THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

XGL, XView are trademarks of Sun Microsystems, Incorporated.
FOREWORD

This manual has been divided into three volumes.

Volume 1
Contains all functions that do not begin with INQUIRE_ or SET_.

Volume 2
Contains all functions that begin with INQUIRE_.

Volume 3
Contains all functions that begin with SET_, followed by an index for the entire three-volume set.

The page numbering of the reference manual is consecutive, starting with Volume 1 and ending with Volume 3 (that is, the page numbering begins at page 1 in Volume 1 and increments throughout the entire set).
**NAME**
intro – introduction to the SunPHIGS graphics library

**DESCRIPTION**
The functions in the SunPHIGS graphics library implement the *Programmer’s Hierarchical Interactive Graphics System* or PHIGS. These reference manual pages describe the syntax of each function in the SunPHIGS graphics library. Their organization is similar to the *SunOS Reference Manual*. The first section is divided into manual entries for each function. These manual entries are further divided into sections that describe C syntax, FORTRAN syntax, function parameters, and error messages.

**Reserved Names**
SunPHIGS applications should avoid using variable and function names that conflict with names that SunPHIGS uses internally. Since SunPHIGS programs are linked to the XGL and Xlib libraries, do not use symbols that start with the prefixes reserved by those packages, symbols in the Math library, or system calls. The public symbol names provided by SunPHIGS all start with \texttt{p}. To avoid conflicts with the SunPHIGS name space, do not define any of the following symbols:

- PHIGS C and FORTRAN function names.
- Names with the prefixes \texttt{phigs}, \texttt{phg}, \texttt{P}, \texttt{p}, or \texttt{hk}.
- Names with XGL or X11 prefixes \texttt{XGL}, \texttt{Xgl}, \texttt{xgl}, \texttt{X11}, or \texttt{X}.
- System library symbols, such as \texttt{time}.

**Abbreviations**
The following list describes the abbreviations used in this manual.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>ASF</td>
<td>aspect source flag</td>
</tr>
<tr>
<td>CGM</td>
<td>computer graphics metafile</td>
</tr>
<tr>
<td>CSS</td>
<td>Central Structure Store</td>
</tr>
<tr>
<td>DC</td>
<td>device coordinates</td>
</tr>
<tr>
<td>GDP</td>
<td>generalized drawing primitive</td>
</tr>
<tr>
<td>GSE</td>
<td>generalized structure element</td>
</tr>
<tr>
<td>HLH3R</td>
<td>hidden line/hidden surface removal</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>LUN</td>
<td>FORTRAN logical unit number</td>
</tr>
<tr>
<td>MC</td>
<td>modelling coordinates</td>
</tr>
<tr>
<td>NPC</td>
<td>normalized projection coordinates</td>
</tr>
<tr>
<td>NURB</td>
<td>non-uniform rational B-spline</td>
</tr>
<tr>
<td>PDT</td>
<td>PHIGS description table</td>
</tr>
<tr>
<td>PET</td>
<td>prompt/echo type</td>
</tr>
<tr>
<td>RGB</td>
<td>red/green/blue colour model</td>
</tr>
<tr>
<td>TLC</td>
<td>text local coordinates</td>
</tr>
<tr>
<td>WC</td>
<td>world coordinates</td>
</tr>
<tr>
<td>WS</td>
<td>workstation</td>
</tr>
</tbody>
</table>

modified 2 April 1993
PHIGS has four operating state variables, each with two possible values:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Open State Code</th>
<th>Closed State Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>System State</td>
<td>PHOP</td>
<td>PHCL</td>
</tr>
<tr>
<td>Workstation State</td>
<td>WSOP</td>
<td>WSCL</td>
</tr>
<tr>
<td>Structure State</td>
<td>STOP</td>
<td>STCL</td>
</tr>
<tr>
<td>Archive State</td>
<td>AROP</td>
<td>ARCL</td>
</tr>
</tbody>
</table>

Most PHIGS functions are allowed only in certain states. The SYNOPSIS section of each function encodes the Required PHIGS Operating States. An asterisk (*) is used when an operating state variable may be in either state (that is, does not matter). All of the operating states are initially closed.

Deviations from the PHIGS Standard

1. Sun’s gray scale systems are monochrome by the PHIGS definition, but INQUIRE WORKSTATION TYPE will return colour.
2. Currently, SunPHIGS supports only the RGB colour model. The coordinate space is [0,1] in all three dimensions.

Limitations

The following limitations are permitted by the PHIGS standard:

1. The PATTERN fill style is not supported.
2. CELL ARRAY uses the minimal simulation by drawing the bounding quadrilateral.
3. No metafile input workstations are supported.

Extensions to the PHIGS Standard

The SunPHIGS library contains optional functions that are not in the PHIGS specification, but are extremely useful in the Sun environment. These, plus other Sun-specific function and extension functions from PHIGS PLUS are described in SunPHIGS Extensions Reference Manual.

Include Files

SunPHIGS applications should use the constants and enumerated values defined in the proper header files to decrease the possibility of passing bad data and avoid conflict with future changes. C applications should include the phigs.h file. Each FORTRAN programming unit should include the phigs77.h file to gain access to the default workstation types and SunPHIGS FORTRAN constants. These files are located in the $PHIGSHOME/include directory. See the example programs in and below $PHIGSHOME/lib/phigsn.n/examples (where n.n is substituted with the current release level) or in Getting Started with SunPHIGS.

Common Problems

1. You cannot mix X Tool and X Drawable workstations. You must decide which workstation type to use at link-time and link appropriately. If the application is inappropriately linked, use of the workstation type produces error 52.
2. SunPHIGS offers significant performance enhancements transparently when a graphics accelerator is available. Features are provided in software when hardware support is unavailable. Use ESCAPE -16 (3P) to determine the level of hardware support. The Device Dependent Acceleration chapter of the SunPHIGS Programming manual.
documents supported accelerators and the SunPHIGS features they support. It also contains a SunOS and Display Devices appendix that describes the steps necessary for an accelerator to be available to SunPHIGS. The example WHAT_HW program, from the examples directory, describes how to determine system configuration and level of support for various SunPHIGS features.

3. The display update state heavily influences performance and visual results. The default deferral mode is ASAP (As Soon As Possible), so almost every structure edit or change to a workstation table entry causes a regeneration (that is, full structure traversal). See SET DISPLAY UPDATE STATE (3P) in this manual.

4. Attempting to insert elements in an open structure is complicated while in REPLACE edit mode. In such a case, each inserted element overwrites the existing element at the current element position. In REPLACE mode, the element pointer is not implicitly incremented prior to inserting an element, as it is in INSERT mode.

5. Some workstation description table inquiries require that the passed workstation type be a specific workstation type; that is, associated with an open workstation. In these cases, if the workstation type passed to the inquiry function is not a specific type, error 051 will be returned, indicating that a specific workstation type must be used.

6. Segmentation faults are often the result of passing bad pointers to the SunPHIGS C binding.

7. The PHIGS standard says that the application should avoid recursive structure networks and that the implementation does not need to check for this problem. SunPHIGS does not check for recursive structure networks. One of the following error messages can occur when an application attempts to create a recursive structure network:

   SunPHIGS: child died on signal 11
   Segmentation fault (core dumped)

SunPHIGS implements the PHIGS International Standard FORTRAN Binding (ISO/IEC 9593-1). There are two PHIGS FORTRAN bindings, one for full FORTRAN 77 and one for compilers that do not fully conform to the FORTRAN 77 specification. In most cases these bindings are identical. However, the bindings for some subroutines vary slightly. For these functions, SunPHIGS provides both binding forms.

If an array size specified by the application is too small to hold all of the data that should be returned, an error is raised. In this case, output parameters indicating the number of array elements being returned are not set to zero, which would reflect that no array elements are in fact being returned. Instead, the SunPHIGS FORTRAN binding returns the actual array size in this case, to tell application how much space is needed to obtain the requested data. (INQUIRE CURRENT ELEMENT CONTENT is an exception to this because the required data is already available from INQUIRE CURRENT ELEMENT TYPE AND SIZE.)
Intermixing calls to the SUNPHIGS C and FORTRAN language bindings is not supported. At the application level, languages may be mixed, but SUNPHIGS functions may be called in only one language binding per application.

C Binding Notes

SUNPHIGS implements the PHIGS International Standard C Binding (ISO/IEC 9593-4). A definition of the binding is in the file $PHIGSHOME/lib/int/phigs.c_bind. This file contains stubs for all of the binding functions and is used to build the lint library. The data structures are defined in the file $PHIGSHOME/include/phigs.h.

Lists of Standard Functions

The 3P functions described in this manual are as follows:

ADD NAMES TO SET

   add_names_to_set (3P)

ANNOTATION TEXT RELATIVE

   annotation_text_relative (3P)

   annotation_text_relative_3 (3P)

APPLICATION DATA

   application_data (3P)

ARCHIVE ALL STRUCTURES

   archive_all_structures (3P)

ARCHIVE STRUCTURE NETWORKS

   archive_structure_networks (3P)

ARCHIVE STRUCTURES

   archive_structures (3P)

AWAIT EVENT

   await_event (3P)

BUILD TRANSFORMATION MATRIX

   build_transformation_matrix (3P)

   build_transformation_matrix_3 (3P)

CELL ARRAY

   cell_array (3P)

   cell_array_3 (3P)

CHANGE STRUCTURE IDENTIFIER

   change_structure_identifier (3P)

CHANGE STRUCTURE IDENTIFIER AND REFERENCES

   change_structure_identifier_and_references (3P)
CHANGE STRUCTURE REFERENCES
  change_structure_references (3P)
CLOSE ARCHIVE FILE
  close_archive_file (3P)
CLOSE PHIGS
  close_phigs (3P)
CLOSE STRUCTURE
  close_structure (3P)
CLOSE WORKSTATION
  close_workstation (3P)
COMPOSE MATRIX
  compose_matrix (3P)
COMPOSE MATRIX 3
  compose_matrix_3 (3P)
COMPOSE TRANSFORMATION MATRIX
  compose_transformation_matrix (3P)
COMPOSE TRANSFORMATION MATRIX 3
  compose_transformation_matrix_3 (3P)
COPY ALL ELEMENTS FROM STRUCTURE
  copy_all_elements_from_structure (3P)
CREATE STORE
  create_store (3P)
DELETE ALL STRUCTURES
  delete_all_structures (3P)
DELETE ALL STRUCTURES FROM ARCHIVE
  delete_all_structures_from_archive (3P)
DELETE ELEMENT
  delete_element (3P)
DELETE ELEMENT RANGE
  delete_element_range (3P)
DELETE ELEMENTS BETWEEN LABELS
  delete_elements_between_labels (3P)
DELETE STORE
  delete_store (3P)
DELETE STRUCTURE
  delete_structure (3P)
DELETE STRUCTURE NETWORK
  delete_structure_network (3P)

