_NOT_WSOP_AND_DISO            (10)   /* Ignoring function, func-
                                                  tion requires state
                                                  (PHOP,WSOP,DISO,*) */

/* Workstation Errors */
#define PE_BAD_TARG_OP                  (65)   /* Ignoring function, one of
                                                  the target operations is
                                                  not supported */
#define PE_INVALID_TRAV_RES             (66)   /* Ignoring function, the
                                                  resource is not valid */
#define PE_TRAV_RES_ASSOC_TARG          (67)   /* Ignoring function, the
                                                  traversal resource is
                                                  already associated with a
                                                  target */
#define PE_TRAV_RES_NO_DISASSOC         (68)   /* Ignoring function, the
                                                  resource cannot be disas-
                                                  sociated */
#define PE_INCOMPAT_TRAV_RES            (69)   /* Ignoring function, source
                                                  and destination traversal
                                                  resources are incompatible
                                                  */
#define PE_DI_STRUCT_NOT_POSTED         (71)   /* Ignoring function, the
                                                  direct interpretation
                                                  structure is not posted on
                                                  the specified workstation
                                                  */
#define PE_DI_REND_TRAV_PROC_ACTIVE     (72)   /* Ignoring function, the
                                                  direct interpretation ren-
                                                  der traversal process is
                                                  already active on the
                                                  specified workstation */
```

```
#define PE_DI_REND_TRAV_PROC_NOT_ACTIVE
                                          (73)  /* Ignoring function, the
                                                   direct interpretation tra-
                                                   versal process is not
                                                   active on the specified
                                                   workstation */
#define PE_CANNOT_CREATE_TARGET           (74)  /* Ignoring function, an
                                                   additional target could
                                                   not be created */
#define PE_NUM_TARGETS_NOT_REDUCED        (75)  /* Ignoring function, the
                                                   number of targets cannot
                                                   be reduced below the
                                                   default number available
                                                   for the respective work-
                                                   station type */
#define PE_TARG_NOT_DESTROYED             (76)  /* Ignoring function, the
                                                   target could not be
                                                   destroyed */
#define PE_DISP_REND_TARG_NOT_DESTROYED   (77)  /* Ignoring function, nei-
                                                   ther the display target
                                                   nor the rendering target
                                                   can be destroyed */
#define PE_MULTIPASS_IN_PROGRESS          (78)  /* Ignoring function, a
                                                   multi-pass operation is
                                                   already in progress */
#define PE_MULTI_PASS_NOT_IN_PROGRESS     (79)  /* Ignoring function, a
                                                   multi-pass operation is
                                                   not in progress */
#define PE_PASS_DEF_IN_PROGRESS           (80)  /* Ignoring function, a pass
                                                   definition is already in
                                                   progress */
#define PE_PASS_DEF_NOT_IN_PROGRESS       (81)  /* Ignoring function, a pass
                                                   definition is not in
                                                   progress */
#define PE_LT_SRC_WS_NO_OUT_CAPABILITY    (82)  /* Ignoring function, the
                                                   specified light source
                                                   workstation does not have
                                                   output capability (i.e,
                                                   the workstation category
                                                   is neither OUTPUT, OUTIN,
                                                   nor MO) */
#define PE_POST_GRP_DEF                   (83)  /* Ignoring function, the
                                                   posting group has already
                                                   been defined */
#define PE_POST_GRP_NOT_DEF               (84)  /* Ignoring function, the
                                                   posting group has not been
                                                   defined */
#define PE_POST_GRP_ID_LESS_THAN_ONE      (85)  /* Ignoring function, the
                                                   posting group identifier
                                                   is less than one */
```

```
#define PE_MAX_POST_GRP_EXCEEDED          (86)  /* Ignoring function, defin-
                                                   ing this posting group
                                                   would exceed the maximum
                                                   number of posting groups
                                                   supported on the worksta-
                                                   tion */
#define PE_IMAGE_RES_NOT_DEF              (87)  /* Ignoring function, image
                                                   resource not defined */
#define PE_BAD_IMAGE_SPEC_METHOD          (88)  /* Ignoring function, the
                                                   image resource image spec-
                                                   ification method is not
                                                   supported for this opera-
                                                   tion */
#define PE_IMAGE_RES_CAP_NOT_SUP          (89)  /* Ignoring function, image
                                                   resource capability is not
                                                   supported on the worksta-
                                                   tion */
#define PE_SPEC_IMAGE_RES_NOT_DEF         (90)  /* Ignoring function, the
                                                   specified image resource
                                                   has not been defined */
#define PE_MAX_IMAGE_RES_EXCEEDED         (91)  /* Ignoring function, associ-
                                                   ating this image resource
                                                   would exceed the maximum
                                                   number of entries allowed
                                                   in the workstation image
                                                   resource table */
#define PE_IMAGE_RES_EXCEEDS_CAP_WS       (92)  /* Ignoring function, the
                                                   image resource exceeds the
                                                   capabilities of the speci-
                                                   fied workstation */
#define PE_VW_WS_OF_CATEGORY_MI           (93)  /* Ignoring function, the
                                                   specified view worksta-
                                                   tion is of category MI */
#define PE_LID_WS_NOT_IN_NOR_OUTIN        (94)  /* Ignoring function, the
                                                   specified LID workstation
                                                   is neither of category
                                                   INPUT nor of category
                                                   OUTIN */
#define PE_NEW_WS_TYPE_NOT_REALIZED       (95)  /* Warning, Not all
                                                   attributes were fully
                                                   realized during the cre-
                                                   ation of the new worksta-
                                                   tion type */
#define PE_WS_TYPE_CANNOT_BE_MODIFIED     (96)  /* Ignoring function, work-
                                                   station type is a default
                                                   type or bound to a work-
                                                   station and cannot be mod-
                                                   ified */


/* Output Attribute Errors */
#define PE_NO_STATE_SAVED                 (141) /* Ignoring function, no
                                                   state has been saved */
```

```
#define PE_SPEC_PAT_TYPE_NOT_SUP         (142) /* Ignoring function, the
                                                  specified pattern type is
                                                  not supported on the spec-
                                                  ified workstation */
#define PE_PAT_TYPE_NOT_SUP_ON_INQ       (143  /* Ignoring function, pat-
                                                  tern type of requested
                                                  pattern table entry is not
                                                  supported by this inquiry
                                                  */
#define PE_MAX_VIEWS_EXCEEDED            (150) /* Ignoring function, set-
                                                  ting this view table entry
                                                  would exceed the maximum
                                                  number of entries allowed
                                                  */
#define PE_INVALID_LIMITS                (151) /* Ignoring function, XMIN ≥
                                                  XMAX, YMIN ≥ YMAX, ZMIN >
                                                  ZMAX, UMIN ≥ UMAX or VMIN ≥
                                                  VMAX */
#define PE_INVALID_VIEWPORT              (152) /* Ignoring function, invalid
                                                  viewport: XMIN ≥ XMAX, YMIN
                                                  ≥ YMAX, ZMIN > ZMAX */
#define PE_INVALID_VIEW_CLIP_LIM         (153) /* Ignoring function, invalid
                                                  view clipping limits: XMIN
                                                  ≥ XMAX, YMIN ≥ YMAX, ZMIN >
                                                  ZMAX */
#define PE_VIEW_CLIP_LIM_NOT_NPC         (154) /* Ignoring function, the
                                                  view clipping limits are
                                                  not within NPC range */
#define PE_PROJ_VP_LIMITS_NOT_NPC        (155) /* Ignoring function, the
                                                  projection viewport lim-
                                                  its are not within NPC
                                                  range */
#define PE_WS_WIN_LIM_NOT_NPC            (156) /* Ignoring function, the
                                                  workstation window limits
                                                  are not within NPC range
                                                  */
#define PE_WS_VP_NOT_IN_DISPLAY_SPACE    (157) /* Ignoring function, the
                                                  workstation viewport is
                                                  not within display space
                                                  */
#define PE_INVALID_VIEW_PLANES           (158) /* Ignoring function, front
                                                  plane and back plane dis-
                                                  tances same and z-extent
                                                  of projection viewport
                                                  non-zero */
#define PE_VPN_LEN_ZERO                  (159) /* Ignoring function, the
                                                  view plane normal has
                                                  length zero */
#define PE_VUP_LEN_ZERO                  (160) /* Ignoring function, the
                                                  view up vector has lenght
                                                  zero */
#define PE_VUP_VPN_PARALLEL              (161) /* Ignoring function, view up
                                                  and view plane normal vec-
                                                  tors parallel */
```

```
#define PE_PRP_BETW_FRONT_BACK            (162) /* Ignoring function, the
                                                   projection reference point
                                                   is between the front and
                                                   back planes */
#define PE_PRP_ON_VIEW_PLANE              (163) /* Ignoring function, the
                                                   projection referece point
                                                   cannot be positioned on
                                                   the view plane */
#define PE_BACK_PLN_BEFORE_FRONT_PLN      (164) /* Ignoring function, the
                                                   back plane is in front of
                                                   the front plane */
#define PE_HIGHL_IND_LT_ONE               (165) /* Ignoring function, the
                                                   highlighting index is less
                                                   than one */
#define PE_HIGHL_METHOD_NOT_AVAIL         (166) /* Ignoring function, the
                                                   specified highlighting
                                                   method is not available on
                                                   the specified workstation
                                                   */
#define PE_LINETYPE_NOT_DEFINABLE         (167) /* Ignoring function, the
                                                   specified linetype does
                                                   not reference a definable
                                                   linetype */
#define PE_MAX_DEF_LINETYPES_EXCEEDED     (168) /* Ignoring function, set-
                                                   ting this table entry
                                                   would exceed the maximum
                                                   number of entries allowed
                                                   in the definable linetype
                                                   table */
#define PE_LINETYPE_NOT_DEFINED           (169) /* Ignoring function, the
                                                   specified linetype has not
                                                   been defined */
#define PE_LINETYPE_NOT_SUP               (170) /* Warning, this linetype
                                                   definition is not sup-
                                                   ported on at least one of
                                                   the available workstation
                                                   types */
#define PE_LINETYPE_DEF_NOT_AVAIL         (171) /* Ignoring function, the
                                                   specified standard  or
                                                   implementation-defined
                                                   linetype definition is not
                                                   available */
#define PE_MARKER_TYPE_NOT_DEFINABLE      (172) /* Ignoring function, the
                                                   specified marker type does
                                                   not reference a definable
                                                   marker type */
#define PE_MAX_DEF_MARKER_TYPES_EXCEEDED  (173) /* Ignoring function, set-
                                                   ting this table entry
                                                   would exceed the maximum
                                                   number of entries allowed
                                                   in the definable marker
                                                   type table */
```

```
#define PE_MARKER_TYPE_NOT_DEFINED        (174) /* Ignoring function, the
                                                   specified marker type has
                                                   not been defined */
#define PE_MARKER_SHAPE_INVALID           (175) /* Ignoring function, the
                                                   marker shape descriptor is
                                                   invalid */
#define PE_MARKER_SHAPE_TYPE_NOT_SUP      (176) /* Ignoring function, the
                                                   specified marker shape
                                                   type is not supported */
#define PE_MARKER_TYPE_DEF_NOT_AVAIL      (177) /* Ignoring function, the
                                                   specified standard or
                                                   implementation defined
                                                   marker type definition is
                                                   not available */
#define PE_FILTER_SEL_INVALID             (178) /* Ignoring function, the
                                                   filter selection is
                                                   invalid */
#define PE_FILTER_NOT_DEFINED             (179) /* Ignoring function, the
                                                   specified application fil-
                                                   ter has not been defined
                                                   */

/* Input Errors */
#define PE_DI_PICK_TRAV_PROC_ACTIVE       (264) /* Ignoring function, the
                                                   direct interpretation pick
                                                   traversal process is
                                                   already active on the
                                                   specified workstation */
#define PE_DI_PICK_TRAV_PROC_NOT_ACTIVE   (265) /* Ignoring function, the
                                                   direct interpretation pick
                                                   traversal process is not
                                                   active on the specified
                                                   workstation */
#define PE_BAD_PICK_TYPE                  (266) /* Ignoring function, the
                                                   specified pick type is not
                                                   available on the speci-
                                                   fied workstation */
#define PE_LISTED_MEAS_PROC_INVALID       (267) /* Ignoring function, one of
                                                   the listed measure pro-
                                                   cesses is invalid */
#define PE_LISTED_TRIG_PROC_INVALID       (268) /* Ignoring function, one of
                                                   the listed trigger pro-
                                                   cesses is invalid */
#define PE_LISTED_ECHO_PROC_INVALID       (269) /* Ignoring function, one of
                                                   the listed echo processes
                                                   is invalid */
#define PE_LISTED_ACK_PROC_INVALID        (270) /* Ignoring function, one of
                                                   the listed acknowledge-
                                                   ment processes is invalid
                                                   */
#define PE_LID_NR_IN_USE                  (271) /* Ignoring function, device
                                                   number already in use */
#define PE_NR_MEAS_INVALID                (272) /* Ignoring function, number
                                                   of measures is invalid */
```

```
#define PE_MEAS_ID_NOT_AVAIL                (273) /* Ignoring function, the
                                                     specified measure identi-
                                                     fier is not available on
                                                     the specified workstation
                                                     */
#define PE_MEAS_ID_IN_USE                   (274) /* Ignoring function, the
                                                     specified measure identi-
                                                     fier is currently part of
                                                     a set or composite logi-
                                                     cal input device defini-
                                                     tion */
#define PE_NO_PICK_MEASURE                  (275) /* Ignoring function, the
                                                     specified set or compos-
                                                     ite device does not have a
                                                     pick measure */
#define PE_LID_INAPPROP_FOR_VW_ACTION       (276) /* Ignoring function, the
                                                     logical input device class
                                                     specified is inappropri-
                                                     ate for the view action
                                                     specified */
#define PE_LID_INAPPROP_FOR_LT_ACT          (277) /* Ignoring function, the
                                                     input device class speci-
                                                     fied is inappropriate for
                                                     the light action speci-
                                                     fied */
#define PE_LT_ACT_INAPPROP_FOR_LT_SRC       (278) /* Ignoring function, the
                                                     light action is inappro-
                                                     priate for the light
                                                     source specified */


/* Miscellaneous Errors */
#define PE_TEST_OP_INVALID                  (451) /* Ignoring function, invalid
                                                     test operator for test type
                                                     */
#define PE_NO_WINDOW_SYSTEM                 (452) /* Ignoring function, the
                                                     specified workstation is
                                                     not operating in a window
                                                     management system envrion-
                                                     ment */
#define PE_WIN_SYS_COLR_NOT_DETER           (453) /* Ignoring function, the
                                                     window system colour can-
                                                     not be determined */
#define PE_TEXTR_COORD_SRC_NOT_AVAIL        (600) /* Ignoring function, the
                                                     specified coordinate
                                                     source is not available on
                                                     the specified workstation
                                                     */
#define PE_DATA_VALUE_IND_LESS_1            (601) /* Ignoring function, the
                                                     specified data value index
                                                     is less than one */

#define PE_TEXTR_COMPOS_METHOD_NOT_AVAIL
                                            (602) /* Ignoring function, the
                                                     specified composition
                                                     method is not available on
```

```
                                                    the specified workstation
                                                    */
#define PE_TEXTR_MIN_METHOD_NOT_AVAIL         (603) /* Ignoring function, the
                                                    specified minification
                                                    method is not available on
                                                    the specified workstation
                                                    */
#define PE_TEXTR_MAG_METHOD_NOT_AVAIL         (604) /* Ignoring function, the
                                                    specified magnification
                                                    method is not available on
                                                    the specified workstation
                                                    */
#define PE_BOUND_COND_NOT_AVAIL               (605) /* Ignoring function, the
                                                    specified boundary condi-
                                                    tion is not available on
                                                    the specified workstation
                                                    */
#define PE_BOUND_CLAMP_METHOD_NOT_AVAIL
                                              (606) /* Ignoring function, the
                                                    specified boundary clamp
                                                    method is not available on
                                                    the specified workstation
                                                    */
#define PE_DEPTH_SAMPL_BIAS_OUT_OF_RANGE
                                              (607) /* Ignoring function, the
                                                    specified depth sampling
                                                    bias hint is out of range
                                                    */
#define PE_TEXTR_IMAGE_RES_NOT_DEFINED        (608) /* Ignoring function, the
                                                    specified texture image
                                                    resource identifier is not
                                                    defined on the specified
                                                    workstation */
#define PE_RENDERING_ORDER_NOT_AVAIL          (612) /* Ignoring function, the
                                                    specified rendering order
                                                    is not available on the
                                                    specified workstation */"
```

Pages 161 to 164

The following definitions are added numerically by function number to the list of function name macro definitions:

```
#define Pfn_ws_type_create                    (268)
#define Pfn_ws_type_set                       (269)
#define Pfn_ws_type_get                       (270)
#define Pfn_ws_type_destroy                   (271)
#define Pfn_redraw_all_structs_from_grp       (272)
#define Pfn_redraw_all_structs_on_targ        (273)
#define
Pfn_redraw_all_structs_from_grp_on_targ       (274)
#define Pfn_upd_targ                          (275)
```

```
#define Pfn_define_post_grp                   (276)
#define Pfn_undefine_post_grp                 (277)
#define Pfn_set_post_grp_status               (278)
#define Pfn_set_post_grp_priority             (279)
#define Pfn_set_post_grp_backg_style          (280)
#define Pfn_set_post_grp_backg_method         (281)
#define Pfn_set_post_grp_border_indic         (282)
#define Pfn_set_post_grp_border_ind           (283)
#define Pfn_assoc_image_res                   (284)
#define Pfn_disassoc_image_res                (285)
#define Pfn_set_dc_clip_regions3              (286)
#define Pfn_set_dc_clip_regions               (287)
#define Pfn_set_targ_manip_mode               (288)
#define Pfn_set_targ_dispos                   (289)
#define Pfn_set_disp_targ                     (290)
#define Pfn_set_rend_targ                     (291)
#define Pfn_clear_targ                        (292)
#define Pfn_copy_targ                         (293)
#define Pfn_create_targ                       (294)
#define Pfn_destroy_targ                      (295)
#define Pfn_set_st_visual_rep                 (296)
#define Pfn_set_targ_st_visual_rep            (297)
#define Pfn_assoc_trav_res                    (298)
#define Pfn_disassoc_trav_res                 (299)
#define Pfn_manip_trav_res                    (300)
#define Pfn_reset_all_trav_res                (301)
#define Pfn_ret_window_system_colr            (302)
#define Pfn_set_appl_int                      (303)
#define Pfn_set_appl_real                     (304)
#define Pfn_circle3                           (305)
#define Pfn_circle                            (306)
#define Pfn_circular_arc3                     (307)
#define Pfn_circular_arc                      (308)
#define Pfn_ellipse3                          (309)
#define Pfn_ellipse                           (310)
#define Pfn_elliptical_arc3                   (311)
#define Pfn_elliptical_arc                    (312)
#define Pfn_fill_circle3                      (313)
#define Pfn_fill_circle                       (314)
#define Pfn_circular_arc_close3               (315)
#define Pfn_circular_arc_close                (316)
#define Pfn_fill_ellipse3                     (317)
#define Pfn_fill_ellipse                      (318)
#define Pfn_elliptical_arc_close3             (319)
#define Pfn_elliptical_arc_close              (320)
#define Pfn_set_highl_ind                     (321)
#define Pfn_set_linetype_adapt                (322)
#define Pfn_set_linetype_cont                 (323)
#define Pfn_set_linetype_offset               (324)
#define Pfn_set_linecap                       (325)
```

```
#define Pfn_set_linejoin                        (326)
#define Pfn_set_linemitre_limit                 (327)
#define Pfn_set_edgetype_adapt                  (328)
#define Pfn_set_edgetype_cont                   (329)
#define Pfn_set_edgetype_offset                 (330)
#define Pfn_set_edgecap                         (331)
#define Pfn_set_edgejoin                        (332)
#define Pfn_set_edgemitre_limit                 (333)
#define Pfn_set_highl_method                    (334)
#define Pfn_set_transparency                    (335)
#define Pfn_set_back_transparency               (336)
#define Pfn_set_alpha_src_sel                   (337)
#define Pfn_set_alpha_data_sel_ind              (338)
#define Pfn_set_active_textures                 (339)
#define Pfn_set_back_active_textures            (340)
#define Pfn_set_texture_perspect_corr           (341)
#define Pfn_set_texture_sampling_freq           (342)
#define Pfn_set_texture_res_opt_heur            (343)
#define Pfn_set_cond_flags                      (344)
#define Pfn_set_cond_flags_from_tests           (345)
#define Pfn_set_line_rep_full                   (346)
#define Pfn_set_line_rep_mask                   (347)
#define Pfn_set_marker_rep_mask                 (348)
#define Pfn_set_text_rep_mask                   (349)
#define Pfn_set_int_rep_mask                    (350)
#define Pfn_set_edge_rep_full                   (351)
#define Pfn_set_edge_rep_mask                   (352)
#define Pfn_set_ext_pat_rep                     (353)
#define Pfn_set_pat_rep_mask                    (354)
#define Pfn_set_texture_rep                     (355)
#define Pfn_set_texture_rep_mask                (356)
#define Pfn_set_texture_param                   (357)
#define Pfn_set_texture_composition             (358)
#define Pfn_set_texture_sampling                (359)
#define Pfn_set_texture_binding                 (360)
#define Pfn_set_highl_rep                       (361)
#define Pfn_set_transparency_mode               (362)
#define Pfn_set_transparency_thresholds         (363)
#define Pfn_set_refl_rep_mask                   (364)
#define Pfn_set_appl_filter                     (365)
#define Pfn_create_mipmap_texture               (366)
#define Pfn_define_linetype                     (367)
#define Pfn_define_marker_type                  (368)
#define Pfn_set_view_ref_point3                 (369)
#define Pfn_set_view_ref_point                  (370)
#define Pfn_set_view_plane_norm                 (371)
#define Pfn_set_view_up_vec3                    (372)
#define Pfn_set_view_up_vec                     (373)
#define Pfn_set_view_win_limits                 (374)
#define Pfn_set_proj_vp3                        (375)
```

```
#define Pfn_set_proj_vp                          (376)
#define Pfn_set_proj_type                        (377)
#define Pfn_set_proj_ref_point                   (378)
#define Pfn_set_view_plane_dist                  (379)
#define Pfn_set_front_plane_dist                 (380)
#define Pfn_set_back_plane_dist                  (381)
#define Pfn_set_xy_clip_indicator                (382)
#define Pfn_set_front_clip_indicator             (383)
#define Pfn_set_back_clip_indicator              (384)
#define Pfn_upd_view_rep                         (385)
#define Pfn_map_dc_to_wsc                        (386)
#define Pfn_map_wsc_to_dc                        (387)
#define Pfn_map_dc_to_wc                         (388)
#define Pfn_open_di_struct                       (389)
#define Pfn_close_di_struct                      (390)
#define Pfn_inst_struct                          (391)
#define Pfn_cond_exec_struct                     (392)
#define Pfn_cond_inst_struct                     (393)
#define Pfn_cond_return                          (394)
#define Pfn_cond_skip_elements                   (395)
#define Pfn_cond_skip_to_label                   (396)
#define Pfn_push_st                              (397)
#define Pfn_pop_st                               (398)
#define Pfn_copy_elem_struct                     (399)
#define Pfn_copy_elem_range_struct               (400)
#define Pfn_copy_elems_between_labels_struct(401)
#define Pfn_move_elem                            (402)
#define Pfn_move_elem_range                      (403)
#define Pfn_move_elems_between_labels            (404)
#define Pfn_set_watch_on_elem_range              (405)
#define Pfn_end_watch_on_elem_range              (406)
#define Pfn_post_struct_to_grp                   (407)
#define Pfn_unpost_structs_from_grps             (408)
#define Pfn_unpost_all_structs_from_grp          (409)
#define Pfn_post_di_struct                       (410)
#define Pfn_post_di_struct_to_grp                (411)
#define Pfn_unpost_di_struct                     (412)
#define Pfn_renew_di_state                       (413)
#define Pfn_set_di_mode                          (414)
#define Pfn_mark_multi_pass_start                (415)
#define Pfn_mark_multi_pass_compl                (416)
#define Pfn_mark_pass_start                      (417)
#define Pfn_mark_pass_compl                      (418)
#define Pfn_ret_num_passes_req                   (419)
#define Pfn_set_set_pick_filter                  (420)
#define Pfn_set_composite_pick_filter            (421)
#define Pfn_set_di_pick_filter                   (422)
#define Pfn_set_di_pick_corr_point3              (423)
#define Pfn_set_di_pick_corr_point               (424)
#define Pfn_set_pick_mapping_data                (425)
```

```
#define Pfn_map_dc_point_to_pick_paths      (426)
#define Pfn_define_locator                  (427)
#define Pfn_define_stroke                   (428)
#define Pfn_define_valuator                 (429)
#define Pfn_define_choice                   (430)
#define Pfn_define_pick                     (431)
#define Pfn_define_string                   (432)
#define Pfn_create_set_measure              (433)
#define Pfn_define_set                      (434)
#define Pfn_create_composite_measure        (435)
#define Pfn_define_composite                (436)
#define Pfn_undefine_locator                (437)
#define Pfn_undefine_stroke                 (438)
#define Pfn_undefine_valuator               (439)
#define Pfn_undefine_choice                 (440)
#define Pfn_undefine_pick                   (441)
#define Pfn_undefine_string                 (442)
#define Pfn_destroy_set_measure             (443)
#define Pfn_undefine_set                    (444)
#define Pfn_destroy_composite_measure       (445)
#define Pfn_undefine_composite              (446)
#define Pfn_init_set3                       (447)
#define Pfn_init_set                        (448)
#define Pfn_init_composite3                 (449)
#define Pfn_init_composite                  (450)
#define Pfn_set_set_mode                    (451)
#define Pfn_set_composite_mode              (452)
#define Pfn_req_set3                        (453)
#define Pfn_req_set                         (454)
#define Pfn_req_composite3                  (455)
#define Pfn_req_composite                   (456)
#define Pfn_sample_set3                     (457)
#define Pfn_sample_set                      (458)
#define Pfn_sample_composite3               (459)
#define Pfn_sample_composite                (460)
#define Pfn_get_set3                        (461)
#define Pfn_get_set                         (462)
#define Pfn_get_composite3                  (463)
#define Pfn_get_composite                   (464)
#define Pfn_attach_lid_to_view              (465)
#define Pfn_detach_lid_from_view            (466)
#define Pfn_attach_lid_to_light_src         (467)
#define Pfn_detach_lid_from_light_src       (468)
#define Pfn_init_di_pick3                   (469)
#define Pfn_init_di_pick                    (470)
#define Pfn_enable_di_pick                  (471)
#define Pfn_disable_di_pick                 (472)
```

```
#define Pfn_trans_di_pick_set                (473)
```
”

Page 164

The following text is appended after the definition of PLINE_DASH_DOT*:*

“
```
#define PLINE_DASH_DOT_DOT                   (5)
```
”

Page 165 (Page 99 of PHIGS PLUS)

The following text is appended the definition of PCOLR_HLS*:*

“
```
#define PCOLR_RGBA                           (5)
#define PCOLR_CIELUVA                        (6)
#define PCOLR_HSVA                           (7)
#define PCOLR_HLSA                           (8)
```
”

Page 166

The following text is appended:

“
```
#define PHATCH_HORIZ                         (1)
#define PHATCH_VERT                          (2)
#define PHATCH_POS_SLOPE                     (3)
#define PHATCH_NEG_SLOPE                     (4)
#define PHATCH_HORIZ_VERT_CROSS              (5)
#define PHATCH_POS_NEG_SLOPE_CROSS           (6)

#define PPICK_HIGHL_PRIM                     (4)
#define PPICK_HIGHL_PRIM_GROUP               (5)
#define PPICK_HIGHL_STRUCT_NETWORK           (6)

#define PDI_PICK_HIGHL                       (1)

#define PPICK_TYPE_WS_DEP                    (1)
#define PPICK_TYPE_PICK_FIRST                (2)
#define PPICK_TYPE_PICK_LAST                 (3)
#define PPICK_TYPE_PICK_CLOSEST              (4)

#define PTARG_MANIP_AUTO                     (1)
#define PTARG_MANIP_MANUAL                   (2)

#define PSET_REND_TARG                       (1)
#define PSET_DISP_TARG                       (2)
#define PCLEAR_TARG                          (3)
#define PCOPY_TARG                           (4)
```

```
#define PCLEAR_TRAV_RES                              (0)
#define PCOPY_TRAV_RES                               (1)

#define PTRAV_RES_Z_BUF                              (1)
#define PTRAV_RES_PAINT_BUF                          (2)

#define PVUM_APPLY_ALL                               (1)
#define PVUM_APPLY_ORIENTATION                       (2)

#define PVWA_ROT_ABOUT_U_VRC_AXIS                    (1)
#define PVWA_ROT_ABOUT_V_VRC_AXIS                    (2)
#define PVWA_ROT_ABOUT_N_VRC_AXIS                    (3)
#define PVWA_TRANS_U_VRC_AXIS_CLIP                   (4)
#define PVWA_TRANS_U_VRC_AXIS_NOCLIP                 (5)
#define PVWA_TRANS_V_VRC_AXIS_CLIP                   (6)
#define PVWA_TRANS_V_VRC_AXIS_NOCLIP                 (7)
#define PVWA_TRANS_N_VRC_AXIS                        (8)
#define PVWA_SCALE_WIN_UNIFORMLY                     (9)
#define PVWA_SCALE_VP_UNIFORMLY                      (10)
#define PVWA_SCALE_WIN_WIDTH                         (11)
#define PVWA_SCALE_VP_WIDTH                          (12)
#define PVWA_SCALE_WIN_HEIGHT                        (13)
#define PVWA_SCALE_VP_HEIGHT                         (14)
#define PVWA_TRANS_FRONT_CLIP_PLANE                  (15)
#define PVWA_TRANS_BACK_CLIP_PLANE                   (16)
#define PVWA_ROT_AROUND_VRP_ALONG_CYL_LAT            (17)
#define PVWA_TRANS_EYE_FROM_VRP_OUT                  (18)
#define PVWA_TRANS_EYE_FROM_VRP_UP                   (19)
#define PVWA_ROT_EYE_FROM_VRP_UP                     (20)
#define PVWA_CHG_EYE_POS_WRT_VRP                     (21)
#define PVWA_CHG_VRP_POS_ROT_UP_WRT_PRP              (22)
#define PVWA_CHG_VRP_POS_ROT_RL_WRT_PRP              (23)

#define PMTT_MEASURE_CHANGE                          (1)
#define PMTT_EVENT_WITH_DISCARD                      (2)
#define PMTT_EVENT_WITH_PROPAGATION                  (3)

#define PBACKGM_COLOUR_TABLE_0                       (1)
#define PBACKGM_COLOUR_INDEX                         (2)
#define PBACKGM_COLOUR                               (3)
#define PBACKGM_IMAGE_RESOURCE                       (4)

#define PIMGSM_UNCOMP_COLR_INDEX_ARRAY               (1)
#define PIMGSM_UNCOMP_COLR_ARRAY                     (2)
#define PIMGSM_WINDOW_SYS_BITMAP                     (3)
#define PIMGSM_VIDEO_SIGNAL_CHAN_ID                  (4)
#define PIMGSM_LUMINANCE                             (5)
#define PIMGSM_LUMINANCE_ALPHA                       (6)
#define PIMGSM_MIPMAP_COLRS                          (7)
#define PIMGSM_MIPMAP_LUMINANCE                      (8)
#define PIMGSM_MIPMAP_LUMINANCE_ALPHA                (9)

#define PPATT_COLOUR_INDEX                           (1)
#define PPATT_COLOUR                                 (2)
```

```
#define PPATT_IMAGE_RESOURCE                    (3)

#define PSHAPE_POLYLINE                         (1)
#define PSHAPE_FILL_AREA                        (2)
#define PSHAPE_CONVEX_FILL_AREA                 (3)

#define PTEST_ALWAYS_FAIL                       (0)
#define PTEST_ALWAYS_SUCCEED                    (1)
#define PTEST_EXTENT3                           (2)
#define PTEST_EXTENT                            (3)
#define PTEST_BOUNDS3                           (4)
#define PTEST_BOUNDS                            (5)
#define PTEST_NAMESET                           (6)
#define PTEST_APPL_INTEGER                      (7)
#define PTEST_APPL_REAL                         (8)