modified 2 April 1993
DELETE STRUCTURE NETWORKS FROM ARCHIVE
   delete_structure_networks_from_archive (3P)
DELETE STRUCTURES FROM ARCHIVE
   delete_structures_from_archive (3P)
ELEMENT SEARCH
   element_search (3P)
EMERGENCY CLOSE PHIGS
   emergency_close_phigs (3P)
EMPTY STRUCTURE
   empty_structure (3P)
ERROR HANDLING
   error_handling (3P)
ERROR LOGGING
   error_logging (3P)
EVALUATE VIEW MAPPING MATRIX
   evaluate_view_mapping_matrix (3P)
EVALUATE VIEW MAPPING MATRIX 3
   evaluate_view_mapping_matrix_3 (3P)
EVALUATE VIEW ORIENTATION MATRIX
   evaluate_view_orientation_matrix (3P)
EVALUATE VIEW ORIENTATION MATRIX 3
   evaluate_view_orientation_matrix_3 (3P)
EXECUTE STRUCTURE
   execute_structure (3P)
FILL AREA
   fill_area (3P)
FILL AREA 3
   fill_area_3 (3P)
FILL AREA SET
   fill_area_set (3P)
FILL AREA SET 3
   fill_area_set_3 (3P)
FLUSH DEVICE EVENTS
   flush_device_events (3P)
GET CHOICE
   get_choice (3P)
GET ITEM TYPE FROM METAFILE
   get_item_type_from_metafile (3P)
GET LOCATOR
  get_locator (3P)
GET LOCATOR 3
  get_locator_3 (3P)
GET PICK
  get_pick (3P)
GET STRING
  get_string (3P)
GET STROKE
  get_stroke (3P)
GET STROKE 3
  get_stroke_3 (3P)
GET VALUATOR
  get_valuator (3P)
INCREMENTAL SPATIAL SEARCH
  incremental_spatial_search (3P)
INCREMENTAL SPATIAL SEARCH 3
  incremental_spatial_search_3 (3P)
INITIALIZE CHOICE
  initialize_choice (3P)
INITIALIZE CHOICE 3
  initialize_choice_3 (3P)
INITIALIZE LOCATOR
  initialize_locator (3P)
INITIALIZE LOCATOR 3
  initialize_locator_3 (3P)
INITIALIZE PICK
  initialize_pick (3P)
INITIALIZE PICK 3
  initialize_pick_3 (3P)
INITIALIZE STRING
  initialize_string (3P)
INITIALIZE STRING 3
  initialize_string_3 (3P)
INITIALIZE STROKE
  initialize_stroke (3P)
INITIALIZE STROKE 3
  initialize_stroke_3 (3P)
INITIALIZE VALUATOR
   initialize_valuator (3P)
INITIALIZE VALUATOR 3
   initialize_valuator_3 (3P)
INQUIRE ALL CONFLICTING STRUCTURES
   inquire_all_conflicting_structures (3P)
INQUIRE ANNOTATION FACILITIES
   inquire_annotation_facilities (3P)
INQUIRE ARCHIVE FILES
   inquire_archive_files (3P)
INQUIRE ARCHIVE STATE VALUE
   inquire_archive_state_value (3P)
INQUIRE CHOICE DEVICE STATE
   inquire_choice_device_state (3P)
INQUIRE CHOICE DEVICE STATE 3
   inquire_choice_device_state_3 (3P)
INQUIRE COLOUR FACILITIES
   inquire_colour_facilities (3P)
INQUIRE COLOUR MAPPING STATE
   inquire_colour_mapping_state (3P)
INQUIRE COLOUR MODEL
   inquire_colour_model (3P)
INQUIRE COLOUR MODEL FACILITIES
   inquire_colour_model_facilities (3P)
INQUIRE COLOUR REPRESENTATION
   inquire_colour_representation (3P)
INQUIRE CONFLICT RESOLUTION
   inquire_conflict_resolution (3P)
INQUIRE CONFLICTING STRUCTURES IN NETWORK
   inquire_conflicting_structures_in_network (3P)
INQUIRE CURRENT ELEMENT CONTENT
   inquire_current_element_content (3P)
INQUIRE CURRENT ELEMENT TYPE AND SIZE
   inquire_current_element_type_and_size (3P)
INQUIRE DEFAULT CHOICE DEVICE DATA
   inquire_default_choice_device_data (3P)
INQUIRE DEFAULT CHOICE DEVICE DATA 3
   inquire_default_choice_device_data_3 (3P)
INQUIRE DEFAULT DISPLAY UPDATE STATE
  inquire_default_display_update_state (3P)
INQUIRE DEFAULT LOCATOR DEVICE DATA
  inquire_default_locator_device_data (3P)
INQUIRE DEFAULT LOCATOR DEVICE DATA 3
  inquire_default_locator_device_data_3 (3P)
INQUIRE DEFAULT PICK DEVICE DATA
  inquire_default_pick_device_data (3P)
INQUIRE DEFAULT PICK DEVICE DATA 3
  inquire_default_pick_device_data_3 (3P)
INQUIRE DEFAULT STRING DEVICE DATA
  inquire_default_string_device_data (3P)
INQUIRE DEFAULT STRING DEVICE DATA 3
  inquire_default_string_device_data_3 (3P)
INQUIRE DEFAULT STROKE DEVICE DATA
  inquire_default_stroke_device_data (3P)
INQUIRE DEFAULT STROKE DEVICE DATA 3
  inquire_default_stroke_device_data_3 (3P)
INQUIRE DEFAULT VALUATOR DEVICE DATA
  inquire_default_valuator_device_data (3P)
INQUIRE DEFAULT VALUATOR DEVICE DATA 3
  inquire_default_valuator_device_data_3 (3P)
INQUIRE DISPLAY SPACE SIZE
  inquire_display_space_size (3P)
INQUIRE DISPLAY SPACE SIZE 3
  inquire_display_space_size_3 (3P)
INQUIRE DISPLAY UPDATE STATE
  inquire_display_update_state (3P)
INQUIRE DYNAMICS OF STRUCTURES
  inquire_dynamics_of_structures (3P)
INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES
  inquire_dynamics_of_workstation_attributes (3P)
INQUIRE EDGE FACILITIES
  inquire_edge_facilities (3P)
INQUIRE EDGE REPRESENTATION
  inquire_edge_representation (3P)
INQUIRE EDIT MODE
  inquire_edit_mode (3P)
INQUIRE ELEMENT CONTENT
   inquire_element_content (3P)
INQUIRE ELEMENT POINTER
   inquire_element_pointer (3P)
INQUIRE ELEMENT TYPE AND SIZE
   inquire_element_type_and_size (3P)
INQUIRE ERROR HANDLING MODE
   inquire_error_handling_mode (3P)
INQUIRE GENERALIZED DRAWING PRIMITIVE
   inquire_generalized_drawing_primitive (3P)
INQUIRE GENERALIZED DRAWING PRIMITIVE 3
   inquire_generalized_drawing_primitive_3 (3P)
INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES
   inquire_generalized_structure_element_facilities (3P)
INQUIRE HIGHLIGHTING FILTER
   inquire_highlighting_filter (3P)
INQUIRE HLHSR IDENTIFIER FACILITIES
   inquire_hlhsr_identifier_facilities (3P)
INQUIRE HLHSR MODE
   inquire_hlhsr_mode_facilities (3P)
INQUIRE HLHSR MODE FACILITIES
   inquire_hlhsr_mode (3P)
INQUIRE INPUT QUEUE OVERFLOW
   inquire_input_queue_overflow (3P)
INQUIRE INTERIOR FACILITIES
   inquire_interior_facilities (3P)
INQUIRE INTERIOR REPRESENTATION
   inquire_interior_representation (3P)
INQUIRE INVISIBILITY FILTER
   inquire_invisibility_filter (3P)
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES
   inquire_list_of_available_generalized_drawing_primitives (3P)
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3
   inquire_list_of_available_generalized_drawing_primitives_3 (3P)
INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS
   inquire_list_of_available_generalized_structure_elements (3P)
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
   inquire_list_of_available_workstation_types (3P)
INQUIRE LIST OF COLOUR INDICES
  inquire_list_of_colour_indices (3P)
INQUIRE LIST OF EDGE INDICES
  inquire_list_of_edge_indices (3P)
INQUIRE LIST OF INTERIOR INDICES
  inquire_list_of_interior_indices (3P)
INQUIRE LIST OF PATTERN INDICES
  inquire_list_of_pattern_indices (3P)
INQUIRE LIST OF POLYLINE INDICES
  inquire_list_of_polyline_indices (3P)
INQUIRE LIST OF POLYMARKER INDICES
  inquire_list_of_polymarker_indices (3P)
INQUIRE LIST OF TEXT INDICES
  inquire_list_of_text_indices (3P)
INQUIRE LIST OF VIEW INDICES
  inquire_list_of_view_indices (3P)
INQUIRE LOCATOR DEVICE STATE
  inquire_locator_device_state (3P)
INQUIRE LOCATOR DEVICE STATE 3
  inquire_locator_device_state_3 (3P)
INQUIRE MODELLING CLIPPING FACILITIES
  inquire_modelling_clipping_facilities (3P)
INQUIRE MORE SIMULTANEOUS EVENTS
  inquire_more_simultaneous_events (3P)
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES
  inquire_number_of_available_logical_input_devices (3P)
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED
  inquire_number_of_display_priorities_supported (3P)
INQUIRE OPEN STRUCTURE
  inquire_open_structure (3P)
INQUIRE PATHS TO ANCESTORS
  inquire_paths_to_ancestors (3P)
INQUIRE PATHS TO DESCENDANTS
  inquire_paths_to_descendants (3P)
INQUIRE PATTERN FACILITIES
  inquire_pattern_facilities (3P)
INQUIRE PATTERN REPRESENTATION
  inquire_pattern_representation (3P)
INQUIRE PHIGS FACILITIES
  inquire_phigs_facilities (3P)
INQUIRE PICK DEVICE STATE
  inquire_pick_device_state (3P)
INQUIRE PICK DEVICE STATE 3
  inquire_pick_device_state_3 (3P)
INQUIRE POLYLINE FACILITIES
  inquire_polyline_facilities (3P)
INQUIRE POLYLINE REPRESENTATION
  inquire_polyline_representation (3P)
INQUIRE POLYLINE REPRESENTATION PLUS
  inquire_polymarker_facilities (3P)
INQUIRE POLYMARKER FACILITIES
  inquire_polymarker_facilities (3P)
INQUIRE POLYMARKER REPRESENTATION
  inquire_polymarker_representation (3P)
INQUIRE POSTED STRUCTURES
  inquire_posted_structures (3P)
INQUIRE PREDEFINED COLOUR REPRESENTATION
  inquire_predefined_colour_representation (3P)
INQUIRE PREDEFINED EDGE REPRESENTATION
  inquire_predefined_edge_representation (3P)
INQUIRE PREDEFINED INTERIOR REPRESENTATION
  inquire_predefined_interior_representation (3P)
INQUIRE PREDEFINED PATTERN REPRESENTATION
  inquire_predefined_pattern_representation (3P)
INQUIRE PREDEFINED POLYLINE REPRESENTATION
  inquire_predefined_polyline_representation (3P)
INQUIRE PREDEFINED POLYMARKER REPRESENTATION
  inquire_predefined_polymarker_representation (3P)
INQUIRE PREDEFINED TEXT REPRESENTATION
  inquire_predefined_text_representation (3P)
INQUIRE PREDEFINED VIEW REPRESENTATION
  inquire_predefined_view_representation (3P)
INQUIRE SET OF OPEN WORKSTATIONS
  inquire_set_of_open_workstations (3P)
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED
  inquire_set_of_workstations_to_which_posted (3P)
INQUIRE STRING DEVICE STATE
    inquire_string_device_state (3P)
INQUIRE STRING DEVICE STATE 3
    inquire_string_device_state_3 (3P)
INQUIRE STROKE DEVICE STATE
    inquire_stroke_device_state (3P)
INQUIRE STROKE DEVICE STATE 3
    inquire_stroke_device_state_3 (3P)
INQUIRE STRUCTURE IDENTIFIERS
    inquire_structure_identifiers (3P)
INQUIRE STRUCTURE STATE VALUE
    inquire_structure_state_value (3P)
INQUIRE STRUCTURE STATUS
    inquire_structure_status (3P)
INQUIRE SYSTEM STATE VALUE
    inquire_system_state_value (3P)
INQUIRE TEXT EXTENT
    inquire_text_extent (3P)
INQUIRE TEXT FACILITIES
    inquire_text_facilities (3P)
INQUIRE TEXT REPRESENTATION
    inquire_text_representation (3P)
INQUIRE VALUATOR DEVICE STATE
    inquire_valuator_device_state (3P)
INQUIRE VALUATOR DEVICE STATE 3
    inquire_valuator_device_state_3 (3P)
INQUIRE VIEW FACILITIES
    inquire_view_facilities (3P)
INQUIRE VIEW REPRESENTATION
    inquire_view_representation (3P)
INQUIRE WORKSTATION CATEGORY
    inquire_workstation_category (3P)
INQUIRE WORKSTATION CLASSIFICATION
    inquire_workstation_classification (3P)
INQUIRE WORKSTATION CONNECTION AND TYPE
    inquire_workstation_connection_and_type (3P)
INQUIRE WORKSTATION STATE TABLE LENGTHS
    inquire_workstation_state_table_lengths (3P)
INTRO PHIGS (3P)

INQUIRE WORKSTATION STATE VALUE
  inquire_workstation_state_value (3P)

INQUIRE WORKSTATION TRANSFORMATION
  inquire_workstation_transformation (3P)

INQUIRE WORKSTATION TRANSFORMATION 3
  inquire_workstation_transformation_3 (3P)

INTERPRET ITEM
  interpret_item (3P)

LABEL
  label (3P)

MESSAGE
  message (3P)

OFFSET ELEMENT POINTER
  offset_element_pointer (3P)

OPEN ARCHIVE FILE
  open_archive_file (3P)

OPEN PHIGS
  open_phigs (3P)

OPEN STRUCTURE
  open_structure (3P)

OPEN WORKSTATION
  open_workstation (3P)

PACK DATA RECORD
  pack_data_record (3P)

POLYLINE
  polyline (3P)

POLYLINE 3
  polyline_3 (3P)

POLYMARKER
  polymarker (3P)

POLYMARKER 3
  polymarker_3 (3P)

POST STRUCTURE
  post_structure (3P)

READ ITEM FROM METAFILE
  read_item_from_metafile (3P)

REDRAW ALL STRUCTURES
  redraw_all_structures (3P)

modified 2 April 1993
REMOVE NAMES FROM SET
    remove_names_from_set (3P)
REQUEST CHOICE
    request_choice (3P)
REQUEST LOCATOR
    request_locator (3P)
REQUEST LOCATOR 3
    request_locator_3 (3P)
REQUEST PICK
    request_pick (3P)
REQUEST STRING
    request_string (3P)
REQUEST STROKE
    request_stroke (3P)
REQUEST STROKE 3
    request_stroke_3 (3P)
REQUEST VALUATOR
    request_valuator (3P)
RESTORE MODELLING CLIPPING VOLUME
    restore_modelling_clipping_volume (3P)
RETRIEVE ALL STRUCTURES
    retrieve_all_structures (3P)
RETRIEVE PATHS TO ANCESTORS
    retrieve_paths_to_ancestors (3P)
RETRIEVE PATHS TO DESCENDANTS
    retrieve_paths_to_descendants (3P)
RETRIEVE STRUCTURE IDENTIFIERS
    retrieve_structure_identifiers (3P)
RETRIEVE STRUCTURE NETWORKS
    retrieve_structure_networks (3P)
RETRIEVE STRUCTURES
    retrieve_structures (3P)
ROTATE
    rotate (3P)
ROTATE X
    rotate_x (3P)
ROTATE Y
    rotate_y (3P)
 ROTATE Z 
  rotate_z (3P) 
 SAMPLE CHOICE 
  sample_choice (3P) 
 SAMPLE LOCATOR 
  sample Locator (3P) 
 SAMPLE LOCATOR 3 
  sample Locator_3 (3P) 
 SAMPLE PICK 
  sample_pick (3P) 
 SAMPLE STRING 
  sample_string (3P) 
 SAMPLE STROKE 
  sample_stroke (3P) 
 SAMPLE STROKE 3 
  sample_stroke_3 (3P) 
 SAMPLE VALUATOR 
  sample_valuator (3P) 
 SCALE 
  scale (3P) 
 SCALE 3 
  scale_3 (3P) 
 SET ANNOTATION STYLE 
  set_annotation_style (3P) 
 SET ANNOTATION TEXT ALIGNMENT 
  set_annotation_text_alignment (3P) 
 SET ANNOTATION TEXT CHARACTER HEIGHT 
  set_annotation_text_character_height (3P) 
 SET ANNOTATION TEXT CHARACTER UP VECTOR 
  set_annotation_text_character_up_vector (3P) 
 SET ANNOTATION TEXT PATH 
  set_annotation_text_path (3P) 
 SET CHARACTER EXPANSION FACTOR 
  set_character_expansion_factor (3P) 
 SET CHARACTER HEIGHT 
  set_character_height (3P) 
 SET CHARACTER SPACING 
  set_character_spacing (3P)
SET CHARACTER UP VECTOR
  set_character_up_vector (3P)
SET CHOICE MODE
  set_choice_mode (3P)
SET COLOUR MODE
  set_colour_model (3P)
SET COLOUR REPRESENTATION
  set_colour_representation (3P)
SET CONFLICT RESOLUTION
  set_conflict_resolution (3P)
SET DISPLAY UPDATE STATE
  set_display_update_state (3P)
SET EDGE COLOUR INDEX
  set_edge_colour_index (3P)
SET EDGE FLAG
  set_edge_flag (3P)
SET EDGE INDEX
  set_edge_index (3P)
SET EDGE REPRESENTATION
  set_edge_representation (3P)
SET EDGETYPE
  set_edgetype (3P)
SET EDGEWIDTH SCALE FACTOR
  set_edgewidth_scale_factor (3P)
SET EDIT MODE
  set_edit_mode (3P)
SET ELEMENT POINTER
  set_element_pointer (3P)
SET ELEMENT POINTER AT LABEL
  set_element_pointer_at_label (3P)
SET ERROR HANDLING
  set_error_handling (3P)
SET ERROR HANDLING MODE
  set_error_handling_mode (3P)
SET GLOBAL TRANSFORMATION
  set_global_transformation (3P)
SET GLOBAL TRANSFORMATION 3
  set_global_transformation_3 (3P)
SET HIGHLIGHTING FILTER
   set_highlighting_filter (3P)
SET HLHSR IDENTIFIER
   set_hlhsr_identifier (3P)
SET HLHSR MODE
   set_hlhsr_mode (3P)
SET INDIVIDUAL ASF
   set_individual ASF (3P)
SET INTERIOR COLOUR INDEX
   set_interior_colour_index (3P)
SET INTERIOR INDEX
   set_interior_index (3P)
SET INTERIOR REPRESENTATION
   set_interior_representation (3P)
SET INTERIOR STYLE
   set_interior_style (3P)
SET INTERIOR STYLE INDEX
   set_interior_style_index (3P)
SET INVISIBILITY FILTER
   set_invisibility_filter (3P)
SET LINETYPE
   set_linetype (3P)
SET LINEWIDTH SCALE FACTOR
   set_linewidth_scale_factor (3P)
SET LOCAL TRANSFORMATION
   set_local_transformation (3P)
SET LOCAL TRANSFORMATION 3
   set_local_transformation_3 (3P)
SET LOCATOR MODE
   set_locator_mode (3P)
SET MARKER SIZE SCALE FACTOR
   set_marker_size_scale_factor (3P)
SET MARKER TYPE
   set_marker_type (3P)
SET MODELLING CLIPPING INDICATOR
   set_modelling_clipping_indicator (3P)
SET MODELLING CLIPPING VOLUME
   set_modelling_clipping_volume (3P)
set_modelling_clipping_volume_3 (3P)
set_pattern_reference_point (3P)
set_pattern_reference_point_and_vectors (3P)
set_pattern_representation (3P)
set_pattern_size (3P)
set_pick_filter (3P)
set_pick_identifier (3P)
set_pick_mode (3P)
set_polyline_colour_index (3P)
set_polyline_index (3P)
set_polyline_representation (3P)
set_polyline_representation_plus (3P)
set_polymarker_colour_index (3P)
set_polymarker_colour_index (3P)
set_polymarker_index (3P)
set_polymarker_representation (3P)
set_rendering_colour_model (3P)
set_string_mode (3P)
set_stroke_mode (3P)
set_text_alignment (3P)
SET TEXT COLOUR INDEX
    set_text_colour_index (3P)

SET TEXT FONT
    set_text_font (3P)

SET TEXT INDEX
    set_text_index (3P)

SET TEXT PATH
    set_text_path (3P)

SET TEXT PRECISION
    set_text_precision (3P)

SET TEXT REPRESENTATION
    set_text_representation (3P)

SET VALUATOR MODE
    set_valuator_mode (3P)

SET VIEW INDEX
    set_view_index (3P)

SET VIEW REPRESENTATION
    set_view_representation (3P)

SET VIEW REPRESENTATION 3
    set_view_representation_3 (3P)

SET VIEW TRANSFORMATION INPUT PRIORITY
    set_view_transformation_input_priority (3P)

SET WORKSTATION VIEWPORT
    set_workstation_viewport (3P)

SET WORKSTATION VIEWPORT 3
    set_workstation_viewport_3 (3P)

SET WORKSTATION WINDOW
    set_workstation_window (3P)

SET WORKSTATION WINDOW 3
    set_workstation_window_3 (3P)

TEXT
    text (3P)

TEXT 3
    text_3 (3P)

TRANSFORM POINT
    transform_point (3P)

TRANSFORM POINT 3
    transform_point_3 (3P)
TRANSLATE
  translate (3P)
TRANSLATE 3
  translate_3 (3P)
UNPACK DATA RECORD
  unpack_data_record (3P)
UNPOST ALL STRUCTURES
  unpost_all_structures (3P)
UNPOST STRUCTURE
  unpost_structure (3P)
UPDATE WORKSTATION
  update_workstation (3P)
WRITE ITEM TO METAFILE
  update_workstation (3P)

SEE ALSO
  COLOUR (7P)
  PHIGS DESCRIPTION TABLE (7P)
  PHIGS TRAVERSAL STATE LIST (7P)
  PHIGS WORKSTATION DESCRIPTION TABLE (7P)
  CGM (7P)
  INTRO (3P)
  (in SunPHIGS 3.0 Sun-specific Reference Manual)
  INTRO (3PP)
  INTRO (7P)

modified 2 April 1993
NAME
ADD NAMES TO SET – create a structure element containing additions to the current name set

SYNOPSIS
C Syntax
void padd_names_set ( set )

Pint_list *set; set of names to be added

FORTRAN Syntax
SUBROUTINE pads ( N, NAMSET )
INTEGER N number of names in the set
INTEGER NAMSET(N) name set

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
The ADD NAMES TO SET function puts a structure element containing additions to the traversal-time current name set into the currently open structure according to the current edit mode. The current name set is compared during traversal to the workstation’s name set filters to determine if primitives that follow in the structure network are invisible, highlighted, and/or selectable by PICK input devices. Each name in the name set is a small non-negative integer.

If the current edit mode is INSERT, the ADD NAMES TO SET element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the ADD NAMES TO SET element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
set A pointer to a Pint_list structure containing the set of names to be added. The Pint_list data structure is defined in phigs.h as:

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

FORTRAN Input Parameters
N The number of names to be added.
NAMSET An array containing the set of N names to be added.

Execution
When traversal of a posted structure network starts, the current name set is empty. During traversal, the member names specified by the ADD NAMES TO SET element are added to the current name set by the union operation. REMOVE NAMES FROM SET elements remove names from the current name set.

modified 2 April 1993
The updated current name set applies to primitives that follow in the structure network by set-intersection with the workstation’s name set filters, which are set by SET INVISIBILITY FILTER (3P), SET HIGHLIGHTING FILTER (3P), and SET PICK FILTER (3P). Each workstation has a single invisibility filter, a single highlighting filter, and a pick filter for each PICK input device. The actual appearance of highlighting is workstation-dependent.

Each filter contains an inclusion set and an exclusion set of names both empty by default. A primitive is eligible if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. If the current name set is empty, subsequent primitives are not eligible. If the workstation’s inclusion set is empty (the default), no primitives are eligible. That is, no primitives are invisible, highlighted, or selectable by PICK input devices.

Each name is a small non-negative integer. PHIGS conformance requires support for at least 64 names; SunPHIGS supports the range 0 to 1023. The same names may be added and removed any number of times during traversal. Since the presence or absence of each name in the current name set and the workstation’s filter affects the eligibility of subsequent primitives, 1024 names provide up to 1024 different simultaneous groupings of primitives.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

REMOVE NAMES FROM SET (3P)
SET INVISIBILITY FILTER (3P)
SET HIGHLIGHTING FILTER (3P)
SET PICK FILTER (3P)
INCREMENTAL SPATIAL SEARCH (3P)
INCREMENTAL SPATIAL SEARCH 3 (3P)
NAME

ANNOTATION TEXT RELATIVE – create structure element specifying 2D annotation
text primitive

SYNOPSIS

C Syntax

void
panno_text_rel ( ref_pt, anno_offset, text )

Ppoint  *ref_pt;  reference point
Pvec   *anno_offset;  annotation offset
char   *text;  annotation text string

FORTRAN Syntax

SUBROUTINE patr ( RPX, RPY, APX, APY, CHARS )

REAL RPX, RPY  reference point (MC)
REAL APX, APY  annotation offset (NPC)
CHARACTER*() CHARS  string of characters

FORTRAN Subset Syntax

SUBROUTINE patrs ( RPX, RPY, APX, APY, LSTR, CHARS )

REAL RPX, RPY  reference point (MC)
REAL APX, APY  annotation offset (NPC)
INTEGER LSTR  length of string (in characters)
CHARACTER*80 CHARS  string of characters

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

The ANNOTATION TEXT RELATIVE subroutine puts a structure element containing the full
specification of a two-dimensional ANNOTATION TEXT RELATIVE primitive into the
currently open structure.