#define PFILTER_TYPE_INVIS                      (1)
#define PFILTER_TYPE_HIGHL                      (2)
#define PFILTER_TYPE_PICK                       (3)
#define PFILTER_TYPE_APPL                       (4)

#define PTEXTRSF_PER_PIXEL                      (1)
#define PTEXTRSF_INTERP_DEP                     (2)

#define PTEXTR_OPT_NO_HINT                      (1)
#define PTEXTR_OPT_SPEED                        (2)
#define PTEXTR_OPT_SPACE                        (3)

#define PALPHASRC_TRANSPARENCY                  (1)
#define PALPHASRC_COLR_ASPECT                   (2)
#define PALPHASRC_FACET_COLR                    (3)
#define PALPHASRC_FACET_DATA                    (4)
#define PALPHASRC_VERTEX_COLR                   (5)
#define PALPHASRC_VERTEX_DATA                   (6)

#define PCOORDSRC_VERTEX_COORD_MC               (1)
#define PCOORDSRC_VERTEX_COORD_WC               (2)
#define PCOORDSRC_VERTEX_NORMAL_MC              (3)
#define PCOORDSRC_VERTEX_NORMAL_WC              (4)
#define PCOORDSRC_NURB_SURF_UV                  (5)
#define PCOORDSRC_DATA                          (6)
#define PCOORDSRC_REFL_SPHERE_VRC               (7)
#define PCOORDSRC_REFL_SPHERE_WC                (8)

#define PGCS_GAMMA_ONE                          (1)
#define PGCS_GAMMA_RED                          (1)
#define PGCS_GAMMA_TWO                          (2)
#define PGCS_GAMMA_GREEN                        (2)
#define PGCS_GAMMA_THREE                        (3)
#define PGCS_GAMMA_BLUE                         (3)

#define PCOMPOS_REPLACE                         (1)
#define PCOMPOS_MODULATE                        (2)
#define PCOMPOS_BLEND_ENVIRONMENT_COLR          (3)
```

```
#define PCOMPOS_DECAL                           (4)
#define PCOMPOS_DECAL_BACKGROUND                (5)

#define PMINM_SINGLE_BASE                       (1)
#define PMINM_LINEAR_BASE                       (2)
#define PMINM_SINGLE_IN_MIPMAP                  (3)
#define PMINM_LINEAR_IN_MIPMAP                  (4)
#define PMINM_SINGLE_BETWEEN_MIPMAPS            (5)
#define PMINM_LINEAR_BETWEEN_MIPMAPS            (6)

#define PMAGM_SINGLE_BASE                       (1)
#define PMAGM_LINEAR_BASE                       (2)
#define PMAGM_SINGLE_IN_MIPMAP                  (3)
#define PMAGM_LINEAR_IN_MIPMAP                  (4)
#define PMAGM_SINGLE_BETWEEN_MIPMAPS            (5)
#define PMAGM_LINEAR_BETWEEN_MIPMAPS            (6)

#define PBOUND_COND_CLAMP_EXPLICIT_METHOD       (1)
#define PBOUND_COND_CLAMP_BORDER                (2)
#define PBOUND_COND_WRAP                        (3)
#define PBOUND_COND_MIRROR                      (4)

#define PBOUND_CLAMP_DISCONTINUE                (1)
#define PBOUND_CLAMP_COLR                       (2)

#define PRENDORDR_PRELIGHT                      (1)
#define PRENDORDR_POST_LIGHT                    (2)

#define PTRANSPMODE_NONE                        (1)
#define PTRANSPMODE_SCREEN_DOOR                 (2)
#define PTRANSPMODE_ALPHA_BLEND                 (3)
#define PTRANSPMODE_TWO_PASS                    (4)
#define PTRANSPMODE_MULTIPASS                   (5)

#define PLTA_ADJ_LT_COLR_COMP_1                 (1)
#define PLTA_ADJ_LT_COLR_COMP_2                 (2)
#define PLTA_ADJ_LT_COLR_COMP_3                 (3)
#define PLTA_ADJ_BRIGHT                         (4)
#define PLTA_ROT_AROUND_X_DIR_VEC               (5)
#define PLTA_ROT_AROUND_Y_DIR_VEC               (6)
#define PLTA_ROT_AROUND_Z_DIR_VEC               (7)
#define PLTA_TRANS_X_LT_POS                     (8)
#define PLTA_TRANS_Y_LT_POS                     (9)
#define PLTA_TRANS_Z_LT_POS                     (10)
#define PLTA_ADJ_CONCENTRATION_EXP              (11)
#define PLTA_ADJ_LT_SPREAD_ANGLE                (12)”
```

Page 169

A.2 Types in compilation order

The following text replaces the definition of Pint_style*:*

“
```
 typedef enum {

   /* start of PHIGS enumerations */
   PSTYLE_HOLLOW,
   PSTYLE_SOLID,
   PSTYLE_HATCH,
   PSTYLE_PATTERN,
   PSTYLE_EMPTY,
   /* end of PHIGS enmerations */
   /* start of Full PHIGS enumerations */
   PSTYLE_TEXTURE
   /* end of Full PHIGS enumerations */

 } Pint_style;”
```

The following text is merged into the definition of Paspect*:*

“
```
 typedef enum {

   /* start of PHIGS enumerations */
   ...,
   /* end of PHIGS enumerations */
   /* start of PHIGS PLUS enumerations */
   ...,
   /* end of PHIGS PLUS enumerations */
   /* start of Full PHIGS enumerations */
   PASPECT_HIGHL_METHOD
   /* end of Full PHIGS enumerations */

 } Paspect;”
```

Page 171

The following text replaces the text of the definition of popen_struct_status*:*

“
```
 typedef enum {

   /* start of PHIGS enumerations */
   PSTRUCT_NONE,
   PSTRUCT_OPEN,
   /* end of PHIGS enumerations */
   /* start of Full PHIGS enumerations */
   PSTRUCT_DI
   /* end of Full PHIGS enumerations */

 } Popen_struct_status;”
```

The following text replaces the text of the definition of pin_class*:*

```
“
 typedef enum {

    /* start of Basic PHIGS enumerations */
    PIN_NONE,
    PIN_LOC,
    PIN_STROKE,
    PIN_VAL,
    PIN_CHOICE,
    PIN_PICK,
    PIN_STRING,
    /* end of Basic PHIGS enumerations */
    /* start of Full PHIGS enumerations */
    PIN_SET,
    PIN_COMPOSITE
    /* end of Full PHIGS enumerations */

 } Pin_class;”
```

Page 174

The following text is merged into the text of the definition of pelem_type*:*

```
“
 typedef enum {

    /* start of PHIGS enumerations */
    ...,
    /* end of PHIGS enumerations */
    /* start of PHIGS PLUS enumerations */
    ...,
    /* end of PHIGS PLUS enumerations */
    /* start of Full PHIGS enumerations */
    PELEM_APPL_INT,
    PELEM_APPL_REAL,
    PELEM_CIRCLE3,
    PELEM_CIRCLE,
    PELEM_CIRCULAR_ARC3,
    PELEM_CIRCULAR_ARC,
    PELEM_ELLIPSE3,
    PELEM_ELLIPSE,
    PELEM_ELLIPTICAL_ARC3,
    PELEM_ELLIPTICAL_ARC,
    PELEM_FILL_CIRCLE3,
    PELEM_FILL_CIRCLE,
    PELEM_CIRCULAR_ARC_CLOSE3,
    PELEM_CIRCULAR_ARC_CLOSE,
    PELEM_FILL_ELLIPSE3,
    PELEM_FILL_ELLIPSE,
    PELEM_ELLIPTICAL_ARC_CLOSE3,
    PELEM_ELLIPTICAL_ARC_CLOSE,
    PELEM_HIGHL_IND,
    PELEM_LINETYPE_ADAPT,
    PELEM_LINETYPE_CONT,
    PELEM_LINETYPE_OFFSET,
    PELEM_LINECAP,
    PELEM_LINEJOIN,
    PELEM_LINEMITRE_LIMIT,
    PELEM_EDGETYPE_ADAPT,
    PELEM_EDGETYPE_CONT,
    PELEM_EDGETYPE_OFFSET,
    PELEM_EDGECAP,
    PELEM_EDGEJOIN,
    PELEM_EDGEMITRE_LIMIT,
    PELEM_HIGHL_METHOD,
    PELEM_TRANSPARENCY,
    PELEM_BACK_TRANSPARENCY,
    PELEM_ALPHA_SRC_SEL,
    PELEM_ALPHA_DATA_SEL_IND,
    PELEM_ACTIVE_TEXTURES,
    PELEM_BACK_ACTIVE_TEXTURES,
    PELEM_TEXTURE_PERSPECT_CORR,
    PELEM_TEXTURE_SAMPLING_FREQ,
    PELEM_TEXTURE_RES_OPT_HEUR,
    PELEM_SET_COND_FLAGS,
    PELEM_SET_COND_FLAGS_FROM_TESTS,
    PELEM_INST_STRUCT,
    PELEM_COND_EXEC_STRUCT,
    PELEM_COND_INST_STRUCT,
    PELEM_COND_RETURN,
    PELEM_COND_SKIP_ELEMS,
    PELEM_COND_SKIP_TO_LABEL,
    PELEM_PUSH_ST,
    PELEM_POP_ST
    /* end of Full PHIGS enumerations */

} Pelem_type;”
```

Page 175

The following text is inserted after the definition of Prgb:

```
“
 typedef struct {

    Pfloat     cieluv_x;       /* x coefficient                  */
    Pfloat     cieluv_y;       /* y coefficient                  */
    Pfloat     cieluv_y_lum;   /* y luminance                    */
    Pfloat     alpha;          /* alpha                          */

 } Pcieluva;
 typedef struct {

    Pfloat     hue;             /* hue                                 */
    Pfloat     lightness;       /* lightness                           */
    Pfloat     saturation;      /* saturation                          */
    Pfloat     alpha;           /* alpha                               */

 } Phlsa;

 typedef struct {

    Pfloat     hue;             /* hue                                 */
    Pfloat     saturation;      /* saturation                          */
    Pfloat     value;           /* value                               */
    Pfloat     alpha;           /* alpha                               */

 } Phsva;

 typedef struct {

    Pfloat     red;             /* red intensity                       */
    Pfloat     green;           /* green intensity                     */
    Pfloat     blue;            /* blue intensity                      */
    Pfloat     alpha;           /* alpha                               */

 } Prgba;”
```

The following text replaces the definition of Pcolr_rep*:*

“

```
 typedef union {

    /* start of Basic PHIGS union members */
    Prgb      rgb;       /* RGB colour specification                   */
    Pcieluv   cieluv;    /* CIE L*u*v* colour specification            */
    Phls      hls;       /* HLS colour specification                   */
    Phsv      hsv;       /* HSV colour specification                   */
    /* end of Basic PHIGS union members */
    /* start of Full PHIGS union members */
    Prgba     rgba;      /* RGB specification with alpha               */
    Pcieluva  cieluva;   /* CIE L*u*v* colour specfication with alpha */
    Phlsa     hlsa;      /* HLS colour specificaiton with alpha        */
    Phsva     hsva;      /* HSV colour specification with alpha        */
    /* end of Full PHIGS union members */
    Pdata     unsupp;    /* colour in an unsupported colour model      */
    int       impl_dep;  /* implementation defined                     */

 } Pcolr_rep;”
```

Page 187 through 193

The following type definitions replace the corresponding type definitions:

"

```
typedef struct {
   union Pchoice_pets {
      struct Pchoice_pet_other {
         Pint      unused;
      } pet_other; /* When no echo-specific data is required */
      struct Pchoice_pet_r1{
         Pint      imp_dep;             /* impl. dependent*/
      } pet_r1;  /* For PET 1 */
      struct Pchoice_pet_r2 {
         Pint         num_prompts;      /* number of prompts            */
         Ppr_switch   *prompts;         /* array of prompts             */
         Pint         imp_dep;          /* impl. defined                */
      } pet_r2;       /* For PET 2 */
      struct Pchoice_pet_r3 {
         Pint      num_strings;         /* number of choice strings     */
         char      **strings;           /* array of choice strings      */
         Pint      imp_dep;             /* impl. defined                */
      } pet_r3;  /* For PET 3 */
      struct Pchoice_pet_r4 {
         Pint      num_strings;         /* number of alternatives       */
         char      **strings;           /* array of strings             */
         Pint      imp_dep;             /* impl. defined                */
      } pet_r4;  /* For PET 4 */
      struct Pchoice_pet_r5 {
         Pint      struct_id;           /* structure identifier         */
         Pint      num_pick_ids;        /* number of alternatives       */
         Pint      *pick_ids;           /* array of pick identifiers    */
         Pint      imp_dep;             /* impl. defined                */
      } pet_r5;  /* For PET 5 */
      Pint         imp_dep;             /* data for impl. defined pets */
   } pets;
   union Pchoice_measure_data {
      struct Pchoice_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;
```

```
    union Pchoice_trigger_data {
       struct Pchoice_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
    union Pchoice_ack_data {
       struct Pchoice_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pchoice_data;

 typedef struct {
    union Pchoice3_pets {
       struct Pchoice3_pet_other {
          Pint      unused;
       } pet_other; /* When no echo-specific data is required */
       struct Pchoice3_pet_r1{
          Pint      imp_dep;            /* impl. dependent              */
       } pet_r1;  /* For PET 1 */
       struct Pchoice3_pet_r2 {
          Pint      num_prompts;        /* number of prompts            */
          Ppr_switch*prompts;           /* array of prompts             */
          Pint      imp_dep;            /* impl. defined                */
       } pet_r2;      /* For PET 2 */
       struct Pchoice3_pet_r3 {
          Pint      num_strings;        /* number of choice strings     */
          char      **strings;          /* array of choice strings      */
          Pint      imp_dep;            /* impl. defined                */
       } pet_r3;  /* For PET 3 */
       struct Pchoice3_pet_r4 {
          Pint      num_strings;        /* number of alternatives       */
          char      **strings;          /* array of strings             */
          Pint      imp_dep;            /* impl. defined                */
       } pet_r4;  /* For PET 4 */
       struct Pchoice3_pet_r5 {
          Pint      struct_id;          /* structure identifier         */
          Pint      num_pick_ids;       /* number of alternatives       */
          Pint      *pick_ids;          /* array of pick identifiers    */
          Pint      imp_dep;            /* impl. defined                */
       } pet_r5;  /* For PET 5 */
```

```
      Pint          imp_dep; /* data for impl. defined pets */
   } pets;
   union Pchoice3_measure_data {
      struct Pchoice_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;
   union Pchoice3_trigger_data {
      struct Pchoice3_trig_other {
         Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      Pint      imp_dep; /* data for impl. defined trigger processes */
   } trigger_data;
   union Pchoice3_ack_data {
      struct Pchoice3_ack_other {
         Pint      unused;
      } ack_other;  /* When no ack.-specific data is required */
      Pint      imp_dep; /* data for impl. defined ack. processes */
   } acknowledgement_data;
} Pchoice_data3;

typedef struct {
   union Ploc_pets {
      struct Ploc_pet_other {
         Pint      unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Ploc_pet_r1 {
         . . .                          /* impl. dependent           */
      } pet_r1;      /* Supports standard PET 1 */
      struct Ploc_pet_r2 {
         . . .                          /* impl. dependent           */
      } pet_r2;      /* Supports standard PET 2 */
      struct Ploc_pet_r3 {
         . . .                          /* impl. dependent           */
      } pet_r3;      /* Supports standard PET 3 */
      struct Ploc_pet_r4 {
         Pline_attrsline_attrs;         /* polyline attributes       */
         Pint      imp_dep;             /* impl. dependent           */
      } pet_r4;      /* Supports standard PET 4 */
      struct Ploc_pet_r5 {
         Pline_fill_ctrl_flagline_fill_ctrl_flag;/*ctrl. flag*/
         union Ploc_attrs {
            Pline_attrs   line_attrs; /* polyline attributes*/
            Pint_attrs    int_attrs;  /* interior attributes*/
            struct Ploc_fill_set {
               Pint_attrs  int_attrs;  /* interior attributes       */
               Pedge_attrs edge_attrs; /* edge attributes           */
            } fill_set;
         } attrs;
         Pint      imp_dep; /* data for impl. defined PET's */
      } pet_r5;      /* Supports standard PET 5 */
      struct Ploc_pet_r6 {
         . . .                        /* impl. dependent            */
      } pet_r6;      /* Supports standard PET 6 */
      Pint        imp_dep; /* data for impl. defined PET's */
   } pets;
   union Ploc_measure_data {
      struct Ploc_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;
   union Ploc_trigger_data{
      struct Ploc_trig_other {
         Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      Pint      imp_dep; /* data for impl. defined trigger processes */
   } trigger_data;
   union Ploc_ack_data {
      struct Ploc_ack_other {
         Pint      unused;
      } ack_other; /* When no ack.-specific data is required */
      Pint      imp_dep; /* data for impl. defined ack. processes */
   } acknowledgement_data;
} Ploc_data;

typedef struct {
   union Ploc3_pets {
      struct Ploc3_pet_other {
         Pint      unused;
      } pet_other;   /* When no echo-specific data is required */
      struct Ploc3_pet_r1 {
         . . .                         /* impl. dependent              */
      } pet_r1;      /* Supports standard PET 1 */
      struct Ploc3_pet_r2 {
         . . .                         /* impl. dependent              */
      } pet_r2;      /* Supports standard PET 2 */
      struct Ploc3_pet_r3 {
         . . .                         /* impl. dependent              */
      } pet_r3;      /* Supports standard PET 3 */
      struct Ploc3_pet_r4 {
         Pline_attrs  line_attrs;      /* polyline attributes          */
         Pint         imp_dep;         /* impl. dependent              */
      } pet_r4;      /* Supports standard PET 4 */
      struct Ploc3_pet_r5 {
         Pline_fill_ctrl_flag line_fill_ctrl_flag;  /* ctrl. flag      */
         union Ploc3_attrs {
            Pline_attrs   line_attrs;  /* polyline attributes          */
            Pint_attrs    int_attrs;   /* interior attributes          */
            struct Ploc3_fill_set {
               Pint_attrs   int_attrs;   /* interior attributes        */
               Pedge_attrs  edge_attrs;  /* edge attributes            */
            } fill_set;
         } attrs;
         Pint      imp_dep; /* data for impl. defined PET's */
      } pet_r5;      /* Supports standard PET 5 */
      struct Ploc3_pet_r6 {
         . . .                         /* impl. dependent              */
      } pet_r6;      /* Supports standard PET 6 */
      Pint      imp_dep; /* data for impl. defined PET's */
   } pets;
   union Ploc3_measure_data {
      struct Ploc3_meas_other {
         Pint      unused;
      } meas_other;  /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;
   union Ploc3_trigger_data {
      struct Ploc3_trig_other {
         Pint      unused;
      } trig_other;  /* When no trigger-specific data is required */
      Pint      imp_dep; /* data for impl. defined trigger processes */
   } trigger_data;
```

```
    union Ploc3_ack_data {
       struct Ploc3_ack_other {
          Pint      unused;
       } ack_other;   /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Ploc_data3;

 typedef struct {
    union Ppick_pets {
       struct Ppick_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Ppick_pet_r1 {
          Pint      imp_dep;            /* impl. dependent            */
       } pet_r1;  /* For PET 1 */
       Pint      imp_dep; /* data for impl. defined pets */
       struct Ppick_pet_r2 {
          ...                           /* implementation dependent    */
       } pet_r2;  /* For PET 2 */
       struct Ppick_pet_r3 {
          ...                           /* implementation dependent    */
       } pet_r3;  /* For PET 3 */
       struct Ppick_pet_r4 {
          Pint      highl_ind;          /* highlighting index          */
          Pint      imp_dep;            /* impl. dependent             */
       } pet_r4;  /* For PET 4 */
       struct Ppick_pet_r5 {
          Pint      highl_ind;          /* highlighting index          */
          Pint      imp_dep;            /* impl. dependent             */
       } pet_r5;  /* For PET 5 */
       struct Ppick_pet_r6 {
          Pint      highl_ind;          /* highlighting index          */
          Pint      imp_dep;            /* impl. dependent             */
       } pet_r6; /* For PET 6 */
       Pint         imp_dep;            /* implementation defined PET's*/
    } pets;
    union Ppick_measure_data {
       struct Ppick_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Ppick_trigger_data {
       struct Ppick_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined pick processes */
    } trigger_data;
    union Ppick_ack_data {
       struct Ppick_ack_other {
          Pint      unused;
       } ack_other; /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Ppick_data;

 typedef struct {
    union Ppick3_pets {
       struct Ppick3_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Ppick3_pet_r1 {
          Pint      imp_dep;            /* impl. dependent              */
       } pet_r1;  /* For PET 1 */
       struct Ppick3_pet_r2 {
          ...                           /* implementation dependent     */
       } pet_r2;  /* For PET 2 */
       struct Ppick3_pet_r3 {
          ...                           /* implementation dependent     */
       } pet_r3;  /* For PET 3 */
       struct Ppick3_pet_r4 {
          Pint      highl_ind;          /* highlighting index           */
          Pint      imp_dep;            /* impl. dependent              */
       } pet_r4;  /* For PET 4 */
       struct Ppick3_pet_r5 {
          Pint      highl_ind;          /* highlighting index           */
          Pint      imp_dep;            /* impl. dependent              */
       } pet_r5;  /* For PET 5 */
       struct Ppick3_pet_r6 {
          Pint      highl_ind;          /* highlighting index           */
          Pint      imp_dep;            /* impl. dependent              */
       } pet_r6;
       Pint      imp_dep; /* data for impl. defined pets */
    } pets;  /* For PET 6 */
    union Ppick3_measure_data {
       struct Ppick3_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Ppick3_trigger_data {
       struct Ppick3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined pick processes */
    } trigger_data;
    union Ppick3_ack_data {
       struct Ppick3_ack_other {
          Pint      unused;
       } ack_other; /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Ppick_data3;

  typedef struct {
    Pint                 in_buf_size;    /* input buffer size            */
    Pint                 init_pos;       /* initial editing position     */
    union Pstring_pets {
       struct Pstring_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstring_pet_r1{
          Pint      imp_dep;             /* impl. dependent              */
       } pet_r1;  /* For PET 1 */
       Pint      imp_dep; /* data for impl. defined pets */
    } pets;
    union Pstring_measure_data {
       struct Pstring_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Pstring_trigger_data {
       struct Pstring_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstring_ack_data {
       struct Pstring_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstring_data;

 typedef struct {
    Pint                in_buf_size;   /* input buffer size          */
    Pint                init_pos;      /* initial editing position   */
    union Pstring3_pets {
       struct Pstring_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstring3_pet_r1{
          Pint      imp_dep;           /* impl. dependent            */
       } pet_r1;  /* For PET 1 */
       Pint      imp_dep; /* data for impl. defined pets */
    } pets;
    union Pstring3_measure_data {
       struct Pstring3_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Pstring3_trigger_data {
       struct Pstring3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
```

```
   union Pstring3_ack_data {
      struct Pstring3_ack_other {
         Pint      unused;
      } ack_other;  /* When no ack.-specific data is required */
      Pint      imp_dep; /* data for impl. defined ack. processes */
   } acknowledgement_data;

} Pstring_data3;

typedef struct {
   Pint                buffer_size;   /* input buffer size */
   Pint                init_pos;      /* initial editing position */
   Pfloat              x_interval;    /* x trigger interval */
   Pfloat              y_interval;    /* y trigger interval */
   Pfloat              time_interval; /* time trigger interval */
   union {
      struct Pstroke_pet_other {
         Pint      unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Pstroke_pet_r1 {
         Pint      imp_dep;           /* impl. dependent            */
      } pet_r1;  /* For std PET 1 */
      struct Pstroke_pet_r2 {
         Pint      imp_dep;           /* impl. dependent            */
      } pet_r2;  /* For std PET 2 */
      struct Pstroke_pet_r3 {
         Pmarker_attrs  marker_attrs; /* marker attributes          */
         Pint           imp_dep;      /* impl. dependent            */
      } pet_r3;  /* For std PET 3 */
      struct Pstroke_pet_r4 {
         Pline_attrs line_attrs;      /* line attributes            */
         Pint        imp_dep;         /* impl. dependent            */
      } pet_r4;  /* For std PET 4 */
      Pint      imp_dep; /* data for impl. defined PET's */
         } pets;
   union Pstroke_measure_data {
      struct Pstroke_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;
```

```
    union Pstroke_trigger_data {
       struct Pstroke_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstroke_ack_data {
       struct Pstroke_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstroke_data;

 typedef struct {
    Pint                 in_buf_size;   /* input buffer size           */
    Pint                 init_pos;      /* initial editing position    */
    Pfloat               x_interval;    /* x trigger interval          */
    Pfloat               y_interval;    /* y trigger interval          */
    Pfloat               z_interval;    /* z trigger interval          */
    Pfloat               time_interval; /* time trigger interval       */
    union Pstroke3_pets {
       struct Pstroke3_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       struct Pstroke3_pet_r1 {
          Pint      imp_dep;            /* impl. dependent             */
       } pet_r1;  /* For std PET 1 */
       struct Pstroke3_pet_r2 {
          Pint      imp_dep;            /* impl. dependent             */
       } pet_r2;  /* For std PET 2 */
       struct Pstroke3_pet_r3 {
          Pmarker_attrs  marker_attrs; /* marker attributes           */
          Pint           imp_dep;      /* impl. dependent             */
       } pet_r3;  /* For std PET 3 */
       struct Pstroke3_pet_r4 {
          Pline_attrs   line_attrs;     /* line attributes             */
          Pint          imp_dep;        /* impl. dependent             */
       } pet_r4;  /* For std PET 4 */
       Pint      imp_dep; /* data for impl. defined PET's */
    } pets;
    union Pstroke3_measure_data {
       struct Pstroke3_meas_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Pstroke3_trigger_data {
       struct Pstroke3_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
    union Pstroke3_ack_data {
       struct Pstroke3_ack_other {
          Pint      unused;
       } ack_other;  /* When no ack.-specific data is required */
       Pint      imp_dep; /* data for impl. defined ack. processes */
    } acknowledgement_data;
 } Pstroke_data3;

 typedef struct {
    Pfloat              low;            /* Low value of valuator range */
       Pfloat      high;                /* High value of valuator range*/
    union Pval_pets {
       struct Pval_pet_other {
          Pint      unused;
       } pet_other;  /* When no echo-specific data is required */
       Pint      imp_dep; /* data for impl. defined pets */
    } pets;
    union Pval_measure_data {
       struct Pval_measure_other {
          Pint      unused;
       } meas_other; /* When no measure-specific data is required */
       Pint      imp_dep; /* data for impl. defined measure processes */
    } measure_data;
    union Pval_trigger_data {
       struct Pval_trig_other {
          Pint      unused;
       } trig_other; /* When no trigger-specific data is required */
       Pint      imp_dep; /* data for impl. defined trigger processes */
    } trigger_data;
```

```
      union Pval_ack_data {
         struct Pval_ack_other {
            Pint      unused;
         } ack_other; /* When no ack.-specific data is required */
         Pint      imp_dep; /* data for impl. defined ack. processes */
      } acknowledgement_data;

   } Pval_data;

   typedef struct {

      Pfloat              low;           /* Low value of valuator range */
         Pfloat      high;               /* High value of valuator range*/

      union Pval3_pets {
         struct Pval3_pet_other {
            Pint      unused;
         } pet_other;  /* When no echo-specific data is required */
         Pint      imp_dep; /* data for impl. defined pets */
      } pets;

      union Pval3_measure_data {
         struct Pval3_measure_other {
            Pint      unused;
         } meas_other; /* When no measure-specific data is required */
         Pint      imp_dep; /* data for impl. defined measure processes */
      } measure_data;

      union Pval3_trigger_data {
         struct Pval3_trig_other {
            Pint      unused;
         } trig_other; /* When no trigger-specific data is required */
         Pint      imp_dep; /* data for impl. defined trigger processes */
      } trigger_data;

      union Pval3_ack_data {
         struct Pval3_ack_other {
            Pint      unused;
         } ack_other; /* When no ack.-specific data is required */
         Pint      imp_dep; /* data for impl. defined ack. processes */
      } acknowledgement_data;

   } Pval_data3;”
```

Page 194 (Page 105 Amendment 1)

The following text replaces the definition of Pcolr_rep_ptr*:*

```
“
 typedef union {
    /* start of PHIGS PLUS union members */
    Prgb        *rgb;          /* pointer to RGB colour values */
    Pcieluv     *cieluv;       /* pointer to CIELUV pointer values */
    Phls        *hls;          /* pointer to HLS pointer values */
    Phsv        *hsv;          /* pointer to HSV pointer values */
    /* end of PHIGS PLUS union members */
    /* start of Full PHIGS union members */
    Prgba       *rgba;         /* pointer to RGBA colour values */
    Pcieluva    *cieluva;      /* pointer to CIELUVA pointer values */
    Phlsa       *hlsa;         /* pointer to HLSA pointer values */
    Phsva       *hsva;         /* pointer to HSVA pointer values */
    /* end of Full PHIGS union members */
    ...                        /* implementation defined */
 } Pcolr_rep_ptr;”
```

Page 194 (Page 118 Amendment 1)

The following text replaces the definition of Ppat_rep_plus*:*

```
“
 typedef struct {
    Pint            colr_type;        /* colour type                */
    Pcolrv_array    colrs;            /* colour array               */
 } Pgcolr_array;

 typedef Pgcolr_array Ppat_rep_plus;

 typedef struct {
    Pint            colr_type;        /* colour type                */
    Pint            num_arrays;       /* number of arrays in set    */
    Pcolrv_array    *colrs;           /* set of colour arrays       */
 } Pgcolr_array_set;
```

Page 194 (Page 122 Amendment 1)

The following text is inserted after the definition of Pws_st_tables_plus:

```
“
 typedef struct {
    Pdyn_mod            posting_status;/* posting status modification */
 } Pdyns_post_grps;
```

```
typedef enum {

 PACCESS_REFERENCED = 0,
 PACCESS_COPIED = 1

} Paccess_flag;

typedef enum {

   P_ACK_ACCEPTANCE = 0,
   P_ACK_NONACCEPTANCE = 1

} Packnowledgement_type;

typedef struct {

 Pack_type        type;
 Pint             process_id;

} Pack_process;

typedef struct {

 Pint             count;
 Pack_process     *procs;

} Pack_process_list;

typedef enum {

 PNOTASSOCIATED = 0,
 PASSOCIATED = 1

} Passoc_flag;

typedef struct {

   Pint      max_simul_trig,   /* max simul. trigger procs          */
   Pint      max_simul_echo,   /* max simul. echo procs             */
   Pint      max_simul_ack,    /* max simul. acknowledgement procs  */
   Pint_list meas_proc_ids,    /* available measure process ids     */
   Pint_list trig_proc_ids,    /* available trigger process ids     */
   Pint_list echo_proc_ids,    /* available echo process ids        */
   Pint_list ack_proc_ids      /* available acknowledgement         */
                               /* process ids                       */

} Patomic_lid_facs;

typedef enum {

   PBACKG_REDISPLAY_OFF = 0,
   PBACKG_REDISPLAY_ON = 1

} Pbackg_redisplay;
```

```
typedef enum {

  PBACKG_STYLE_TRANSPARENT = 0,
  PBACKG_STYLE_OPAQUE = 1

} Pbackg_style;

typedef enum {

  PFALSE = 0,
  PTRUE = 1

} Pboolean;

typedef enum {

  PBORDER_OFF = 0,
  PBORDER_ON = 1

} Pborder_indic;

typedef enum {

  PCLIPCOND_IS_NOT_CLIPPED = 0,
  PCLIPCOND_IS_PARTIALLY_CLIPPED = 1,
  PCLIPCOND_IS_FULLY_CLIPPED = 2

} Pclip_cond;

typedef enum {

    PCLOSURE_PIE = 0,
    PCLOSURE_SEGMENT = 1

} Pclosure;

union _Pin_class_data;

typedef struct _Pset_data {

    Pin_class        dev_class;
    Pint             num_in_set;
    _Pin_class_data  *data;