The ANNOTATION TEXT RELATIVE primitive is a character string. The location of the string
in the display is controlled by the reference point and annotation offset subroutine
parameters. ANNOTATION TEXT RELATIVE primitives differ from TEXT primitives in that the
characters are generated in the Normalized Projection Coordinate (NPC) space. The
reference point is specified in Modeling Coordinates (MC). The \( z \) coordinate is assumed
to be 0. Aspects of the text display such as the font, colour, spacing, height, and
alignment are controlled by the current values of the primitive attributes listed below.

If the current edit mode is INSERT, the structure element created by the ANNOTATION TEXT
RELATIVE subroutine is inserted into the open structure after the element pointed to by the
current element pointer. If the current edit mode is REPLACE, the new ANNOTATION TEXT
RELATIVE element replaces the element in the structure pointed to by the element pointer.
In either case, the element pointer is updated to point to the new ANNOTATION TEXT
RELATIVE element.
### C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref_pt</td>
<td>A pointer to a <code>Ppoint</code> structure containing the x and y MC that locate the annotation text. The <code>Ppoint</code> structure is defined in <code>phigs.h</code> as follows:</td>
</tr>
</tbody>
</table>
|           | ```
|           | typedef struct {
|           |     Pfloat x;    /* x coordinate */
|           |     Pfloat y;    /* y coordinate */
|           | } Ppoint;
|           | ``` |
| anno_offset | A pointer to a `Pvec` structure containing the x and y coordinates of the offset of the text string from the transformed reference point. The annotation offset specifies an offset in NPC. The `Pvec` structure is defined in `phigs.h` as follows: |
|           | ```
|           | typedef struct {
|           |     Pfloat delta_x; /* delta x value */
|           |     Pfloat delta_y; /* delta y value */
|           | } Pvec;
|           | ``` |
| text      | A pointer to the character string to be written into the display. |

### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPX</td>
<td>The x MC of the point locating the annotation text.</td>
</tr>
<tr>
<td>RPY</td>
<td>The y MC of the point locating the annotation text.</td>
</tr>
<tr>
<td>APX</td>
<td>The x offset in NPC locating the text string relative to RPX.</td>
</tr>
<tr>
<td>APY</td>
<td>The y offset in NPC locating the text string relative to RPY.</td>
</tr>
<tr>
<td>CHARS</td>
<td>A character array containing the string to be written into the display. It is best to use substrings or constants and to use only the portion desired. Avoid use of the blank-padded portion. An example of a substring is <code>LABEL(1:8)</code>, where label might be declared as <code>CHARACTER*256</code>. An example of a character constant is <code>abcdefg</code>. Another way to achieve the same result is to null terminate the string, that is, <code>hello\0</code>. Strings returned from SunPHIGS inquiry functions, however, will not be null terminated, even though they may have been specified that way.</td>
</tr>
</tbody>
</table>

### Execution

When the structure is traversed, the `ANNOTATION TEXT RELATIVE` element draws the specified character string on the plane in the NPC System defined by the reference point and the annotation offset. These parameters define a Text Local Coordinate (TLC) System in the NPC System. The annotation offset added to the transformed reference point defines the origin of this TLC System. The x and y axes of the TLC System are parallel to and have the same direction as the x and y axes of the NPC System.

The precise position of the text is defined in relation to this plane by the current values of the text primitive attributes `ANNOTATION TEXT CHARACTER UP VECTOR`, `ANNOTATION TEXT CHARACTER`, `BASE VECTOR`, `ANNOTATION TEXT PATH`, and `ANNOTATION TEXT ALIGNMENT`. The reference point is subject to the current transformations in the transformation pipeline from the MC System to the workstation display. The text itself is only subject to

Modified 2 April 1993
the transformations in the transformation pipeline from the NPC System to the workstation display.

Other aspects of the appearance of the text are controlled by the attributes TEXT FONT, TEXT PRECISION, ANNOTATION STYLE, ANNOTATION TEXT CHARACTER HEIGHT, ANNOTATION TEXT CHARACTER WIDTH, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, and TEXT COLOUR INDEX.

ANNOTATION TEXT CHARACTER WIDTH and ANNOTATION TEXT BASE VECTOR are implicit attributes derived from ANNOTATION TEXT CHARACTER HEIGHT and ANNOTATION TEXT UP VECTOR, respectively.

Note that ANNOTATION TEXT RELATIVE will only render ASCII text. When other character sets are required, the application should use GENERALIZED DRAWING PRIMITIVE -18.

Attributes Applied

The attributes listed below are used to display the ANNOTATION TEXT RELATIVE primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- text font
- text precision
- character expansion factor
- character spacing
- text colour
- annotation character height
- annotation character up vector
- annotation text path
- annotation text alignment
- annotation style
- text index
- depth cue index
- name set

The polyline attributes are applied to the Lead Line (when the annotation style is Lead Line). These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- polyline colour
- linewidth scale factor
- linetype
- polyline shading method
- polyline index
- polyline colour index
- linewidth scale factor
- linetype
- polyline shading method
- polyline index
ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
TEXT (3P)
INTRO INTERNATIONALIZATION (7P)

modified 2 April 1993
NAME

ANNOTATION TEXT RELATIVE 3 – create structure element specifying 3D annotation text primitive

SYNOPSIS

C Syntax

```c
void
panno_text_rel3 ( ref_pt, anno_offset, text )
Ppoint3 *ref_pt; reference point
Pvec3 *anno_offset; annotation offset
char *text; annotation text string
```

FORTRAN Syntax

```fortran
SUBROUTINE patr3 ( RPX, RPY, RPZ, APX, APY, APZ, CHARS )
REAL RPX, RPY, RPZ reference point (MC)
REAL APX, APY, APZ annotation offset (NPC)
CHARACTER*80 CHARS string of characters
```

FORTRAN Subset Syntax

```fortran
SUBROUTINE patr3s ( RPX, RPY, RPZ, APX, APY, APZ, LSTR, CHARS )
REAL RPX, RPY, RPZ reference point (MC)
REAL APX, APY, APZ annotation offset (NPC)
INTEGER LSTR length of string (in characters)
CHARACTER*80 CHARS string of characters
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

The ANNOTATION TEXT RELATIVE 3 subroutine puts a structure element containing the full specification of a three-dimensional ANNOTATION TEXT RELATIVE 3 primitive into the currently open structure.

The ANNOTATION TEXT RELATIVE 3 primitive is a character string. The location of the string in the display is controlled by the reference point and annotation offset subroutine parameters. ANNOTATION TEXT RELATIVE 3 primitives differ from TEXT 3 primitives in that the reference point is specified in Modelling Coordinates (MC), and the plane upon which the characters are generated will always be parallel to the display surface and in the Normalized Projection Coordinate (NPC) Space. Aspects of the text display such as the font, colour, spacing, height, and alignment are controlled by the current values of the primitive attributes listed below.

If the current edit mode is INSERT, the structure element created by the ANNOTATION TEXT RELATIVE 3 subroutine is inserted into the open structure after the element pointed to by the current element pointer. If the current edit mode is REPLACE, the new ANNOTATION TEXT RELATIVE 3 element replaces the element in the structure pointed to by the element pointer. In either case, the element pointer is updated to point to the new ANNOTATION TEXT RELATIVE 3 element.

modified 2 April 1993
C Input Parameters

`ref_pt` A pointer to a Ppoint3 structure containing the x, y, and z MC that locate the annotation text. The Ppoint3 structure is defined in phigs.h as follows:

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

`anno_offset` A pointer to a Pvec3 structure containing the x, y, and z coordinates of the offset of the text string from the transformed reference point. The annotation offset specifies an offset in NPC. The Pvec3 structure is defined in phigs.h as follows:

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

`text` A pointer to the character string to be written into the display.

FORTRAN Input Parameters

`RPX` The x MC of the point locating the annotation text.

`RPY` The y MC of the point locating the annotation text.

`RPZ` The z MC of the point locating the annotation text.

`APX` The x offset in NPC locating the text string relative to `RPX`.

`APY` The y offset in NPC locating the text string relative to `RPY`.

`APZ` The z offset in NPC locating the text string relative to `RPZ`.

`CHARS` A character array containing the string to be written into the display.

It is best to use substrings or constants and to use only the portion desired. Avoid use of the blank-padded portion. An example of a substring is `LABEL(1:8)`, where `label` might be declared as `CHARACTER*256`. An example of a character constant is `abcdefg`. Another way to achieve the same result is to null terminate the string, that is, `hello\0`. Strings returned from SunPHIGS inquiry functions, however, will not be null terminated, even though they may have been specified that way.

Execution

When the structure is traversed, the `ANNOTATION TEXT RELATIVE 3` element draws the specified character string on the plane in the NPC System defined by reference point and the annotation offset. These parameters define a Text Local Coordinate (TLC) System in the NPC System. The annotation offset added to the transformed reference point defines the origin of this TLC System. The x and y axes of the TLC System are parallel to and have the same direction as the x and y axes of the NPC System.

modified 2 April 1993
The precise position of the text is defined in relation to this plane by the current values of the text primitive attributes ANNOTATION TEXT CHARACTER UP VECTOR, ANNOTATION TEXT CHARACTER, BASE VECTOR, ANNOTATION TEXT PATH, and ANNOTATION TEXT ALIGNMENT. The reference point is subject to the current transformations in the transformation pipeline from the MC System to the workstation display. The text itself is subject only to the transformations in the transformation pipeline from the NPC System to the workstation display.

Other aspects of the appearance of the text are controlled by the attributes TEXT FONT, TEXT PRECISION, ANNOTATION STYLE, ANNOTATION TEXT CHARACTER HEIGHT, ANNOTATION TEXT CHARACTER WIDTH, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, and TEXT COLOUR INDEX.

ANNOTATION TEXT CHARACTER WIDTH and ANNOTATION TEXT BASE VECTOR are implicit attributes derived from ANNOTATION TEXT CHARACTER HEIGHT and ANNOTATION TEXT UP VECTOR, respectively.

Note that ANNOTATION TEXT RELATIVE 3 will only render ASCII text. When other character sets are required, the application should use GENERALIZED DRAWING PRIMITIVE 3-18 (3P).

**Attributes Applied**

The attributes listed below are used to display the ANNOTATION TEXT RELATIVE 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- text font
- text precision
- character expansion factor
- character spacing
- text colour
- annotation character height
- annotation character up vector
- annotation text path
- annotation text alignment
- annotation style
- text index
- depth cue index
- name set

The polyline attributes are applied to the Lead Line (when the annotation style is Lead Line). These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- polyline colour
- linewidth scale factor
- linetype

SunPHIGS Release 3.0

30 modified 2 April 1993
polyline shading method       polyline shading method ASF
polyline index

ERRORS

005   Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
INTRO INTERNATIONALIZATION (7P)
TEXT 3 (3P)

modified 2 April 1993
NAME APPLICATION DATA – create a structure element containing application data

SYNOPSIS
C Syntax
void pappl_data ( data )
Pdata *data;  // application data

FORTRAN Syntax
SUBROUTINE pap ( LDR, DATREC )
INTEGER LDR  // dimension of data record array
CHARACTER*80 DATREC(LDR)  // data record

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
The APPLICATION DATA subroutine puts a structure element containing application data into the currently open structure.

If the current edit mode is INSERT, then the APPLICATION DATA element is inserted into the currently open structure after the element currently pointed to by the element pointer. If the edit mode is REPLACE, then the APPLICATION DATA element replaces the element to which the element pointer points. In either case, the element pointer is updated to point to the new APPLICATION DATA element.

C Input Parameter
data A pointer to a Pdata structure containing the application data. Pdata is defined in phigs.h as follows:
typedef struct {
    size_t size;  // size of data
    void *data;   // pointer to data
} Pdata;

The size component specifies the number of bytes to which the data component points.

FORTRAN Input Parameters
LDR The dimension of the DATREC array.
DATREC A character array containing the application data.

Execution
The APPLICATION DATA element is ignored during structure traversal.

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)
SEE ALSO

INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
INQUIRE CURRENT ELEMENT CONTENT (3P)
INQUIRE ELEMENT TYPE AND SIZE (3P)
INQUIRE ELEMENT CONTENT (3P)
## NAME
ARCHIVE ALL STRUCTURES – archive all structures into an archive file

## SYNOPSIS

### C Syntax

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

### FORTRAN Syntax