} Pset_data;
```

```
typedef struct {

   Pint      max_measures,            /* max instances               */
   Pint      max_simul_trig,          /* max simul. trigger procs    */
   Pint      max_simul_echo,          /* max simul. echo procs       */
   Pint      max_simul_ack,           /* max simul. ack. procs       */
   Pint_list loc_meas_proc_ids,       /* avail locator meas proc ids */
   Pint_list stroke_meas_proc_ids,    /* avail stroke meas proc ids  */
   Pint_list val_meas_proc_ids,       /* avail valuator meas proc ids*/
   Pint_list choice_meas_proc_ids,    /* avail choice meas proc ids  */
   Pint_list pick_meas_proc_ids,      /* avail pick meas proc ids    */
   Pint_list string_meas_proc_ids,    /* avail string meas proc ids  */
   Pint_list set_meas_proc_ids,       /* avail set meas proc ids     */
   Pint_list composite_meas_proc_ids, /* avail compos meas proc ids  */
   Pint_list trig_proc_ids,           /* avail trigger proc ids      */
   Pint_list loc_echo_proc_ids,       /* avail locator echo proc ids */
   Pint_list stroke_echo_proc_ids,    /* avail stroke echo proc ids  */
   Pint_list val_echo_proc_ids,       /* avail valuator echo proc ids*/
   Pint_list choice_echo_proc_ids,    /* avail choice echo proc ids  */
   Pint_list pick_echo_proc_ids,      /* avail pick echo proc ids    */
   Pint_list string_echo_proc_ids,    /* avail string echo proc ids  */
   Pint_list set_echo_proc_ids,       /* avail set echo proc ids     */
   Pint_list comp_echo_proc_ids,      /* avail compos echo proc ids  */
   Pint_list ack_proc_ids             /* avail ack. proc ids         */

} Pset_facs;


typedef struct _Pcomposite_individual {

   Pin_class        class;
   _Pin_class_data  *data;

} Pcomposite_individual;


typedef struct {

   Pint                     count;
   Pcomposite_individual   *individual;

} Pcomposite_data;
```

```
typedef struct {
   Pint      max_simul_meas,       /* max simul. measure procs       */
   Pint      max_simul_trig,       /* max simul. trigger procs       */
   Pint      max_simul_echo,       /* max simul. echo procs          */
   Pint      max_simul_ack,        /* max simul. acknowledgement procs
                                      */
   Pint_list loc_meas_proc_ids,    /* avail. locator measure proc. ids
                                      */
   Pint_list stroke_meas_proc_ids, /* avail. stroke measure proc. ids
                                      */
   Pint_list val_meas_proc_ids,    /* avail. valuator measure proc. ids
                                      */
   Pint_list choice_meas_proc_ids, /* avail. choice measure proc. ids
                                      */
   Pint_list pick_meas_proc_ids,   /* avail. pick measure proc. ids  */
   Pint_list string_meas_proc_ids, /* avail. string measure proc. ids
                                      */
   Pint_list set_meas_proc_ids,    /* avail. set measure proc. ids   */
   Pint_list composite_meas_proc_ids,/*avail. composite meas. proc. ids
                                      */
   Pint_list trig_proc_ids,        /* available trigger process ids  */
   Pint_list loc_echo_proc_ids,    /* available locator echo proc. ids
                                      */
   Pint_list stroke_echo_proc_ids, /* available stroke echo proc. ids
                                      */
   Pint_list val_echo_proc_ids,    /* available valuator echo proc. ids
                                      */
   Pint_list choice_echo_proc_ids, /* available choice echo proc. ids
                                      */
   Pint_list pick_echo_proc_ids,   /* available pick echo proc. ids  */
   Pint_list string_echo_proc_ids, /* available string echo proc. ids
                                      */
   Pint_list set_echo_proc_ids,    /* available set echo proc. ids   */
   Pint_list comp_echo_proc_ids,   /* available composite echo proc. ids
                                      */
   Pint_list ack_proc_ids          /* available ack. proc. ids       */
} Pcomposite_facs;
```

```
typedef union _Pin_class_data {

   Pint                 unused;
   Ploc_data            loc;
   Ploc_data3           loc3;
   Pstroke_data         stroke;
   Pstroke_data3        stroke3;
   Pval_data            val;
   Pval_data3           val3;
   Pchoice_data         choice;
   Pchoice_data3        choice3;
   Ppick_data           pick;
   Ppick_data3          pick3;
   Pstring_data         string;
   Pstring_data3        string3;
   Pset_data            set;
   Pcomposite_data      composite;
   ...                                 /* other classes of input data */

} Pin_class_data;


typedef enum {

  PCOMP_IN_STATUS_NONE = 0,
  PCOMP_IN_STATUS_OK = 1,
  PCOMP_IN_STATUS_INCOMPLETE = 2

} Pcomp_in_style;


typedef struct {

   Pcond_flag_mask      enable_mask;   /* condition flag enable mask  */
   Pcond_flag_mask      disable_mask;  /* condition flag disable mask */

} Pcond_flags;


typedef struct {

   Ptest                test;          /* test operation              */
   Pcond_flags          mask;          /* condition test mask         */

} Pcond_flag_test;


typedef struct {

   Pint                 num_tests;     /* number of condition test    */
   Pcond_flag_test      *tests;        /* list of condition flag tests*/

} Pcond_test_list;
```

```
typedef struct {

   Pint       num_invis_filters;/* number of avail. invisibility filters
                                   */
   Pint       num_highl_filters;/* number of avail. highlighting filters
                                   */
   Pint       num_pick_filters; /* number of avail. pick filters    */
   Pint       num_appl_filters; /* number of avail. application filters
                                   */
   Pint       num_appl_integers;/* number of application integer    */
                                /* values supported                 */
   Pint       num_appl_reals;   /* number of application real values */
                                /* supported                        */
   Pint_list  test_methods;     /* list of test methods available   */

} Pcond_trav_facs;

typedef struct {

   Pint             name_size;       /* size of name                  */
   char             *name;           /* configuration setting name    */

} Pconfig_name;

typedef struct {

   Pint             list_size;       /* number of names               */
   Pconfig_name     *name;           /* configuration name list       */

} Pconfig_name_list;

typedef struct {

   Pint            num_supported;   /* number of simultaneously supported
                                       */
                                    /* device coordinate clip regions */

} Pconfig_setting_facs;

typedef union {

   Pint        int_value;            /* integer configuration value   */
   Pfloat      real_value;           /* real configuration value      */
   char        *string_value;        /* string configuration value    */

} Pconfig_value;

typedef struct {

   char             *name;           /* configuration setting name    */
   Pconfig_value    *value;          /* configuration setting value   */

} Pconfig_settings;
```

```
typedef struct {

   Pint      num_supported;    /* number of simultaneously supported */
                               /* device coordinate clip regions     */

} Pdc_clip_facs;


typedef struct {

   unsigned int mode:1;
   unsigned int ref_planes:1;
   unsigned int scale:1;
   unsigned int gcolr:1;
   unsigned int :28;

} Pdepth_cue_mask;


typedef enum {

   PDIM_INTERPRET_NONE = 0,
   PDIM_INTERPRET_STATE = 1,
   PDIM_INTERPRET_ALL = 2

} Pdi_mode;


typedef struct {

   Pint   num_trav_procs;       /* number of simultaneously posted */
                                /* traversal processes             */
   Pint   num_rend_algorithms;  /* number of multi-pass algorithms */
                                /* supported for rendering         */
   Pint   num_pick_algorithms;  /* number of multi-pass algortihms */
                                /* supported for picking           */

} Pdi_trav_facs;


typedef struct {

  Pin_class         dev_class;       /* input class*/
  Pint              instance;        /* i = ith of class in LID def.*/

} Pecho_measure_proc;


typedef struct {

  Pint                pet;
  Pecho_measure_proc *measures_echoed;
  Pint                process_id;

} Pecho_process;
```

```
typedef struct {

 Pint             count;
 Pecho_process    *procs;

} Pecho_process_list;


typedef struct {

   Pedge_flag         flag;          /* edge flag                   */
   Pint               type;          /* edgetype                    */
   Pfloat             width;         /* edgewidth scale factor      */
   Pgolr              colr;          /* edge colour                 */
   Plinetype_adapt    adapt;         /* edgetype adaptability       */
   Plinetype_cont     cont;          /* edgetype continuity         */
   Pfloat             offset;        /* edgetype offset             */
   Plinecap           cap;           /* edgecap                     */
   Plinejoin          join;          /* edgejoin                    */
   Pfloat             limit;         /* edgemitre limit             */

} Pedge_bundle_full;


typedef struct {

   unsigned int flag:1;
   unsigned int type:1;
   unsigned int width:1;
   unsigned int colr:1;
   unsigned int adapt:1;
   unsigned int cont:1;
   unsigned int offset:1;
   unsigned int cap:1;
   unsigned int join:1;
   unsigned int limit:1;
   unsigned int :22;

} Pedge_mask;


typedef enum {

   PEDTA_EXACT = 0,
   PEDTA_ADAPT = 1

} Pedgetype_adapt;


typedef enum {

   PEDC_BUTT = 0,
   PEDC_ROUND = 1,
   PEDC_SQUARE = 2

} Pedgecap;
```

```
typedef enum {

   PEDTC_CONTINUOUS = 0,
   PEDTC_RESTART = 1

} Pedgetype_cont;

typedef enum {

   PEDJ_FLAT = 0,
   PEDJ_MITRE = 1,
   PEDJ_ROUND = 2,
   PEDJ_BEVEL = 3

} Pedgejoin;

typedef struct {

   Pint_list pattern_types;      /* list of supported pattern types   */
   Pint_list image_spec_methods;/* list of supported image           */
                                /* specifcation methods             */

} Pext_pat_facs;

typedef enum {

   PERASE_OFF = 0,
   PERASE_ON = 1

} Perase;

typedef struct {

   Pint_list image_spec_methods;/* list of supported image           */
                                /* specification methods            */
   Pint      num_def_ids;       /* number of definable image         */
                                /* resource identifiers             */
   Pint      num_predef_ids;    /* number of predefined image        */
                                /* resources                        */

} Pimage_res_facs;

typedef struct {

   unsigned int style:1;
   unsigned int style_ind:1;
   unsigned int colr:1;
   unsigned int shad_method:1;
   unsigned int :28;

} Pinterior_mask;
```

```
typedef struct {

  Pint            num_limits;        /* number of limits          */
  Plimit          *limits;           /* list of limits            */

} Plimit_list;


typedef struct {

  Pint            num_limits;        /* number of limits          */
  Plimit3         *limits;           /* list of limits            */

} Plimit3_list;


typedef struct {

    Pint    light_ws_id;             /* workstation id for view   */
    Pint    light_src_ind;           /* view index                */
    Pint    light_action_type;       /* view action type          */
    Pint    transfer_type;           /* measure transfer type     */

} Plight_attach;


typedef enum {

    PLNC_BUTT = 0,
    PLNC_ROUND = 1,
    PLNC_SQUARE = 2

} Plinecap;


typedef enum {

    PLNJ_FLAT = 0,
    PLNJ_MITRE = 1,
    PLNJ_ROUND = 2,
    PLNJ_BEVEL = 3

} Plinejoin;


typedef enum {

    PLNTA_EXACT = 0,
    PLNTA_ADAPT = 1

} Plinetype_adapt;


typedef enum {

    PLNTC_CONTINUOUS = 0,
    PLNTC_RESTART = 1

} Plinetype_cont;
```

```
typedef struct {

   Pint                 type;          /* linetype                  */
   Pfloat               width;         /* linewidth scale factor    */
   Pgolr                colr;          /* polyline colour           */
   Pint                 shad_method;   /* polyline shading method   */
   Pcurve_approx_crit_data
                        curve_approx_crit_data;
                                       /* curve approx. criteria    */
   Plinetype_adapt      adapt;         /* linetype adaptability     */
   Plinetype_cont       cont;          /* linetype continuity       */
   Pfloat               offset;        /* linetype offset           */
   Plinecap             cap;           /* linecap                   */
   Plinejoin            join;          /* linejoin                  */
   Pfloat               limit;         /* linemitre limit           */

} Pline_bundle_full;

typedef struct {

   Pint      num_definable;       /* number definable linetypes      */
   Pint      num_predefined;      /* number predefined linetypes     */
   Pint      max_dash_segments;   /* maximum dash segments supported */
   Pfloat    offset;              /* offset to definable linetypes   */
   Pint_list predef_def_linetypes /* list of predef. definable linetypes
                                   */

} Plinetype_def_facs;

typedef struct {

   Pfloat            lum_value;        /* luminance_value           */
   Pfloat            alpha_value;      /* alpha value               */

} Plum_alpha_pair;

typedef struct {

   Pint_size         dims;        /* array dimensions                 */
   Plum_alpha_pair   *values;     /* 2D array of luminance/alpha pairs */

} Plum_alpha_array;

typedef struct {

   Pint              num_arrays;  /* number of arrays in set          */
   Plum_alpha_array  *sets;       /* set of 2D luminance/alpha arrays */

} Plum_alpha_array_set;
```

```
typedef struct {

   Pint                  shape_type;     /* shape type                    */

   union _Pshape_data {
      Ppoint_list  polyline;             /* for PSHAPE_POLYLINE           */
      Ppoint_list  fill_area;            /* for PSHAPE_FILL_AREA          */
      Ppoint_list  convex_fill_area;     /* for PSHAPE_CONVEX_FILL_AREA */
      ...                                /* implementation defined        */
   } shape_data;                         /* marker shape data             */

} Pmarker_data;

typedef struct {

   Pint               number_shapes;     /* number of shapes              */
   Pmarker_data       *marker;           /* array of shape descriptors    */

} Pmarker_desc;

typedef struct {

   Pint        num_definable;       /* number definable marker types */
   Pint        num_predefined;      /* number predefined marker types */
   Pfloat      offset;              /* offset to definable linetypes */
   Pint        max_allowable_pts;   /* maximum allowable points      */
   Pint_list   shape_types;         /* list of available shape types */
   Pint_list   predef_marker_types  /* list of predefined marker types
                                        */

} Pmarker_type_def_facs;

typedef union _Pin_class_measure;/*  used for defining _Pmeasure    */
                                /*   and _Pset_measure              */

typedef struct _Pmeasure {

   Pin_class            class;
   _Pin_class_measure   *data;

} Pmeasure;

typedef struct _Pset_measure {

   Pin_class            dev_class;      /* input class of set            */
   Pint                 set_size;       /* size of measure set           */
   _Pin_class_measure   *measure_set;   /* array of measures of same     */
                                        /* class                         */

} Pset_measure;
```

```
typedef struct _Pcomposite_measure {

   Pint         count;                 /* number of measure components*/
   Pmeasure     *measures;             /* list of measure components  */

} Pcomposite_measure;


typedef union _Pin_class_measure {

   struct {
      Pint         view_ind;
      Ppoint       loc_pos;
   } loc;

   struct {
      Pint         view_ind;
      Ppoint3      loc_pos;
   } loc3;

   struct {
      Pint         view_ind;
      Ppoint_list  points;
   } stroke;

   struct {
      Pint         view_ind;
      Ppoint_list3 points;
   } stroke3;

   Pfloat              val;

   struct {
      Pin_status   istat;
      Pint         selection;
   } choice;

   struct {
      Pin_status   istat;
      Ppick_path   *path;
   } pick;

   char                 *string;

   Pset_measure         *set;

   Pcomposite_measure   *composite;

   ...                 /* other classes which may be added */

} Pin_class_measure;
```

```
typedef struct {

  Pin_class        dev_class;
  Pint             process_id;

} Pmeas_process;


typedef struct _Pmeas_process_list {

    Pint               count;           /* number of process in list   */
    Pmeas_process      *measure_procs;  /* list of measure processes   */

} Pmeas_process_list;


typedef struct {

  Pint             set_maxsize;         /* maximum set size            */
  Pmeas_process    process_id;          /* measure process identifier  */

} Pmeas_process_set;


typedef enum {

    PNA_IN_STATUS_NONE = 0,
    PNA_IN_STATUS_INCOMPLETE = 1,
    PNA_IN_STATUS_OK = 2

} Pna_in_status;


typedef struct {

    Pint         set;             /* number available set LIDs         */
    Pint         composite;       /* number available composite LIDs   */

} Pnum_na_in;


typedef struct {

    unsigned int pat_bundle:1;
    unsigned int pat_rep_plus:1;
    unsigned int ext_pat_rep:1;
    unsigned int :29

} Ppattern_mask;


typedef struct _Ppick_path_list {

    Pint         num_paths;       /* number paths in list              */
    Ppick_path   *paths;          /* array of pick paths               */

} Ppick_path_list;
```

```
typedef struct {
   Pint                type;           /* pattern type */
   union _Ppattern_data {
      struct Ppat_colr_ind_array {
         Ppat_rep  pat_rep;            /* pattern representation      */
      } pat_colr_ind_array;
      struct Ppat_colr_array {
         Ppat_rep_pluspat_rep_plus;    /* pattern representation plus */
      } pat_colr_array;
      struct Ppat_image_res {
         Pint      image_res_id;       /* pattern image resource      */
      } pat_image_res;
      ...                              /* implementation defined      */
   } data;                             /* pattern data */
} Ppattern;

typedef enum {
   PPICK_STAT_FALSE = 0,
   PPICK_STAT_TRUE = 1
} Ppick_stat;

typedef struct {
   Pint                pick_type;
   union _Ppick_type_data {
      Pint          unused;            /* workstation-dependent       */
      Pfloat_size3 rect_aperture;      /* 3D rectangular aperture     */
      Pfloat_size3 vis_rect_aperture;  /* 3D rectangular aperture     */
                                       /* visible primitives only     */
      Ppoint_list  user_aperture;      /* user-specified aperture     */
      . . .        /* data for impl. defined pick types */
   } data;
} Ppick_type_data;

typedef enum {
   PPICT_STAT_COMP = 0,
   PPICT_STAT_INCOMP = 1
} Ppict_stat;
```

```
typedef struct {

   unsigned int type:1;
   unsigned int width:1;
   unsigned int colr:1;
   unsigned int shad_method:1;
   unsigned int curve_approx_crit_data:1;
   unsigned int adapt:1;
   unsigned int cont:1;
   unsigned int offset:1;
   unsigned int cap:1;
   unsigned int join:1;
   unsigned int limit:1;
   unsigned int :21;

} Ppolyline_mask;


typedef struct {

   unsigned int type:1;
   unsigned int size:1;
   unsigned int colr:1;
   unsigned int :29;

} Ppolymarker_mask;


typedef enum {

   PPOSTED_FALSE = 0,
   PPOSTED_TRUE = 1

} Pposted_ind;


typedef struct {

   Pint      num_def_grps;    /* number of definable posting groups   */
   Pint      num_predef_grps; /* number of predefined posting groups */
   Pint_list predef_grps;     /* list of predefined posting groups   */
   Pint_list backg_methods;   /* list of supported posting group     */
                              /* background methods                  */

} Ppost_grp_facs;


typedef enum {

   PPGSTAT_INACTIVE = 0,
   PPGSTAT_ACTIVE = 1

} Ppost_grp_status;
```

```
typedef struct {

   unsigned int refl_model:1;
   unsigned int data:1;
   unsigned int :30;

} Prefl_mask;

typedef enum {

   PREF_EXECUTE = 0,
   PREF_INSTANCE = 1

} Pref_type;

typedef enum {

  PSCALE_NOT_SCALABLE,
  PSCALE_SCALABLE

} Pscale_flag;

typedef enum {

   PSI_NOT_SUPPORTED = 0,
   PSI_SUPPORTED = 1

} Psupport_indication;

typedef enum {

   PTARG_DATA_ST_ABSENT = 0,
   PTARG_DATA_ST_PRESENT = 1

} Ptarg_data_st;

typedef enum {

   PTARG_EMPTY_ST_EMPTY = 0,
   PTARG_EMPTY_ST_NOT_EMPTY = 1

} Ptarg_empty_st;

typedef struct {

   Pint            num_avail_targets; /* number of available targets */

} Ptarget_facs;
```

```
typedef enum {

   PBASE_TARG = 0,
   PREND_TARG = 1,
   PDISP_TARG = 2

} Ptarg_type;

typedef struct {

   Pint               num_targ_types;/* number of target types      */
   Ptarg_type         *targ_types;   /* list of target types        */

} Ptarg_type_list;

typedef struct {

  Ptarg_type      ref_targ;          /* reference target            */
  Pint            offset;            /* offset from reference target*/

} Ptarg_addr;

typedef struct {

      Pint        type;              /* target operation type       */

   union _Ptarget_op_data {
      struct Prend_targ {
         Ptarg_addrtarg_addr;        /* target address              */
      } rend_targ;
      struct Pdisp_targ {
         Ptarg_addrtarg_addr;        /* target address              */
      } disp_targ;
      struct Pclear_targ {
         Ptarg_addrtarg_addr;        /* target address              */
      } clear_targ;
      struct Pcopy_targ {
         Ptarg_addrsrc_targ_addr;    /* source target address       */
         Ptarg_addrdest_targ_addr;   /* destination target address  */
      } copy_targ;
      ...                            /* implementation defined      */
   } data;                           /* target operation data       */

} Ptarg_op;

typedef struct {

  Pint            num_ops;           /* number of target operations */
  Ptarg_op        *targ_ops;         /* list of target operations   */

} Ptarg_op_list;
```

```
typedef enum {

   PTRL_EQUAL = 0,
   PTRL_NOT_EQUAL = 1,
   PTRL_GREATER = 2,
   PTRL_LESS = 3,
   PTRL_GREATER_OR_EQUAL = 4,
   PTRL_LESS_OR_EQUAL = 5

} Ptest_comp_rel;


typedef enum {

   PTLG_BITWISE_AND = 1,
   PTLG_BITWISE_OR = 2,
   PTLG_BITWISE_XOR = 3

} Ptest_comp_logical;


typedef struct {

   unsigned int font:1;
   unsigned int precision:1;
   unsigned int char_expan:1;
   unsigned int char_space:1;
   unsigned int colr:1;
   unsigned int :27;

} Ptext_mask;


typedef struct {

  Pint             id;              /* traversal resource identifier*/
  Pint             type;            /* traversal resource type*/

} Ptrav_res;


typedef struct {

  Pint             num_trav_res;    /* number of traversal resources*/
  Ptrav_res        *trav_res;       /* list of traversal resources*/

} Ptrav_res_list;
```

```
typedef struct {

   Ptrav_res_list avail_trav_res;/* list of avail trav. resources   */

} Ptrav_res_facs;

typedef enum {

   PTRAV_RENDER = 0,
   PTRAV_PICK = 1

} Ptrav_type;

typedef enum {

   PTRIGGER_BREAK = 0,
   PTRIGGER_SELECT = 1,
   PTRIGGER_EVENT = 2,
   PTRIGGER_DELETE = 3

} Ptrig_type;

typedef struct {

  Ptrig_type       type;
  Pint             process_id;

} Ptrig_process;

typedef struct {

  Pint             count;
  Ptrig_process    *procs;

} Ptrig_process_list;

typedef struct {

   Pint     view_ws_id;              /* workstation id for view     */
   Pint     view_ind;                /* view index                  */
   Pint     view_action_type;        /* view action type            */
   Pint     transfer_type;           /* measure transfer type       */
   Pint     view_upd_method;         /* view update method          */

} Pview_attach;
```

```
typedef enum {

   PVIEW_ST_CORRESPOND = 0,
   PVIEW_ST_DIFFERENT = 1

} Pview_status;

typedef enum {

   PWATCH_OFF = 0,
   PWATCH_ON = 1

} Pwatch_enable;”

typedef struct {

  Pint_size        dims;              /* float array dimensions      */
  Pfloat           *floats;           /* float array                 */

} Pfloat_array;

typedef struct {

  Pint             num_arrays;      /* number of float arrays in set*/
  Pfloat_array     *floats;         /* float arrays*/

} Pfloat_array_set;

typedef struct {

   Pint image_spec_method;

   union _Pimage_data {
      struct Puncomp_colr_ind_array {
         Pint      *colr_ind_array;   /* uncompressed array of       */
                                      /* colour indices              */
      } uncomp_colr_ind_array;
      struct Puncomp_colr_array {
         Pgcolr_array  *colr_array;   /* uncompressed array of       */
                                      /*  colours                    */
      } uncomp_colr_array;
      struct Pwindow_sys_bitmap {
         Pint          handle;        /* bitmap handle               */
      } window_sys_bitmap;
      struct Pvideo_sig_chan_id {
         Pint          chan_id;       /* video signal channel id     */
      } video_sig_chan_id;
      struct Pluminance {
         Pfloat_array  *lum_values;   /* uncompressed array of*/
                                      /* luminance values*/
      } luminance;
```

```
        struct Pluminance_alpha {
           Plum_alpha_array  *lum_values;  /* uncompressed array of    */
                                           /* luminance values         */
        } luminance_alpha;
        struct Pmipmap_colrs {
           Pgcolr_array_set  *colr_arrays; /* set of uncompressed      */
                                           /* colour arrays            */
        } mipmap_colrs;
        struct Pmipmap_luminance {
           Pfloat_array_set  *lum_values;  /* set of uncompressed      */
                                           /* luminance arrays         */
        } mipmap_luminance;
        struct Pmipmap_luminance_alpha {
           Plum_alpha_array_set *lum_values;  /* set of uncompressed   */
                                              /* luminance/alpha arrays*/
        } mipmap_luminance_alpha;
        int       impl_def            /* implementation defined methods */
     } image_data;

  } Pimage_res;

  typedef struct {
     Pint                method;        /* background method           */
     union Pbackg_method_data {
        struct Pcolr_table_0 {
           Pint      unused;            /* unused                      */
        } colr_table_0;
        struct Pcolr_ind {
           Pint      colr_ind;          /* colour index                */
        } colr_ind;
        struct Pcolr {
           Pcolr_rep colr_rep;          /* target address              */
        } colr;
        struct Pimage_resource {
           Pint      image_resource_id; /* image resource identifier   */
        } image_resource;
        ...                             /* implementation defined      */
     } data;
  } Pbackg_method;
```

```
typedef struct {

   Pint                type;          /* trav. res. operation type   */

   union Ptrav_res_op_data {
      struct Ptrav_res_clear {
         Pint      res_id;            /* traversal resource id.      */
      } trav_res_clear;
      struct Ptrav_res_copy {
         Pint      src_res_id;        /* source traversal res. id.   */
         Pint      dest_res_id;       /* destination trav. res. id.  */
      } trav_res_copy;
      ...                             /* implementation defined      */
   } data;                            /* trav. res. operation data   */

} Ptrav_res_op;


typedef struct {

   union Pdi_pick_pets {
      struct Pdi_pick_pet_other {
         Pint      unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Pdi_pick_pet_r1 {
         Pint      imp_dep;           /* impl. dependent             */
      } pet_r1;  /* For PET 1 */
      ...                             /* implementation defined PET's*/
   } pets;

   union Pdi_pick_measure_data {
      struct Pdi_pick_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;

   union Pdi_pick_trigger_data {
      struct Pdi_pick_trig_other {
         Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      Pint      imp_dep; /* data for impl. defined di_pick processes */
   } trigger_data;

   union Pdi_pick_ack_data {
      struct Pdi_pick_ack_other {
         Pint      unused;
      } ack_other; /* When no ack.-specific data is required */
      Pint      imp_dep; /* data for impl. defined ack. processes */
   } acknowledgement_data;

} Pdi_pick_data;

typedef struct {

   union Pdi_pick3_pets {
      struct Pdi_pick3_pet_other {
         Pint      unused;
      } pet_other;  /* When no echo-specific data is required */
      struct Pdi_pick3_pet_r1 {
         Pint      imp_dep;           /* impl. dependent           */
      } pet_r1;  /* For PET 1 */
      Pint      imp_dep; /* data for impl. defined pets */
   } pets;

   union Pdi_pick3_measure_data {
      struct Pdi_pick3_meas_other {
         Pint      unused;
      } meas_other; /* When no measure-specific data is required */
      Pint      imp_dep; /* data for impl. defined measure processes */
   } measure_data;

   union Pdi_pick3_trigger_data {
      struct Pdi_pick3_trig_other {
         Pint      unused;
      } trig_other; /* When no trigger-specific data is required */
      Pint      imp_dep; /* data for impl. defined di_pick processes */
   } trigger_data;

   union Pdi_pick3_ack_data {
      struct Pdi_pick3_ack_other {
         Pint      unused;
      } ack_other; /* When no ack.-specific data is required */
      Pint      imp_dep; /* data for impl. defined ack. processes */
   } acknowledgement_data;

} Pdi_pick_data3;

typedef struct {

   Pint_list  alpha_srcs;     /* list available alpha sources       */
   Pint_list  transp_modes;   /* list transparency modes supported  */

} Palpha_facs;
```

```
typedef struct {

 Pdyn_mod    texture_rep;         /* texture representation         */

} Pdyns_ws_attrs_texture;


typedef struct {

 Pint      max_dims[3];        /* max dimensions mipmap base level  */
 Pboolean  equal_dims_req;     /* equal mipmap dimensions required  */
 Pboolean  power_2_dims_req;   /* power of 2 mipmap dimensions req. */

} Pmipmap_facs;


typedef enum {

   PPERSP_CORR_NONE = 0,
   PPERSP_CORR_AT_VERTICES = 1,
   PPERSP_CORR_INTERIOR = 2

} Pperspect_corr;


typedef struct {

   Pint_list  avail_pick_types;     /* list available pick types     */
   Pint_list  avail_echo_types;     /* list of available echo types  */

} Ppick_mapping_facs;


typedef enum {

 PRES_HINTS_UTILIZED_NO,
 PRES_HINTS_UTILIZED_YES

} Pres_hints;


typedef struct {

 Pint      image_res_id;        /* texture image resource identifier */
 Pint      rendering_order;     /* rendering order                   */

} Ptexture_binding;


typedef struct {

   Pint           method;          /* composition method             */

   union {
      Pint          unused;          /* used by methods 1, 2, and 4    */
      struct {                       /* used by method 3               */
       Pgcolr       environ;         /* environment colour             */
       Pint         gamma_chan_sel;  /* gamma channel selector         */
      } blend_env;
      Pgcolr      background_colr;/* used by method 5              */
      ...                         /* implementation defined        */
   } data;

} Ptexture_compos;

typedef struct {

 Pint_list  coord_srcs;       /* list avail. coordinate sources    */
 Pint_list  compos_methods;   /* list avail. composition methods   */
 Pint_list  min_methods;      /* list avail. minification methods  */
 Pint_list  mag_methods;      /* list avail. magnification methods */
 Pint_list  bound_cond;       /* list avail. boundary conditions   */
 Pint_list  clamp_methods;    /* list avail. boundary clamp methods */
 Pint_list  render_orders;    /* list avail. rendering orders      */
 Pint       num_pred_inds;    /* number predefined texture indices */

} Ptexture_facs;

typedef enum {

 PTUP_UTILIZED_NO = 0,
 PTUP_UTILIZED_YES = 1

} Ptup_util;

typedef struct {

 Pint       max_simul;        /* max number simultaneously         */
                              /* appliable textures                */
 Pint_list  perspect_corr;    /* list of available perspective     */
                              /* correction methods                */
 Pint_list  sampling_freqs;   /* list avail. sampling frequencies  */
 Pint_list  opt_hints;        /* list of available resource        */
                              /* optimization hints                */
 Ptup_util  usage;            /* utilization of texture usage      */
                              /* priorities                        */
 Pint_list  image_res_types;  /* list of image resource types      */
                              /* available for texture mapping     */

} Ptexture_map_facs;
```

```
typedef struct {

   unsigned int coord_src:1;
   unsigned int coord_src_data:1;
   unsigned int orientation:1;
   unsigned int compos_method:1;
   unsigned int compos_data:1;
   unsigned int min_method:1;
   unsigned int max_method:1;
   unsigned int bound_conds:1;
   unsigned int clamp_method:1
   unsigned int clamp_data:1
   unsigned int depth_sampl_hint:1
   unsigned int freq_wt_hints:1
   unsigned int image_res_id:1
   unsigned int rendering_order:1
   unsigned int :18

} Ptexture_mask;

typedef struct {

   Pint             coord_src;       /* coordinate source              */

   union Pcoord_src_data {           /* coordinate source data record  */
      Pint          unused;          /* used by coord sources 1 - 5    */
      Pint          coord_ind;       /* used by coord source 6         */
      Pmatrix3      refl_matrix;     /* used by coord sources 7 and 8  */
      ...                            /* implementation defined         */
   } data;

} Pcoord_src;

typedef struct {

  Pcoord_src        coord_src;          /* coordinate source and data  */
  Pmatrix3          orientation;        /* orientation matrix          */

} Ptexture_param;

typedef struct {

   Pint             method;          /* clamp method                   */

   union Pclamp_method_data {
      Pint          unused;          /* used by method 1               */
      Pgcolr        clamp_colr;      /* used by method 2               */
      ...                            /* implementation defined         */
   } data;