```fortran
SUBROUTINE parast ( AFID )
INTEGER AFID    /* archive file identifier */
```

### Required PHIGS Operating States

(PHOP, *, *, AROP)

## DESCRIPTION

### Purpose
Use ARCHIVE ALL STRUCTURES to archive all structures in the Central Structure Store (CSS) to the specified open archive file.

### C Input Parameter

*archive_id*
     The archive identifier specifying the open archive file to write to.

### FORTRAN Input Parameter

*AFID*
     The archive identifier specifying the open archive file to write to.

### Execution

All structures in the CSS are copied to the indicated open archive file. If any of the structures to be archived currently exist in the archive file, the conflict is resolved as follows:

- If the archival conflict resolution flag is MAINTAIN, the conflicting structure will not be copied into the archive (the archive contents are maintained).
- If the archival conflict resolution flag is UPDATE, the conflicting structure in the archive will be overwritten (the archive contents are updated).
- If the archival conflict resolution flag is ABANDON, no structures will be copied to the archive at all. The archival conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine.

## ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>Ignoring function, function requires state (PHOP, *, *, AROP)</td>
</tr>
<tr>
<td>404</td>
<td>Ignoring function, the specified archive file is not open</td>
</tr>
<tr>
<td>405</td>
<td>Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON</td>
</tr>
<tr>
<td>406</td>
<td>Warning, the archive file is full. Any structures that were archived were archived in total</td>
</tr>
</tbody>
</table>

34 modified 2 April 1993
SEE ALSO

OPEN ARCHIVE FILE (3P)
SET CONFLICT RESOLUTION (3P)
ARCHIVE STRUCTURE NETWORKS (3P)
ARCHIVE STRUCTURES (3P)
DELETE ALL STRUCTURES FROM ARCHIVE (3P)
NAME
ARCHIVE STRUCTURE NETWORKS – archive specified structure networks into an archive file

SYNOPSIS
C Syntax
void
par_struct_nets ( archive_id, struct_ids )
Pint archive_id; // archive identifier
Pint_list struct_ids; // list of structure identifiers

FORTRAN Syntax
SUBROUTINE parsn ( AFID, N, LSTRID )
INTEGER AFID // archive file identifier
INTEGER N // number of structure identifiers in the list
INTEGER LSTRID(N) // list of structure identifiers

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use ARCHIVE STRUCTURE NETWORKS to archive a list of structure networks in the Central Structure Store (CSS) to the specified open archive file.

C Input Parameters
archive_id
The archive identifier specifying the open archive file to write to.

struct_ids
A pointer to a Pint_list structure containing the list of the root structure identifiers of the networks to be archived. The Pint_list structure is defined in phigs.h as follows:

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

The num_ints component specifies the number of structure identifiers in the list. The ints component is a pointer to a list, num_ints long, of the structure identifiers.

FORTRAN Input Parameters
AFID The archive identifier specifying the open archive file to write to.
N The number of structure networks to be archived.
LSTRID An array of integers containing the root structure identifiers of the networks to be archived.

Execution
The structures belonging to the specified networks are copied from the CSS to the indicated open archive file. If any of the structures to be archived currently exist in the archive file, the conflict is resolved as follows:

modified 2 April 1993
If the archival conflict resolution flag is MAINTAIN, the conflicting structure will not be copied into the archive (the archive contents are maintained).

- If the archival conflict resolution flag is UPDATE, the conflicting structure in the archive will be overwritten (the archive contents are updated).
- If the archival conflict resolution flag is ABANDON, no structures will be copied to the archive at all.

The archival conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine. If any of the structures to be archived do not exist in the Central Structure Store, a warning is generated, and the archiving operation continues for the remaining structures.

**ERRORS**

- **007** Ignoring function, function requires state (PHOP, *, *, AROP)
- **404** Ignoring function, the specified archive file is not open
- **200** Warning, ignoring structures that do not exist
- **405** Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON
- **406** Warning, the archive file is full. Any structures that were archived were archived in total

**SEE ALSO**

- OPEN ARCHIVE FILE (3P)
- SET CONFLICT RESOLUTION (3P)
- ARCHIVE ALL STRUCTURES (3P)
- ARCHIVE STRUCTURES (3P)
- DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
NAME
ARCHIVE STRUCTURES – archive specified structures into an archive file

SYNOPSIS
C Syntax
void
par_structs ( archive_id, struct_ids )
Pint archive_id; // archive identifier
Pint_list *struct_ids; // list of structure identifiers

FORTRAN Syntax
SUBROUTINE parst ( AFID, N, LSTRID )
INTEGER AFID // archive file identifier
INTEGER N // number of structure identifiers in the list
INTEGER LSTRID(N) // list of structure identifiers

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use ARCHIVE STRUCTURES to archive a list of structures in the Central Structure Store (CSS) to the specified open archive file.

C Input Parameters
archive_id
The archive identifier specifying the open archive file to write to.
struct_ids
A pointer to a Pint_list structure containing the list of structure identifiers to be archived. The Pint_list structure is defined in phigs.h as follows:

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

The num_ints component specifies the number of structure identifiers in the list. The ints component is a pointer to a list, num_ints long, of the structure identifiers.

FORTRAN Input Parameters
AFID // The archive identifier specifying the open archive file to write to.
N // The number of structure identifiers to be archived.
LSTRID // An array of integers containing the structure identifiers to be archived.

Execution
The specified structures are copied from the CSS to the indicated open archive file. If any of the structures to be archived currently exist in the archive file, the conflict is resolved as follows:

38
modified 2 April 1993
• If the archival conflict resolution flag is MAINTAIN, the conflicting structure will not be copied into the archive (the archive contents are maintained)

• If the archival conflict resolution flag is UPDATE, the conflicting structure in the archive will be overwritten (the archive contents are updated)

• If the archival conflict resolution flag is ABANDON, no structures will be copied to the archive at all

The archival conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine.

If any of the structures to be archived do not exist in the CSS, a warning is generated and the archiving operation continues for the remaining structures.

ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>Ignoring function, function requires state (PHOP, *, *, AROP)</td>
</tr>
<tr>
<td>404</td>
<td>Ignoring function, the specified archive file is not open</td>
</tr>
<tr>
<td>200</td>
<td>Warning, ignoring structures that do not exist</td>
</tr>
<tr>
<td>405</td>
<td>Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON</td>
</tr>
<tr>
<td>406</td>
<td>Warning, the archive file is full. Any structures that were archived were archived in total</td>
</tr>
</tbody>
</table>

SEE ALSO

OPEN ARCHIVE FILE (3P)
SET CONFLICT RESOLUTION (3P)
ARCHIVE STRUCTURE NETWORKS (3P)
ARCHIVE ALL STRUCTURES (3P)
DELETE STRUCTURES FROM ARCHIVE (3P)
NAME

AWAIT EVENT – move event from input queue to current event report

SYNOPSIS

C Syntax

```c
void pawait_event ( timeout, ws_id, class, in_num )
```

C Input Parameters

- `timeout`: The time, in seconds, to wait for an event if none are present in the input queue.

C Output Parameters

- `ws_id`: If the event queue is not empty, PHIGS sets the variable pointed to by `ws_id` to the identifier of the workstation that was the source of the input event. If the queue is empty, PHIGS does not set this variable.
- `class`: If the event queue is not empty, PHIGS sets the variable pointed to by `class` to the class of the device that generated the input event. If the queue is empty, PHIGS sets this variable to PIN_NONE. `Pin_class` is an enumerated type that may take the following values:

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

FORTRAN Syntax

```fortran
SUBROUTINE pwait ( TOUT, WKID, ICL, IDNR )
```

FORTRAN Parameters

- `TOUT`: Time out (seconds)
- `WKID`: Workstation identifier
- `ICL`: Input class
- `IDNR`: Logical input device number

DESCRIPTION

Purpose

AWAIT EVENT makes an event available to the application program by moving an event from the input event queue to the current event report in the PHIGS state list.

If the input queue is empty, AWAIT EVENT suspends PHIGS until an event is entered into the queue or the time specified by `timeout` has elapsed.

Required PHIGS Operating States

- PHOP
- WSOP
- *
- *

modified 2 April 1993
AWAIT EVENT

in_num  A pointer to an integer variable. If the event queue is not empty, PHIGS sets this variable to the identifier of the input device that generated the input event.

TOUT  The time, in seconds, to wait for an event if none are present in the input queue.

FORTRAN Input Parameters

FORTRAN Output Parameters

WKID  The identifier of the workstation that generated the event.

ICL  The class of the input device that generated the event. Valid classes as defined in phigs77.h are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PNCLAS</td>
</tr>
<tr>
<td>1</td>
<td>PLOCAT</td>
</tr>
<tr>
<td>2</td>
<td>PSTROK</td>
</tr>
<tr>
<td>3</td>
<td>PVALUA</td>
</tr>
<tr>
<td>4</td>
<td>PCHOIC</td>
</tr>
<tr>
<td>5</td>
<td>PPICK</td>
</tr>
<tr>
<td>6</td>
<td>PSTRIN</td>
</tr>
</tbody>
</table>

PNCLAS is returned if a timeout occurs before an event is added to the input queue.

IDNR  The device number of the input device that generated the event.

Execution

The AWAIT EVENT function writes the logical input value of the oldest input event in the input queue to the current event report in the PHIGS state list. The workstation id, device class, and device number of the input device that generated the event are returned in the output parameters. The current event report is accessed by the appropriate GET device class function; for example, GET LOCATOR or GET STRING.

If there are no events in the input queue when AWAIT EVENT is called, PHIGS is placed in a wait state until an event is added to the queue or until the time specified by the timeout parameter has elapsed. If a timeout occurs, NONE is returned as the device class. The other two values, workstation id and device number, are undefined.

While PHIGS is in a wait state, window system events such as repaints, resizes, and so on are processed as usual.

While timeout can be any positive floating-point value, the effective timeout resolution is approximately 2 hundredths of a second.

The device class NONE is returned immediately if there are no events in the input queue.

This function is available on X Tool workstations only.

ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>256</td>
<td>Warning, the input queue has overflowed</td>
</tr>
</tbody>
</table>

SEE ALSO

SET VALUATOR MODE (3P)
SET STROKE MODE (3P)
SET STRING MODE (3P)

modified 2 April 1993
AWAIT EVENT (3P) SunPHIGS Release 3.0

SET PICK MODE (3P)
SET LOCATOR MODE (3P)
SET CHOICE MODE (3P)
GET CHOICE (3P)
GET LOCATOR (3P)
GET LOCATOR 3 (3P)
GET PICK (3P)
GET STRING (3P)
GET STROKE (3P)
GET STROKE 3 (3P)
GET VALUATOR (3P)
INQUIRE MORE SIMULTANEOUS EVENTS (3P)
INQUIRE INPUT QUEUE OVERFLOW (3P)

modified 2 April 1993
NAME
BUILD TRANSFORMATION MATRIX — generate a 2D transformation matrix to perform a transformation specified by a shift vector, rotation angle, and scale factors relative to a specified fixed point.

SYNOPSIS
C Syntax
void pbuild_tran_matrix ( pt, shift, angle, scale, error_ind, matrix );

FORTRAN Syntax
SUBROUTINE pbltm ( X0, Y0, DX, DY, PHI, FX, FY, ERRIND, XFRMT );

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use BUILD TRANSFORMATION MATRIX to calculate the 2D homogeneous \((3 \times 3)\) transformation matrix that performs the transformation specified by the input parameters.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION or SET GLOBAL TRANSFORMATION to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameters

- **pt**
  - *pt* points to the Ppoint structure containing the \(x\) and \(y\) coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point. Ppoint is defined in phigs.h as follows:

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

- **shift**
  - *shift* points to a Pvec structure containing the \(x\) and \(y\) coordinates that define the shift (translation) applied by the transformation. Pvec is defined in phigs.h as follows:

  ```c
  ```
typedef struct {
    Pfloat delta_x; /* x magnitude */
    Pfloat delta_y; /* y magnitude */
} Pvec;

angle  Angle of rotation, in radians, applied by the transformation. A positive angle is a counterclockwise rotation; a negative angle is clockwise rotation.

scale  scale points to a Pvec structure containing \( x \) and \( y \) values defining the scale factors to be applied by the transformation.

C Output Parameters

error_ind  error_ind points to the location that stores the error number for any error detected by this function.

matrix  A \( 3 \times 3 \) homogeneous transformation matrix that performs the transformation defined by the input parameters. The matrix is returned in a Pmatrix array defined in phigs.h as follows:

typedef Pfloat Pmatrix[3][3];

FORTRAN Input Parameters

X0, Y0  The \( x \) and \( y \) coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point.

DX, DY  The \( x \) and \( y \) coordinates defining the shift (translation) to be applied by the transformation.

PHI  The angle of rotation, in radians, to be applied by the transformation. A positive angle is a counterclockwise rotation; a negative angle is clockwise rotation.

FX, FY  The \( x \) and \( y \) values defining the scale factors to be applied by the transformation.

FORTRAN Output Parameters

ERRIND  The error number of any error detected by this function.

XFRMT  A \( 3 \times 3 \) homogeneous transformation matrix that performs the transformation defined by the input parameters.

Execution

BUILD TRANSFORMATION MATRIX returns \( 2D \) (\( 3 \times 3 \)) homogeneous transformation matrix that performs the transformation specified by the input parameters.

The transformation is performed in the following order:

- scale
- rotate
- shift

Scaling and rotation are done in relation to the fixed point.

ERRORS

002  Ignoring function, function requires state (PHOP, *, *, *)
SEE ALSO

- SET LOCAL TRANSFORMATION (3P)
- SET GLOBAL TRANSFORMATION (3P)
- COMPOSE TRANSFORMATION MATRIX (3P)
- COMPOSE MATRIX (3P)
NAME
BUILD TRANSFORMATION MATRIX 3 – generate a 3D transformation matrix to perform a transformation specified by a shift vector, rotation angles, and scale factors relative to a specified fixed point

SYNOPSIS
C Syntax

```c
void pbuild_tran_matrix3 ( pt, shift, x_angle, y_angle, z_angle, scale, error_ind, matrix )
```

Ppoint3 *pt; // fixed point
Pvec3 *shift; // shift vector
Pfloat x_angle; // rotation angle X
Pfloat y_angle; // rotation angle Y
Pfloat z_angle; // rotation angle Z
Pvec3 *scale; // scale vector
Pint *error_ind; // OUT error indicator
Pmatrix3 matrix; // OUT transformation matrix

FORTRAN Syntax

```fortran
SUBROUTINE pbltm3 ( X0, Y0, Z0, DX, DY, DZ, PHIX, PHIY, PHIZ, FX, FY, FZ, ERRIND, XFRMT )
```

REAL X0, Y0, Z0 // fixed point
REAL DX, DY, DZ // shift vector
REAL PHIX, PHIY, PHIZ // rotation angles (radians)
REAL FX, FY, FZ // scale factor vector
INTEGER ERRIND // OUT error indicator
REAL XFRMT(4, 4) // OUT transformation matrix

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose
Use BUILD TRANSFORMATION MATRIX 3 to build a 3D homogeneous (4 × 4) transformation matrix that performs the transformation specified by the input parameters. The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation applied to output primitives during traversal.

C Input Parameters

pt A pointer to a Ppoint3 structure containing the x, y, and z coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point. The Ppoint3 structure is defined in phigs.h as follows:

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

modified 2 April 1993
**shift** A pointer to a Pvec3 structure containing x, y, and z coordinates defining the shift (translation) to be applied by the transformation. A Pvec3 structure is defined in phigs.h as follows:

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

**x_angle, y_angle, z_angle**

The angles, in radians, of rotation around the x, y, and z axes to be applied by the transformation. Positive angles specify counterclockwise rotations; negative angles specify clockwise rotations.

**scale** A pointer to a Pvec3 structure containing x, y, and z values defining the scale factors to be applied by the transformation.

---

**C Output Parameters**

- **error_ind**
  The error number of any error detected by this function.

- **matrix**
  A 4 x 4 homogeneous transformation matrix that performs the transformation defined by the input parameters. The matrix is returned in a Pmatrix array defined in phigs.h as follows:
  ```c
  typedef Pfloat Pmatrix3[4][4];
  ```

---

**FORTRAN Input Parameters**

- **X0, Y0, Z0**
  The x, y, and z coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point.

- **DX, DY, DZ**
  The x, y, and z coordinates defining the shift (translation) to be applied by the transformation.

- **PHIX, PHIY, PHIZ**
  The angles, in radians, of rotation around the x, y, and z axes to be applied by the transformation. Positive angles specify counterclockwise rotations; negative angles specify clockwise rotations.

- **FX, FY, FZ**
  The x, y, and z values defining the scale factors to be applied by the transformation.

---

**FORTRAN Output Parameters**

- **ERRIND**
  The error number of any error detected by this function.

- **XFRMT**
  A 4 x 4 homogeneous transformation matrix that performs the transformation defined by the input parameters.

---

modified 2 April 1993
Execution

BUILD TRANSFORMATION MATRIX 3 returns a 3D homogeneous (4 × 4) transformation matrix which performs the transformation specified by the values of the input parameters. The transformation is performed in the following order:

- scale
- rotate
- shift

Scaling and rotation are done in relation to the fixed point defined by pt.

ERRORS

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

SEE ALSO

SET LOCAL TRANSFORMATION 3 (3P)
SET GLOBAL TRANSFORMATION 3 (3P)
COMPOSE TRANSFORMATION MATRIX 3 (3P)
COMPOSE MATRIX 3 (3P)
NAME

CELL ARRAY – create structure element specifying 2D cell array

SYNOPSIS

C Syntax

```c
void pcell_array ( rectangle, colr_array )
```

```c
Prect *rectangle;   // cell rectangle
Ppat_rep *colr_array; // colour array
```

FORTRAN Syntax

```fortran
SUBROUTINE pca ( PX, PY, QX, QY, DIMX, DIMY, ISC, ISR, DX, DY, COLIA )
```

```fortran
REAL PX, PY, QX, QY    // two points (P, Q) (MC)
INTEGER DIMX, DIMY    // the dimensions of COLIA, which contains
                       // the cell array
INTEGER ISC, ISR      // indices of start column, start row
INTEGER DX, DY        // number of columns, number of rows
INTEGER COLIA(DIMX, DIMY) // colour index array
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

The CELL ARRAY function places a structure element containing the full specification of a 2D cell array into the currently open structure, according to the current edit mode. A 2D cell array primitive is an array of cells with individual colours. Its area is defined by two points in Modelling Coordinates (MC) giving the opposite corners of a rectangle, which is taken to be aligned with the modelling coordinate axes.

If the current edit mode is INSERT, the CELL ARRAY element is inserted into the open structure after the element pointed to by the element pointer. If the edit mode is REPLACE, the CELL ARRAY element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new CELL ARRAY element.

Note: SunPHIGS supports the minimal allowed simulation for the CELL ARRAY primitive by drawing the transformed boundaries of the cell rectangle.

C Input Parameters

`rectangle`

A pointer to a Prect structure giving the lower left and upper right corners of the rectangle that defines the area of the CELL ARRAY element. The Prect structure is defined in phigs.h as:

```c
typedef struct {
    Ppoint p;  // lower left
    Ppoint q;  // upper right
} Prect;
```

The Ppoint structure is defined in phigs.h as:

modified 2 April 1993
typedef struct {
    Pfloat x; /* x coordinate */
    Pfloat y; /* y coordinate */
} Ppoint;

colr_array
A pointer to the colour array. The Ppat_rep structure is defined in phigs.h as:

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

Pint_size is defined in phigs.h as follows:

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

FORTRAN Input Parameters

PX The x coordinate of the point giving the position of one corner of the rectangle that defines the area of the CELL ARRAY element.

PY The y coordinate of the point giving the position of one corner of the rectangle that defines the area of the CELL ARRAY element.

QX The x coordinate of the point giving the position of the opposite corner of the rectangle that defines the area of the CELL ARRAY element.

QY The y coordinate of the point giving the position of the opposite corner of the rectangle that defines the area of the CELL ARRAY element.

DIMX The x dimension, or number of columns, of the entire COLIA array.

DIMY The y dimension, or number of rows, of the entire COLIA array.

ISC The application may specify a portion of the COLIA array by indicating a starting position and number of rows and columns of the sub-array. ISC is the x coordinate, or start column, of the sub-array. To pass the entire array, this value should be one.

ISR The y coordinate, or start row, of the sub-array. To pass the entire array, this value should be one.

DX The x dimension, or number of columns, of the sub-array. To pass the entire array, this value should be the same as that of DIMX.

DY The y dimension, or number of rows, of the sub-array. To pass the entire array, this value should be the same as that of DIMY.

COLIA An array of integers containing the colour indices specifying cell colours.
**Execution**

When the structure is traversed, the cell array element draws the transformed boundaries of the cell rectangle, using the polyline attributes currently in effect. This is the minimal simulation for cell arrays.

**Note:** Applications should not depend on this behavior to remain the same in future releases.

The aspect source flags (ASFs) for line type, line width scale factor, and polyline colour index control whether the values used for these attributes are taken from the polyline representation specified by POLYLINE INDEX (ASF = BUNDLED), or from the individual specifications set for these attributes (ASF = INDIVIDUAL).

The coordinates used to specify the position of the cell array primitive are MC. These may be any coordinate units that are convenient to the application. At traversal, these coordinate values will be transformed by the current Local and Global Modelling Transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

**Attributes Applied**

The current values of the following attributes will be used to display the CELL ARRAY primitive when the structure is traversed. The value of an attribute is modified with the appropriate SET routine. The aspect source flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- polyline colour
- polyline colour index ASF
- linewidth scale factor
- linewidth scale factor ASF
- linetype
- linetype ASF
- polyline shading method
- polyline shading method ASF
- polyline index
- depth cue index
- name set

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)
113 Ignoring function, the colour index value is less than zero
117 Ignoring function, one of the dimensions of the colour index array is less than zero

**SEE ALSO**

CELL ARRAY 3 (3P)
**NAME**

CELL ARRAY 3 – create structure element specifying 3D cell array

**SYNOPSIS**

**C Syntax**

```c
void pcell_array3 ( parallelogram, colr_array )
pat_rep: colr_array; colour array
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pca3 ( CPXA, CPYA, CPZA, DIMX, DIMY, ISC, ISR, DX, DY, COLIA )
REAL CPXA(3), CPYA(3), CPZA(3)
cell parallelogram (P, Q, R) (MC)
INTEGER DIMX, DIMY
the dimensions of COLIA which contains the cell array
INTEGER ISC, ISR
indices of start column, start row
INTEGER DX, DY
number of columns, number of rows
INTEGER COLIA(DIMX, DIMY)
colour index array
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

The CELL ARRAY 3 function places a structure element containing the full specification of a 3D cell array into the currently open structure, according to the current edit mode. A 3D cell array primitive is an array of cells with individual colours. Its area is defined by three points P, Q, and R in Modelling Coordinates (MC) giving the corners of a parallelogram as P, Q, R, and \((Q + R - P)\).

If the current edit mode is INSERT, the CELL ARRAY 3 element is inserted into the open structure after the element pointed to by the element pointer. If the edit mode is REPLACE, the CELL ARRAY 3 element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new CELL ARRAY 3 element.

**Note:** SunPHIGS supports the minimal allowed simulation for the CELL ARRAY 3 primitive by drawing the transformed boundaries of the cell parallelogram.

**C Input Parameters**

parallellogram

A pointer to a Pparal structure giving the three corners of the parallelogram defining the area of the CELL ARRAY 3 element. The Pparal structure is defined in phigs.h as follows:

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

Ppoint3 is defined in phigs.h as follows:

```
typedef struct {
    Ppoint3  p; /* point p */
    Ppoint3  q; /* point q */
    Ppoint3  r; /* point r */
} Ppoint3;
```
typedef struct {
  Pfloat x; /* x coordinate */
  Pfloat y; /* y coordinate */
  Pfloat z; /* z coordinate */
} Ppoint3;

Ppoint3

\textit{colr\_array}

The colour array. The Ppat_rep structure is defined in phigs.h as follows:

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

Pint_size is defined in phigs.h as follows:

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

\textbf{FORTRAN Input Parameters}

\textbf{CPXA} The \textit{x} coordinates of the three corners of the parallelogram that define the area of the \texttt{CELL ARRAY 3} element.

\textbf{CPYA} The \textit{y} coordinates of the three corners of the parallelogram that define the area of the \texttt{CELL ARRAY 3} element.

\textbf{CPZA} The \textit{z} coordinates of the three corners of the parallelogram that define the area of the \texttt{CELL ARRAY 3} element.

\textbf{DIMX} The \textit{x} dimension, or number of columns, of the entire \texttt{COLIA} array.

\textbf{DIMY} The \textit{y} dimension, or number of rows, of the entire \texttt{COLIA} array.

\textbf{ISC} The application may specify a portion of the \texttt{COLIA} array by indicating a starting position and number of rows and columns of the sub-array. ISC is the \textit{x} coordinate, or start column, of the sub-array. To pass the entire array, this value should be 1.

\textbf{ISR} The \textit{y} coordinate, or start row, of the sub-array. To pass the entire array, this value should be 1.

\textbf{DX} The \textit{x} dimension, or number of columns, of the sub-array. To pass the entire array, this value should be the same as that of \texttt{DIMX}.

\textbf{DY} The \textit{y} dimension, or number of rows, of the sub-array. To pass the entire array, this value should be the same as that of \texttt{DIMY}.

\textbf{COLIA} An array of integers containing the colour indices specifying cell colours.

\textit{modified 2 April 1993}
**Execution**

When the structure is traversed, the cell array 3 element draws the transformed boundaries of the cell parallelogram, using the polyline attributes currently in effect. The aspect source flags (ASFs) for line type, line width scale factor, and polyline colour index control whether the values used for these attributes are taken from the polyline representation specified by POLYLINE INDEX (ASF = BUNDLED), or from the individual specifications set for these attributes (ASF = INDIVIDUAL). This is the minimal simulation for cell arrays.

**Note:** Applications should not depend on this behaviour to remain the same in future releases.

The coordinates used to specify the position of the cell array 3 primitive are Modelling Coordinates. These may be any coordinate units that are convenient to the application. At traversal, these coordinate values will be transformed by the current Local and Global Modelling Transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted. For more information.

**Attributes Applied**

The current values of the following attributes will be used to display the CELL ARRAY primitive when the structure is traversed. The value of an attribute is modified with the appropriate SET routine. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- polyline colour
- polyline colour index ASF
- linewidth scale factor
- linewidth scale factor ASF
- linetype
- linetype ASF
- polyline shading method
- polyline shading method ASF
- polyline index
- depth cue index
- name set

**ERRORS**

- 005 Ignoring function, function requires state (PHOP, *, STOP, *)
- 113 Ignoring function, the colour index value is less than zero
- 117 Ignoring function, one of the dimensions of the colour index array is less than zero

**SEE ALSO**

CELL ARRAY (3P)
**NAME**

CHANGE STRUCTURE IDENTIFIER – change the identifier assigned to a structure

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pcstid ( OLDSID, NEWSID )
INTEGER OLDSID      original structure identifier
INTEGER NEWSID      resulting structure identifier
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use CHANGE STRUCTURE IDENTIFIER to change the identifier of a specified structure. Subroutines that create new structures, for example OPEN STRUCTURE or EXECUTE STRUCTURE, include a parameter to assign an identifier for the newly created structure. The identifier is the number used to reference the structure in the Central Structure Store (CSS).

**C Input Parameters**

- `orig_struct_id`
  - Specifies the structure for which the identifier is to be changed.
- `result_struct_id`
  - Specifies the new identifier to assign to the structure.

**FORTRAN Input Parameters**

- `OLDSID`
  - Specifies the structure for which the identifier is to be changed.
- `NEWSID`
  - Specifies the identifier assigned to this structure when CHANGE STRUCTURE IDENTIFIER returns.

**Execution**

The CHANGE STRUCTURE IDENTIFIER subroutine changes the identifier associated with `original structure identifier` to `resulting structure identifier`. The subroutine does not change any references that may exist in other structures to the original or the resulting structure. The result of CHANGE STRUCTURE IDENTIFIER varies with the state of the original and resulting structures when the subroutine is called. The possible effects on the structures are described below.

modified 2 April 1993
The Original Structure
If there are no references to the original structure anywhere in the CSS, CHANGE STRUCTURE IDENTIFIER deletes the original structure identifier from the CSS. If there are references to the original structure, the original structure continues to exist after the subroutine returns, but it will be empty.

If original structure identifier is the open structure when CHANGE STRUCTURE IDENTIFIER is called, it continues to exist as the open structure after the subroutine returns, but it will be empty. The element pointer will be set to 0.

If the original structure is posted, it remains posted after CHANGE STRUCTURE IDENTIFIER, but it remains as an empty structure. The display reflects the change in the posted structure according to the workstation’s display update state.

The Resulting Structure
The resulting structure always exists at the end of CHANGE STRUCTURE IDENTIFIER. If the resulting structure identifier does not exist when the subroutine is called, it is created and contains the elements of the original structure. If the original structure identifier does not exist when the subroutine is called, the resulting structure is empty.

If the resulting structure identifier already exists before CHANGE STRUCTURE IDENTIFIER, the subroutine will replace its contents with the contents of the original structure identifier. Any existing references to the resulting structure identifier are not changed.

If the resulting structure identifier is the open structure when the subroutine is called, CHANGE STRUCTURE IDENTIFIER closes the structure, replaces the contents with the elements from original structure identifier, and reopens the structure. The element pointer will be set to point to the last element.

If the resulting structure identifier is posted when CHANGE STRUCTURE IDENTIFIER is called, it will remain posted after the subroutine is complete. The display is changed to reflect the new contents of the structure according to the display update state of the workstation to which the structure is posted.

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

SEE ALSO
CHANGE STRUCTURE IDENTIFIER AND REFERENCES (3P)
CHANGE STRUCTURE REFERENCES (3P)
NAME

CHANGE STRUCTURE IDENTIFIER AND REFERENCES – change the identifier assigned to a structure and all references to it

SYNOPSIS

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pcstir ( OLDSID, NEWSID )
INTEGER OLDSID      /* original structure identifier */
INTEGER NEWSID      /* resulting structure identifier */
```

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose

CHANGE STRUCTURE IDENTIFIER AND REFERENCES changes the identifier of a specified structure, and all references to the original identifier, to reference the new identifier. References may be both EXECUTE STRUCTURE elements and workstation postings.

C Input Parameters

`orig_struct_id`

Specifies the structure for which the identifier and references are to be changed.

`result_struct_id`

Specifies the new identifier to assign to the structure.

FORTRAN Input Parameters

`OLDSID`

Specifies the structure for which the identifier and references are to be changed.

`NEWSID`

Specifies the identifier assigned to this structure when CHANGE STRUCTURE IDENTIFIER AND REFERENCES returns.

Execution

The CHANGE STRUCTURE IDENTIFIER AND REFERENCES subroutine changes the identifier associated with `original structure identifier` to `resulting structure identifier`. This subroutine also changes all references to the original structure identifier to instead refer to the resulting structure identifier. The result of CHANGE STRUCTURE IDENTIFIER AND REFERENCES varies with the state of the original and resulting structures when the subroutine is called. The possible effects on the structures are described below.

Structure references are established by creating EXECUTE STRUCTURE elements or by posting a structure. A single call to CHANGE STRUCTURE IDENTIFIER AND REFERENCES changes both types of references to the specified structure. Details of how the subroutine changes each type of structure reference are provided below.

modified 2 April 1993
If original structure identifier and resulting structure identifier specify the same structure, the subroutine does not take any action.

**The Original Structure**
The structure identified by original structure identifier will no longer exist after this subroutine is executed, unless it was the open structure. In this case, it continues to exist as the open structure after the subroutine returns, but it will be empty and the element pointer will be set to 0.

**The Resulting Structure**
The resulting structure always exists at the end of CHANGE STRUCTURE IDENTIFIER AND REFERENCES. If resulting structure identifier does not exist when the subroutine is called, it is created and contains the elements of the original structure. If original structure identifier does not exist when the subroutine is called, the resulting structure is empty. If resulting structure identifier already exists before CHANGE STRUCTURE IDENTIFIER AND REFERENCES, the subroutine will replace its contents with the contents of original structure identifier.

If resulting structure identifier is the open structure when the subroutine is called, CHANGE STRUCTURE IDENTIFIER AND REFERENCES closes the structure, replaces the contents with the elements from original structure identifier, and reopenes the structure. The element pointer will be set to point to the last element.

**How EXECUTE STRUCTURE References Are Changed**
CHANGE STRUCTURE IDENTIFIER AND REFERENCES changes all EXECUTE STRUCTURE elements throughout the Central Structure Store that reference original structure identifier so that they reference resulting structure identifier instead. Any references to resulting structure identifier that already exist when CHANGE STRUCTURE IDENTIFIER AND REFERENCES is called will not be changed by the subroutine.

**How Posted Structures Are Changed**
If original structure identifier is posted to a workstation when CHANGE STRUCTURE IDENTIFIER AND REFERENCES is called, the subroutine unposts original structure identifier and posts resulting structure identifier with the same priority that original structure identifier had.

If resulting structure identifier is posted when CHANGE STRUCTURE IDENTIFIER AND REFERENCES is called, it will remain posted and its priority will not change. If original structure identifier is also posted, it will be unposted.