} Pclamp_method;
```

```
typedef struct {

  Pint           min_method;         /* minification method         */
  Pint           mag_method;         /* magnification method        */
  Pint           bound_conds[3];     /* boundary conditions         */
      Pclamp_methodclamp;            /* boundary clamp method & data*/
  Pfloat         depth_sampl_hint;   /* depth sampling hint         */
  Pfloat         freq_wt_hints[3];   /* frequency weight hints      */

} Ptexture_sampling;

typedef struct {

  Ptexture_param     param;          /* texture parametrization     */
  Ptexture_compos    compos;         /* texture composition         */
  Ptexture_sampling  sampling;       /* texture sampling            */
  Ptexture_binding   binding;        /* texture binding             */

} Ptexture_rep;

typedef struct {

   Pint              opt_hint;           /* optimization hint        */
   Pint_list         usage_priorities;   /* texture usage priorities */

} Ptexture_res_opt_heur;

typedef struct {

   Pint      max_simul_trig,   /* max simul. trigger procs           */
   Pint      max_simul_echo,   /* max simul. echo procs              */
   Pint      max_simul_ack,    /* max simul. acknowledgement procs   */
   Pint_list meas_proc_ids,    /* available measure process ids      */
   Pint_list trig_proc_ids,    /* available trigger process ids      */
   Pint_list echo_proc_ids,    /* available echo process ids         */
   Pint_list ack_proc_ids      /* available acknowledgement process ids
                                  */

} Pval_facs;

typedef struct {

   Pint   highl_rep;        /* max. # of highlighting table entries */

} Pws_st_tables_highl;

typedef struct {

   Pint   texture_rep;         /* max. # of texture table entries    */

} Pws_st_tables_texture;”
```

Page 195

The following text is inserted before the definition of pelem_data*:*

“
```
 typedef struct {

    Pint            num_pred;        /* number of predefined methods   */
    Pint_list       methods;         /* number of highl. methods supp. */
    Pint            max_highl_ind;   /* max highl. table indices       */

 } Phighl_facs;


 typedef struct {

    Pint                   method;

    union _Phighl_data {
       Pint         unused;
       Pfloat       blink_rate;
       Pint         colr_ind;
       Pgcolr       colr;
       struct _Phighl_blink_colr {
          Pfloat    blink_rate;
          Pgcolr    blink_colr;
       } highl_blink_colr;
       ...          /* implementation defined methods*/
    } highl_data;

 } Phighl_method;


 typedef enum {

    PTRL_EQUAL = 0,
    PTRL_NOT_EQUAL = 1,
    PTRL_GREATER = 2,
    PTRL_LESS = 3,
    PTRL_GREATER_OR_EQUAL = 4,
    PTRL_LESS_OR_EQUAL = 5,
    PTLG_BITWISE_AND = 6,
    PTLG_BITWISE_OR = 7,
    PTLG_BITWISE_XOR = 8

 } Ptest_comp;


 typedef enum {

    PFILTER_FAILS = 0,
    PFILTER_PASSES = 1

 } Pfilter_op;
```

```
typedef enum {

   PCOND_ALL_ENABLED,
   PCOND_ALL_DISABLED,
   PCOND_NOT_ALL_DISABLED,
   PCOND_NOT_ALL_ENABLED,
   PCOND_ALWAYS_PASS,
   PCOND_NEVER_PASS

} Pcond_flags_eval_op;


typedef union {

   unsigned int mask;
   struct _Pcond_flag_set {
      unsigned int flag0:1;
      unsigned int flag1:1;
      unsigned int flag2:1;
      unsigned int flag3:1;
      unsigned int flag4:1;
      unsigned int flag5:1;
      unsigned int flag6:1;
      unsigned int flag7:1;
      unsigned int flag8:1;
      unsigned int flag9:1;
      unsigned int flag10:1;
      unsigned int flag11:1;
      unsigned int flag12:1;
      unsigned int flag13:1;
      unsigned int flag14:1;
      unsigned int flag15:1;
      unsigned int flag16:1;
      unsigned int flag17:1;
      unsigned int flag18:1;
      unsigned int flag19:1;
      unsigned int flag20:1;
      unsigned int flag21:1;
      unsigned int flag22:1;
      unsigned int flag23:1;
      unsigned int flag24:1;
      unsigned int flag25:1;
      unsigned int flag26:1;
      unsigned int flag27:1;
      unsigned int flag28:1;
      unsigned int flag29:1;
      unsigned int flag30:1;
      unsigned int flag31:1;
   } flags;

} Pcond_flag_mask;
```

```
typedef struct {

   Pint                method;        /* test method                     */

   union _Ptest_data {
      struct Ptest_always_fail {
         Pint                unused;
      } test_always_fail;
      struct Ptest_always_succeed {
         Pint                unused;
      } test_always_succeed;
      struct Ptest_extent_3 {
         Plimit3             rect;          /* extent rectangle            */
         Ptest_comp          relation;      /* test relation               */
         Pfloat              threshhold;    /* threshhold value            */
      } test_extent_3;
      struct Ptest_extent {
         Plimit              rect;          /* extent rectangle            */
         Ptest_comp          relation;      /* test relation               */
         Pfloat              threshhold;    /* threshhold value            */
      } test_extent;
      struct Ptest_bounds_3 {
         Ppoint_list_list3 point_lists;     /* test object                 */
         Pclip_cond          cond;          /* test condition              */
      } test_bounds_3;
      struct Ptest_bounds {
         Ppoint_list_list  point_lists;     /* test object                 */
         Pclip_cond          cond;          /* test condition              */
      } test_bounds;
      struct Ptest_nameset {
         Pint                filter_type;   /* type of filter              */
         Pint                filter_id;     /* filter identifier           */
         Pfilter_op          cond;          /* test condition              */
      } test_nameset;
      struct Ptest_appl_int {
         Pint                appl_id;       /* appl. integer selector      */
         Ptest_comp          comparison;    /* test comparison             */
         Pint                data;          /* appl integer data value     */
      } test_appl_int;
      struct Ptest_appl_real {
         Pint                appl_id;       /* appl. real selector         */
         Ptest_comp          comparison;    /* test comparison             */
         Pfloat              data;          /* appl real data value        */
      } test_appl_real;
      struct Ptest_cond_flags {
         Pcond_flags_eval_op  op;      /* condition flags eval. op.        */
      } test_cond_flags;
```

```
        struct Ptest_appl_int_as_logical {
           Pint                 appl_id;     /* appl. integer selector   */
           Ptest_comp_logical comparison;  /* test comparison          */
           Pint                 data;        /* appl integer data value  */
        } test_appl_int;
        ...                             /* implementation defined      */
     } test_data;       /*              test data record*/
  } Ptest;”
```

Pages 195 and 196

The following text is merged into the text of the definition of pelem_data*:*

“
```
  typedef union {
    /* start of PHIGS element data */
    ...
    /* end of PHIGS element data */
    /* start of PHIGS PLUS element data */
    ...
    /* end of PHIGS PLUS element data */
    /* start of Full PHIGS element data */
    struct Pappl_int {
     Pint   integer_id;    /* application integer identifier          */
     Pint   value;         /* value for application integer           */
    } appl_int;
    struct Pappl_real {
     Pint      real_id;       /* application real identifier          */
     Pfloat    value;         /* value for application real           */
    } appl_real;
    struct Pcircle3 {
        Ppoint3      center_point;      /* center point                 */
        Pfloat       radius;            /* radius                       */
        Pvec3        ref_vecs[2];       /* reference vectors            */
    } circle3;
    struct Pcircle {
     Ppoint      center_point;          /* center point                 */
     Pfloat      radius;                /* radius                       */
    } circle;
    struct Pcircular_arc3 {
     Ppoint3              center_point;  /* center point                */
     Pfloat               radius;        /* radius                      */
     Pvec3                ref_vecs[2];   /* reference vectors           */
     Pfloat               start;         /* start angle                 */
     Pfloat               end;           /* end angle                   */
    } circular_arc3;
```

```
struct Pcircular_arc {
  Ppoint              center_point;   /* center point                     */
  Pfloat              radius;         /* radius                           */
  Pfloat              start;          /* start angle                      */
  Pfloat              end;            /* end angle                        */
} circular_arc;
struct Pellipse3 {
  Ppoint3         center_point;       /* center point                     */
  Pvec3           major_ref_vec;      /* major axis reference vector */
  Pvec3           minor_ref_vec;      /* minor axis reference vector */
} ellipse3;
struct Pellipse {
  Ppoint          center_point;       /* center point                     */
  Pvec            major_ref_vec;      /* major axis reference vector */
  Pvec            minor_ref_vec;      /* minor axis reference vector */
} ellipse;
struct Pelliptical_arc3 {
  Ppoint3         center_point;       /* center point                     */
  Pvec3           major_ref_vec;      /* major axis reference vector */
  Pvec3           minor_ref_vec;      /* minor axis reference vector */
  Pfloat          start;              /* start angle                      */
  Pfloat          end;                /* end angle                        */
} elliptical_arc3;
struct Pelliptical_arc {
  Ppoint          center_point;       /* center point                     */
  Pvec            major_ref_vec;      /* major axis reference vector */
  Pvec            minor_ref_vec;      /* minor axis reference vector */
  Pfloat          start;              /* start angle                      */
  Pfloat          end;                /* end angle                        */
} elliptical_arc;
struct Pfill_circle3 {
  Ppoint3      center_point;          /* center point                     */
  Pfloat       radius;                /* radius                           */
  Pvec3        ref_vecs[2];           /* reference vectors                */
} fill_circle3;
struct Pfill_circle {
  Ppoint       center_point;          /* center point                     */
  Pfloat       radius;                /* radius                           */
} fill_circle;
struct Pcircular_arc_close3 {
  Ppoint3             center_point;   /* center point                     */
  Pfloat              radius;         /* radius                           */
  Pvec3               ref_vecs[2];    /* reference vectors                */
  Pfloat              start;          /* start angle                      */
  Pfloat              end;            /* end angle                        */
  Pclosure            type;           /* closure type                     */
} circular_arc_close3;
```

```
struct Pcircular_arc_close {
 Ppoint              center_point;  /* center point                */
 Pfloat              radius;        /* radius                      */
 Pfloat              start;         /* start angle                 */
 Pfloat              end;           /* end angle                   */
 Pclosure            type;          /* closure type                */
} circular_arc_close;
struct Pfill_ellipse3 {
 Ppoint3         center_point;      /* center point                */
 Pvec3           major_ref_vec;     /* major axis reference vector */
 Pvec3           minor_ref_vec;     /* minor axis reference vector */
} fill_ellipse3;
struct Pfill_ellipse {
 Ppoint          center_point;      /* center point                */
 Pvec            major_ref_vec;     /* major axis reference vector */
 Pvec            minor_ref_vec;     /* minor axis reference vector */
} fill_ellipse;
struct Pelliptical_arc_close3 {
 Ppoint3         center_point;      /* center point                */
 Pvec3           major_ref_vec;     /* major axis reference vector */
 Pvec3           minor_ref_vec;     /* minor axis reference vector */
 Pfloat          start;             /* start angle                 */
 Pfloat          end;               /* end angle                   */
 Pclosure        type;              /* closure type                */
} elliptical_arc_close3;
struct Pelliptical_arc_close {
 Ppoint          center_point;      /* center point                */
 Pvec            major_ref_vec;     /* major axis reference vector */
 Pvec            minor_ref_vec;     /* minor axis reference vector */
 Pfloat          start;             /* start angle                 */
 Pfloat          end;               /* end angle                   */
 Pclosure        type;              /* closure type                */
} elliptical_arc_close;
Plinetype_adapt  linetype_adapt;    /* linetype adaptability       */
Plinetype_cont   linetype_cont;     /* linetype continuity         */
Phighl_method    highl_method;      /* highlighting method         */
Pint_list        active_textures;      /* active textures          */
Pint_list        back_active_textures; /* back active textures     */
Ppperspect_corr  perspect_corr;   /* textr perspective correction  */
struct Ptexture_res_opt_heur {
   Pint          opt_hint;         /* optimization hint            */
   Pint_list     usage_priorities; /* texture usage priorities     */
} texture_res_opt_heur;
```

```
   Pint_list       alpha_src_sel;  /* alpha source selector           */
   struct Pcond_exec_struct {
    Pint           struct_id;         /* structure identifier         */
    Ptest          test;              /* condition test               */
   } cond_exec_struct;
   struct Pcond_inst_struct {
    Pint           struct_id;         /* structure identifier         */
    Ptest          test;              /* condition test               */
   } cond_inst_struct;
   Ptest           cond_return;       /* condition test               */
   struct Pcond_skip_elems {
    Pint           skip_count;        /* number of elements to skip   */
    Ptest          test;              /* condition test               */
   } cond_skip_elems;
   struct Pcond_skip_to_label {
    Pint           label;             /* label to which to skip       */
    Ptest          test;              /* condition test               */
   } cond_skip_to_label;
   /* end of Full PHIGS element data */

 } Pelem_data;”
```

Page 247

A.3 External functions

The following text is appended:

```
“/* WORKSTATION TYPE CREATE */

void pws_type_create (

   Pint                    ws_type,        /* workstation type         */
   const Pconfig_setting   *settings_list,/* array of config. settings*/
   Pint                    *new_ws_type    /* OUT new workstation type */

);

/* WORKSTATION TYPE SET */

void pws_type_set (

   Pint                    ws_type,        /* workstation type         */
   const Pconfig_setting   *settings_list  /* array of config. settings*/

);
```

```
/* WORKSTATION TYPE GET */

void pws_type_get (

   Pint           ws_type,          /* workstation type                */
   const char     *setting_name,    /* configuration setting name      */
   Pstore         store,            /* store object for returned value */
   Pconfig_value  **setting_value   /* OUT configuration setting value */

);

/* WORKSTATION TYPE DESTROY */

void pws_type_destroy (

   Pint        ws_type                     /* workstation type        */

);

/* REDRAW ALL STRUCTURES ON TARGET */

void predraw_all_structs_on_targ (

   Pint               ws_id,           /* workstation identifier      */
   const Ptarg_addr   *targ_addr,      /* target address              */
   Pctrl_flag         ctrl_flag        /* control flag                */

);

/* REDRAW ALL STRUCTURES FROM POSTING GROUP */

void predraw_all_structs_from_grp (

   Pint               ws_id,           /* workstation identifier      */
   Pint               grp_id,          /* posting group               */
   Pbackg_redisplay   redisplay_flag   /* background redisplay flag   */

);

/* UPDATE TARGET */

void pupd_targ (

   Pint               ws_id,           /* workstation identifier      */
   const Ptarg_addr   *targ_addr,      /* target address              */
   Pregen_flag        regen_flag       /* update regeneration flag    */

);
```

```
/* DEFINE POSTING GROUP */

void pdefine_post_grp (

   Pint                  ws_id,           /* workstation identifier      */
   Pint                  grp_id,          /* posting group identifier    */
   Pint                  ref_grp_id,      /* reference posting group id  */
   Prel_pri              rel_pri,         /* relative priority           */
   Pint                  def_view_ind,    /* default view index          */
   Pbackg_style          backg_style,     /* posting group backg. style  */
   const Pbackg_method   *backg_method,   /* posting group backg. method */
   Pborder_indic         border_indic,    /* posting group border indic. */
   Pint                  border_ind       /* posting group border index  */

);

/* UNDEFINE POSTING GROUP */

void pundefine_post_grp (

   Pint                ws_id,             /* workstation identifier      */
   Pint                grp_id             /* posting group identifier    */

);

/* SET POSTING GROUP STATUS */

void pset_post_grp_status (

   Pint                ws_id,             /* workstation identifier      */
   Pint                grp_id,            /* posting group identifier    */
   Ppost_grp_status    post_grp_status    /* posting group status        */

);

/* SET POSTING GROUP PRIORITY */

void pset_post_grp_priority (

   Pint                ws_id,             /* workstation identifier      */
   Pint                grp_id,            /* posting group identifier    */
   Pint                ref_grp_id,        /* reference posting group id  */
   Prel_pri            rel_pri            /* relative priority           */

);

/* SET POSTING GROUP BACKGROUND STYLE */

void pset_post_grp_backg_style (

   Pint             ws_id,            /* workstation identifier          */
   Pint             grp_id,           /* posting group identifier        */
   Pbackg_style     backg_style       /* posting group background style */

);
```

```
/* SET POSTING GROUP BACKGROUND METHOD */

void pset_post_grp_backg_method (

   Pint                 ws_id,          /* workstation identifier      */
   Pint                 grp_id,         /* posting group identifier    */
   const Pbackg_method  *backg_method   /* posting group background    */
                                        /* method                      */

);

/* SET POSTING GROUP BORDER INDICATOR */

void pset_post_grp_border_indicator (

   Pint           ws_id,            /* workstation identifier          */
   Pint           grp_id,           /* posting group identifier        */
   Pborder_indic  border_indic      /* posting group border indicator */

);

/* SET POSTING GROUP BORDER INDEX */

void pset_post_grp_border_ind (

   Pint       ws_id,              /* workstation identifier            */
   Pint       grp_id,             /* posting group identifier          */
   Pint       border_ind          /* posting group border index        */

);

/* ASSOCIATE IMAGE RESOURCE */

void passoc_image_res (

   Pint                ws_id,           /* workstation identifier      */
   Pint                image_res_id,    /* image resource identifier   */
   const Pint_size     *dims,           /* m,n dimensions              */
   const Pimage_res    *image_res,      /* image resource              */
   Paccess_flag        *access_flag     /* OUT access flag             */

);

/* DISASSOCIATE IMAGE RESOURCE */

void pdisassoc_image_res (

   Pint                ws_id,           /* workstation identifier      */
   Pint                image_res_id     /* image resource identifier   */

);
```

```
/* SET DEVICE COORDINATE CLIP REGIONS 3 */

void pset_dc_clip_regions3 (

   Pint                 ws_id,          /* workstation identifier      */
   const Plimit3_list   *limit_list     /* list of clip regions        */

);

/* SET DEVICE COORDINATE CLIP REGIONS */

void pset_dc_clip_regions (

   Pint                ws_id,           /* workstation identifier      */
   const Plimit_list   *limit_list      /* list of clip regions        */

);

/* SET TARGET MANIPULATION MODE */

void pset_targ_manip_mode (

   Pint               ws_id,            /* workstation identifier      */
   Pint               targ_manip_mode   /* target manipulation mode    */

);

/* SET TARGET DISPOSITION */

void pset_targ_dispos (

  Pint                  ws_id,          /* workstation identifier      */
  const Ptarg_op_list   *targ_ops       /* target operations list      */

);

/* SET DISPLAY TARGET */

void pset_disp_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *targ_addr       /* target address              */

);

/* SET RENDERING TARGET */

void pset_rend_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *targ_addr       /* target address              */

);
```

```
/* CLEAR TARGET */

void pclear_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *targ_addr       /* target address              */

);

/* COPY TARGET */

void pcopy_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *src_targ_addr,  /* source target address       */
   const Ptarg_addr    *dest_targ_addr  /* destination target address  */

);

/* CREATE TARGET */

void pcreate_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *targ_addr       /* target address              */

);

/* DESTROY TARGET */

void pdestroy_targ (

   Pint                ws_id,           /* workstation identifier      */
   const Ptarg_addr    *targ_addr       /* target address              */

);

/* SET STATE OF VISUAL REPRESENTATION */

void pset_st_visual_rep (

   Pint             ws_id,          /* workstation identifier          */
   Pvisual_st       visual_st       /* state of visual representation  */

);

/* SET TARGET STATE OF VISUAL REPRESENTATION */

void pset_targ_st_visual_rep (

   Pint                ws_id,         /* workstation identifier         */
   const Ptarg_addr    *targ_addr,    /* target address                 */
   Pvisual_st          visual_st      /* state of visual representation */

);
```

```
/* ASSOCIATE TRAVERSAL RESOURCE */

void passoc_trav_res (

   Pint               ws_id,        /* workstation identifier          */
   Pint               res_id,       /* traversal resource identifier   */
   const Ptarg_addr   *targ_addr    /* source target address           */

);

/* DISASSOCIATE TRAVERSAL RESOURCE */

void pdisassoc_trav_res (

   Pint               ws_id,        /* workstation identifier          */
   Pint               res_id        /* traversal resource identifier   */

);

/* MANIPULATE TRAVERSAL RESOURCE */

void pmanip_trav_res (

  Pint                 ws_id,        /* workstation identifier          */
  Pint                 res_id,       /* traversal resource identifier */
  const Ptrav_res_op   *manip_data   /* traversal resource operation    */

);

/* RESET ALL TRAVERSAL RESOURCES */

void preset_all_trav_res (

  Pint                 ws_id          /* workstation identifier        */

);

/* RETRIEVE WINDOW SYSTEM COLOUR */

void pret_window_system_colr (

  Pint           ws_id,           /* workstation identifier             */
  const Pgcolr   *colr,           /* colour sought                      */
  Pstore         store,           /* store object for returned value    */
  Pdata          **winsys_colr    /* OUT window system colour           */

);

/* SET APPLICATION INTEGER */

void pset_appl_int (

   Pint    integer_id,    /* application integer identifier              */
   Pint    value          /* value of application integer                */

);
```

```
/* SET APPLICATION REAL */

void pset_appl_real (

   Pint      real_id,        /* application real identifier           */
   Pfloat    value           /* value of application real             */

);

/* CIRCLE 3 */

void pcircle3 (

   const Ppoint3        *center_point,    /* center point             */
   Pfloat               radius,           /* radius                   */
   const Pvec3          ref_vecs[2]       /* reference vectors        */

);

/* CIRCLE */

void pcircle (

   const Ppoint         *center_point,    /* center point             */
   Pfloat               radius            /* radius                   */

);

/* CIRCULAR ARC 3 */

void pcircular_arc3 (

   const Ppoint3        *center_point,    /* center point             */
   Pfloat               radius,           /* radius                   */
   const Pvec3          ref_vecs[2],      /* reference vectors        */
   Pfloat               start,            /* start angle              */
   Pfloat               end               /* end angle                */

);

/* CIRCULAR ARC */

void pcircular_arc (

   const Ppoint         *center_point,    /* center point             */
   Pfloat               radius,           /* radius                   */
   Pfloat               start,            /* start angle              */
   Pfloat               end               /* end angle                */

);
```

```
/* ELLIPSE 3 */

void pellipse3 (

   const Ppoint3    *center_point,      /* center point                */
   const Pvec3      *major_ref_vec,     /* major axis reference vector */
   const Pvec3      *minor_ref_vec      /* minor axis reference vector */

);

/* ELLIPSE */

void pellipse (

   const Ppoint     *center_point,      /* center point                */
   const Pvec       *major_ref_vec,     /* major axis reference vector */
   const Pvec       *minor_ref_vec      /* minor axis reference vector */

);

/* ELLIPTICAL ARC 3 */

void pelliptical_arc3 (

   const Ppoint3    *center_point,      /* center point                */
   const Pvec3      *major_ref_vec,     /* major axis reference vector */
   const Pvec3      *minor_ref_vec,     /* minor axis reference vector */
   Pfloat           start,              /* start angle                 */
   Pfloat           end                 /* end angle                   */

);

/* ELLIPTICAL ARC */

void pelliptical_arc (

   const Ppoint     *center_point,      /* center point                */
   const Pvec       *major_ref_vec,     /* major axis reference vector */
   const Pvec       *minor_ref_vec,     /* minor axis reference vector */
   Pfloat           start,              /* start angle                 */
   Pfloat           end                 /* end angle                   */

);

/* FILL CIRCLE 3 */

void pfill_circle3 (

   const Ppoint3       *center_point,   /* center point                */
   Pfloat              radius,          /* radius                      */
   const Pvec3         ref_vecs[2]      /* reference vectors           */

);
```

```
/* FILL CIRCLE */

void pfill_circle (

   const Ppoint        *center_point,   /* center point                */
   Pfloat              radius           /* radius                      */

);

/* CIRCULAR ARC CLOSE 3 */

void pcircular_arc_close3 (

   const Ppoint3       *center_point,   /* center point                */
   Pfloat              radius,          /* radius                      */
   const Pvec3         ref_vecs[2],     /* reference vectors           */
   Pfloat              start,           /* start angle                 */
   Pfloat              end,             /* end angle                   */
   Pclosure            type             /* closure type                */

);

/* CIRCULAR ARC CLOSE */

void pcircular_arc_close (

   const Ppoint        *center_point,   /* center point                */
   Pfloat              radius,          /* radius                      */
   Pfloat              start,           /* start angle                 */
   Pfloat              end,             /* end angle                   */
   Pclosure            type             /* closure type                */

);

/* FILL ELLIPSE 3 */

void pfill_ellipse3 (

   const Ppoint3    *center_point,      /* center point                */
   const Pvec3      *major_ref_vec,     /* major axis reference vector */
   const Pvec3      *minor_ref_vec      /* minor axis reference vector */

);

/* FILL ELLIPSE */

void pfill_ellipse (

   const Ppoint     *center_point,      /* center point                */
   const Pvec       *major_ref_vec,     /* major axis reference vector */
   const Pvec       *minor_ref_vec      /* minor axis reference vector */

);
```

```
/* ELLIPTICAL ARC CLOSE 3 */

void pelliptical_arc_close3 (

   const Ppoint3    *center_point,     /* center point              */
   const Pvec3      *major_ref_vec,    /* major axis reference vector */
   const Pvec3      *minor_ref_vec,    /* minor axis reference vector */
   Pfloat           start,             /* start angle               */
   Pfloat           end,               /* end angle                 */
   Pclosure         type               /* closure type              */

);

/* ELLIPTICAL ARC CLOSE */

void pelliptical_arc_close (

   const Ppoint     *center_point,     /* center point              */
   const Pvec       *major_ref_vec,    /* major axis reference vector */
   const Pvec       *minor_ref_vec,    /* minor axis reference vector */
   Pfloat           start,             /* start angle               */
   Pfloat           end,               /* end angle                 */
   Pclosure         type               /* closure type              */

);

/* SET HIGHLIGHTING INDEX */

void pset_highl_ind (

   Pint       highl_ind                /* highlighting index        */

);

/* SET LINETYPE ADAPTABILITY */

void pset_linetype_adapt (

   Plinetype_adapt     adaptability    /* linetype adaptability     */

);

/* SET LINETYPE CONTINUITY */

void pset_linetype_cont (

   Plinetype_cont      continuity      /* linetype continuity       */

);

/* SET LINETYPE OFFSET */

void pset_linetype_offset (

   Pfloat              offset          /* linetype offset           */

);
```

```
/* SET HIGHLIGHTING METHOD */

void pset_highl_method (

   const Phighl_method  *method        /* highlighting method         */

);

/* SET CONDITION FLAGS */

void pset_cond_flags (

   const Pcond_flags   *enable_mask,    /* condition flag enable mask  */
   const Pcond_flags   *disable_mask    /* condition flag disable mask */

);

/* SET CONDITION FLAGS FROM TESTS */

void pset_cond_flags_from_tests (

   const Pcond_test_list   *tests       /* list of condition flag tests
                                           */

);

/* SET ACTIVE TEXTURES */

void pset_active_textures (

   const Pint_list  *active_textures    /* list of active textures     */

);

/* SET BACK ACTIVE TEXTURES */

void pset_back_active_textures (

   const Pint_list  *back_active_textures /* list of active textures   */

);

/* SET TEXTURE PERSPECTIVE CORRECTION */

void pset_texture_perspect_corr (

   Pperspect_corr   perspect_corr    /* texture perspective correction */

);

/* SET TEXTURE SAMPLING FREQUENCY */

void pset_texture_sampling_freq (

   Pint          sampling_freq          /* texture sampling frequency  */

);
```

```
/* SET TEXTURE RESOURCE OPTIMIZATION HEURISTICS */

void pset_texture_res_opt_heur (

   Pint              opt_hint,           /* optimization hint          */
   const Pint_list   *usage_priorities   /* texture usage priorities   */

);

/* SET TRANSPARENCY */

void pset_transparency (

   Pfloat         transparency           /* transparency               */

);

/* SET BACK TRANSPARENCY */

void pset_back_transparency (

  Pfloat           back_transparency     /* back transparency          */

);

/* SET ALPHA SOURCE SELECTOR */

void pset_alpha_src_sel (

   const Pint_list  *alpha_src_sel       /* alpha source selector list  */

);

/* SET ALPHA DATA SELECTION INDEX */

void pset_alpha_data_sel_ind (

   Pint       alpha_data_sel_ind         /* alpha data selection index  */

);

/* DEFINE LINETYPE */

void pdefine_linetype (

   Pint                   linetype,          /* linetype               */
   Pfloat                 repeat_length,     /* repeat length          */
   const Pfloat_list      *segment_lengths   /* segment length list    */

);

/* DEFINE MARKER TYPE */

void pdefine_marker_type (

   Pint                  marker_type,      /* marker type              */
   const Pmarker_desc    *marker           /* marker descriptor        */

);
```

```
/* SET EXTENDED PATTERN REPRESENTATION */

void pset_ext_pat_rep (

   Pint               ws_id,        /* workstation identifier         */
   Pint               pat_ind,      /* pattern bundle index           */
   const Ppattern     *pattern      /* extended pattern representation */

);

/* SET HIGHLIGHTING REPRESENTATION */

void pset_highl_rep (

   Pint                  ws_id,          /* workstation identifier     */
   Pint                  highl_ind,      /* highlighting bundle index  */
   const Phighl_method  *method          /* highlighting method        */

);

/* SET TEXTURE  REPRESENTATION */

void pset_texture_rep (

   Pint                     ws_id,     /* workstation identifier       */
   Pint                     index,     /* texture index                */
   const Ptexture_rep       *rep       /* texture representation       */

);

/* SET TEXTURE PARAMETRIZATION */

void pset_texture_param (

   Pint                    ws_id,   /* workstation identifier          */
   Pint                    index,   /* texture index                   */
   const Ptexture_param    *param   /* texture parametrization         */

);

/* SET TEXTURE COMPOSITION */

void pset_texture_composition (

   Pint                     ws_id,      /* workstation identifier      */
   Pint                     index,      /* texture index               */
   const Ptexture_compos    *compos     /* texture composition         */

);
```

```
/* SET TEXTURE SAMPLING */

void pset_texture_sampling (

   Pint                        ws_id,     /* workstation identifier      */
   Pint                        index,     /* texture index               */
   const Ptexture_sampling     *sampling  /* texture sampling            */

);

/* SET TEXTURE BINDING */

void pset_texture_binding (

   Pint                       ws_id,      /* workstation identifier      */
   Pint                       index,      /* texture index               */
   const Ptexture_binding     *binding    /* texture binding             */

);

/* SET TRANSPARENCY MODE */

void pset_transparency_mode (

   Pint                       ws_id,      /* workstation identifier      */
   Pint                       mode        /* transparency mode           */

);

/* SET TRANSPARENCY THRESHOLDS */

void pset_transparency_thresholds (

   Pint                ws_id,         /* workstation identifier          */
   const Pfloat_list   *thresholds    /* list of transparency thresholds */

);

/* SET APPLICATION FILTER */

void pset_appl_filter (

   Pint            ws_id,                 /* workstation identifier      */
   Pint            filter_selector,       /* application filter selector */
   const Pfilter   *filter                /* application filter          */

);
```

```
/* CREATE MIPMAP TEXTURE */

void pcreate_mipmap_texture (

   Pint     ws_id,                  /* workstation identifier          */
   Pint     src_image_res_id,       /* source image resource id        */
   Pint     mipmap_image_res_id,    /* mipmap image resource id        */
   Pint     max_levels,             /* max mipmap levels               */
   Pint     creation_heuristic      /* mipmap creation heuristic       */

);

/* SET VIEW REFERENCE POINT 3 */

void pset_view_ref_point3 (

   Pint             ws_id,              /* workstation id              */
   Pint             view_ind,           /* view index                  */
   const Ppoint3    *view_ref_point     /* view reference point        */

);

/* SET VIEW REFERENCE POINT */

void pset_view_ref_point (

   Pint             ws_id,              /* workstation id              */
   Pint             view_ind,           /* view index                  */
   const Ppoint     *view_ref_point     /* view reference point        */

);

/* SET VIEW PLANE NORMAL */

void pset_view_plane_norm (

   Pint                ws_id,           /* workstation id              */
   Pint                view_ind,        /* view index                  */
   const Pvec3         *view_norm_vec   /* view normal vector          */

);

/* SET VIEW UP VECTOR 3 */

void pset_view_up_vec3 (

   Pint                ws_id,           /* workstation id              */
   Pint                view_ind,        /* view index                  */
   const Pvec3         *view_up_vec     /* view up vector              */

);
```