Changes in a posted structure network are processed immediately and may immediately affect the display. The actual visual effects that appear on the display surface will depend on the workstation's current display update state.
| **ERRORS** | 002 | Ignoring function, function requires state (PHOP, *, *, *) |
| **SEE ALSO** | CHANGE STRUCTURE IDENTIFIER (3P) |
| | CHANGE STRUCTURE REFERENCES (3P) |
### NAME
CHANGE STRUCTURE REFERENCES – change all references to a specified structure to instead refer to another specified structure

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pcstrf ( OLDSID, NEWSID )
INTEGER OLDSID  // original structure identifier
INTEGER NEWSID  // resulting structure identifier
```

### Required PHIGS Operating States

( PHOP, *, *, * )

### DESCRIPTION

**Purpose**

CHANGE STRUCTURE REFERENCES modifies the arrangement of the structure network by changing all references to a specified structure (the *original structure identifier*) to refer to a different structure (the *resulting structure identifier*) instead. The subroutine changes both EXECUTE STRUCTURE references and any posting to a workstation.

The effect is to substitute the resulting structure for the original structure wherever it appears in a structure network or on a workstation posting.

### C Input Parameters

- `orig_struct_id`
  
  CHANGE STRUCTURE REFERENCES changes the references to this structure.

- `result_struct_id`
  
  Specifies the structure the original structure references will point to when CHANGE STRUCTURE REFERENCES returns.

### FORTRAN Input Parameters

- `OLDSID`
  
  CHANGE STRUCTURE REFERENCES changes the references to this structure.

- `NEWSID`
  
  Specifies the structure the original structure references will point to when CHANGE STRUCTURE REFERENCES returns.

### Execution

The CHANGE STRUCTURE REFERENCES subroutine changes all references to the original structure identifier to refer instead to the resulting structure identifier. Structure references are established by creating EXECUTE STRUCTURE elements or by posting a structure. A single call to CHANGE STRUCTURE REFERENCES changes both types of references to the specified structure. The following details how the subroutine changes each type of structure reference.

---

60 modified 2 April 1993
How EXECUTE STRUCTURE References Are Changed

CHANGE STRUCTURE REFERENCES changes all EXECUTE STRUCTURE elements throughout the Central Structure Store that reference original structure identifier so that they reference resulting structure identifier instead. Any references to resulting structure identifier that already exist when CHANGE STRUCTURE REFERENCES is called will not be changed by the subroutine.

If resulting structure identifier does not exist when the subroutine is called, it will be created as an empty structure and the original structure identifier references assigned to it.

If original structure identifier and resulting structure identifier specify the same structure, the subroutine does not take any action.

How Posted Structures Are Changed

If original structure identifier is posted to a workstation when CHANGE STRUCTURE REFERENCES is called, the subroutine unposts original structure identifier and posts resulting structure identifier with the same priority that original structure identifier had.

If resulting structure identifier is posted when CHANGE STRUCTURE REFERENCES is called, it will remain posted and its priority will not change. If original structure identifier is also posted, it will be unposted.

Changes in a posted structure network are processed immediately and may immediately affect the display. The actual visual effects that appear on the display surface will depend on the workstation’s current display update state.

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

SEE ALSO

CHANGE STRUCTURE IDENTIFIER (3P)
CHANGE STRUCTURE IDENTIFIER AND REFERENCES (3P)
NAME
CLOSE ARCHIVE FILE – close named archive file

SYNOPSIS

C Syntax
void
close_ar_file ( archive_id )
int archive_id; archive identifier

FORTRAN Syntax
SUBROUTINE pclarf ( AFID )
INTEGER AFID archive file identifier

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use CLOSE ARCHIVE FILE to close an archive file.

C Input Parameter
archive_id
An integer identifying the archive file to be closed.

FORTRAN Input Parameter
AFID An integer identifying the archive file to be closed. The physical file associated
with this identifier will be closed when this function returns, even if it was open
when passed to OPEN ARCHIVE.

Execution
If the archive associated with the specified archive file identifier has changed since it was
opened, it is written to the file and the file is closed. If no changes were made, the file will
be closed without being rewritten.

The archive file identifier is disassociated with the file it referred to and removed from
the set of open archive files maintained in the PHIGS state list. The PHIGS archive state is
changed to archive closed (ARCL) if no other archive files are open.

Two archive file formats are supported: clear text format (defined by the PHIGS standard)
and a PEX binary format. The binary format is supported for users who want compact
archives over standard conformance. SunPHIGS writes clear text as the default archive file
format. The ESCAPE function can be used to control the type of archive written; see the
ESCAPE -15 reference manual page for further information. SunPHIGS reads either clear
text or binary archives, as appropriate for how they were written.

SunPHIGS reads SunPHIGS 1.X binary archive files to ensure backward compatibility with
previous SunPHIGS releases.

ERRORS
007 Ignoring function, function requires state (PHOP, *, *, AROP)
404 Ignoring function, the specified archive file is not open
412 Ignoring function, the archive file is read-only.

modified 2 April 1993
SEE ALSO

ESCAPE -15 (3P)
INQUIRE ARCHIVE FILES (3P)
INQUIRE ARCHIVE STATE VALUE (3P)
OPEN ARCHIVE FILE (3P)

modified 2 April 1993
**NAME**  
CLOSE PHIGS – close PHIGS environment

**SYNOPSIS**

**C Syntax**

```c
void
pclose_phigs()
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pclph
```

**Required PHIGS Operating States**

( PHOP, WSCL, STCL, ARCL )

**DESCRIPTION**

**Purpose**

Use the CLOSE PHIGS function to set the PHIGS system state to PHCL and to close the PHIGS environment.

**Note:** You must close any open structures, archives, or workstations by calling the appropriate function (CLOSE STRUCTURE, CLOSE ARCHIVE, or CLOSE WORKSTATION) before calling CLOSE PHIGS.

**C Input Parameters**

None.

**FORTRAN Input Parameters**

None.

**Execution**

When the CLOSE PHIGS function is called:

- The PHIGS system state variable is set to PHCL.
- The PHIGS state list becomes unavailable.
- The workstation description tables become unavailable.
- All PHIGS buffers are released.
- All PHIGS files are closed.

**ERRORS**

004 Ignoring function, function requires state (PHOP, WSCL, STCL, ARCL)

**SEE ALSO**

INQUIRE SYSTEM STATE VALUE (3P)  
OPEN PHIGS (3P)
<table>
<thead>
<tr>
<th>NAME</th>
<th>CLOSE STRUCTURE – close the currently open structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td></td>
</tr>
<tr>
<td>C Syntax</td>
<td>void</td>
</tr>
<tr>
<td></td>
<td>pclose_struct ( )</td>
</tr>
<tr>
<td>FORTRAN Syntax</td>
<td>SUBROUTINE pclst</td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Use CLOSE STRUCTURE to close the currently open structure.</td>
</tr>
<tr>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>C Input Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>FORTRAN Input Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Execution</td>
<td>CLOSE STRUCTURE ends the editing session on the currently open structure. The structure state is set to STCL.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>005 Ignoring function, function requires state (PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRE OPEN STRUCTURE (3P)</td>
</tr>
<tr>
<td></td>
<td>INQUIRE STRUCTURE STATUS (3P)</td>
</tr>
<tr>
<td></td>
<td>OPEN STRUCTURE (3P)</td>
</tr>
</tbody>
</table>

modified 2 April 1993
NAME
CLOSE WORKSTATION – close specified workstation

SYNOPSIS
C Syntax
void
pclose_ws ( ws_id )
Pint  ws_id;  workstation identifier

FORTRAN Syntax
SUBROUTINE pclwk ( WKID )
INTEGER  WKID  workstation identifier

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
The CLOSE WORKSTATION function closes the specified workstation and releases all
resources used by it.

C Input Parameter
ws_id  The identifier of the workstation to close.

FORTRAN Input
Parameter
WKID  The identifier of the workstation to close.

Execution
The CLOSE WORKSTATION function performs the following functions:

- An implicit UPDATE WORKSTATION is performed on the specified workstation,
  with the regeneration flag set to PERFORM.

- The workstation state list is released.

- The workstation identifier is deleted from the set of open workstations in the
  PHIGS state list.

- The workstation identifier is deleted from the list of workstations to which
  structures are posted in the structure state list.

- Any events in the input queue from devices on this workstation are flushed from
  the queue.

- The specific workstation description table for this workstation, created when the
  workstation was opened, becomes unavailable and the workstation type value
  associated with it becomes undefined.

- The connection to this workstation is released.

- CGM output workstations write the END METAFILE element to the CGM file.

- If no other workstations are open, the PHIGS workstation operating state is set to
  WSCL.
If an input queue overflow has been caused by a logical input device on the workstation being closed, the error state list entry (identification of one of the logical input devices that caused an input queue overflow) will become undefined when the workstation is closed.

**ERRORS**

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **256** Warning, the input queue has overflowed

**SEE ALSO**

- INQUIRE SET OF OPEN WORKSTATIONS (3P)
- INQUIRE WORKSTATION STATE VALUE (3P)
- OPEN WORKSTATION (3P)
NAME  COMPOSE MATRIX – generate the composition of two homogeneous 2D matrices

SYNOPSIS
C Syntax

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

FORTRAN Syntax

SUBROUTINE pcom ( XFRMTA, XFRMTB, ERRIND, XFRMTO )
REAL XFRMTA(3, 3)  // transformation matrix A
REAL XFRMTB(3, 3)  // transformation matrix B
INTEGER ERRIND  // OUT error indicator
REAL XFRMTO(3, 3)  // OUT composed transformation matrix

Required PHIGS
Operating States (PHOP, *, *, *)

DESCRIPTION
Purpose
Use COMPOSE MATRIX to produce a 2D (3 × 3) homogeneous matrix that is the composition (matrix multiplication product) of two specified 3 × 3 matrices.

The returned matrix may be passed to the SET LOCAL TRANSFORMATION and SET GLOBAL TRANSFORMATION functions, or to any PHIGS function accepting a parameter of type Pmatrix.

C Input Parameters
a, b  The 2D (3 × 3) matrices used to calculate the composition matrix. Matrices are specified as a Pmatrix array defined in phigs.h as follows:
typedef Pfloat Pmatrix[3][3];

C Output Parameters
error_ind  A pointer to the location to store the error number of any error detected by this function.
m  The resulting composition matrix.

FORTRAN Input Parameters
XFRMTA, XFRMTB  The 2D (3 × 3) matrices used to calculate the composition matrix.

FORTRAN Output Parameters
ERRIND  The error number of any error detected by this function.
XFRMTO  The resulting composition matrix.
**Execution**

COMPOSE MATRIX returns a $3 \times 3$ matrix that is the result of multiplying the two specified matrices $a$ and $b$:

$$result = a \times b$$

**ERRORS**

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

**SEE ALSO**

- COMPOSE MATRIX 3 (3P)
- COMPOSE TRANSFORMATION MATRIX (3P)
- BUILD TRANSFORMATION MATRIX (3P)
- SET LOCAL TRANSFORMATION (3P)
- SET GLOBAL TRANSFORMATION (3P)
**NAME**

COMPOSE MATRIX 3 – generate the composition of two homogeneous 3D matrices

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pcom3 ( XFRMTA, XFRMTB, ERRIND, XFRMTO )
REAL XFRMTA(4, 4)  /* transformation matrix A */
REAL XFRMTB(4, 4)  /* transformation matrix B */
INTEGER ERRIND  /* OUT error indicator */
REAL XFRMTO(4, 4)  /* OUT composed transformation matrix */
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use COMPOSE MATRIX 3 to produce a 3D (4 × 4) homogeneous matrix that is the composition (matrix multiplication product) of two specified 4 × 4 matrices. The returned matrix may be passed to the SET LOCAL TRANSFORMATION 3 and SET GLOBAL TRANSFORMATION 3 functions, or to any PHIGS function accepting a parameter of type Pmatrix3.

**C Input Parameters**

- `a, b` The 3D (4 × 4) matrices to use to calculate the composition matrix. Matrices are specified as a Pmatrix3 array defined in phigs.h as follows:
  ```c
typedef Pfloat Pmatrix3[4][4];
```

**C Output Parameters**

- `error_ind` A pointer to the location to store the error number of any error detected by this function.
- `m` The resulting composition matrix.

**FORTRAN Input Parameters**

- `XFRMTA, XFRMTB` The 3D (4 × 4) matrices to use to calculate the composition matrix.

**FORTRAN Output Parameters**

- `ERRIND` The error number of any error detected by this function.
- `XFRMTO` The resulting composition matrix.
**Execution**

COMPOSE MATRIX 3 returns a 4 x 4 matrix that is the result of multiplying the two specified matrices $a$ and $b$:

$$\text{result} = a \times b$$

**ERRORS**

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

**SEE ALSO**

- COMPOSE MATRIX (3P)
- COMPOSE TRANSFORMATION MATRIX 3 (3P)
- BUILD TRANSFORMATION MATRIX 3 (3P)
- SET LOCAL TRANSFORMATION 3 (3P)
- SET GLOBAL TRANSFORMATION 3 (3P)
COMPOSE TRANSFORMATION MATRIX – compose a 2D transformation matrix, which is the composition of a specified matrix and a transformation matrix defined by a fixed point, shift vector, rotation angle, and scale factors.

SYNOPSIS

C Syntax

```c
void pcompose_tran_matrix ( m, pt, shift, angle, scale, error_ind, result )
```

Fortran Syntax

```fortran
SUBROUTINE pcotm ( XFRMTI, X0, Y0, DX, DY, PHI, FX, FY, ERRIND, XFRMTO )
```

DESCRIPTION

Purpose

Use COMPOSE TRANSFORMATION MATRIX to generate a 2D homogeneous (3 × 3) matrix that composes an existing 3 × 3 matrix with a transformation specified by 2D values for scaling, rotation, and translation about a fixed point.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION or SET GLOBAL TRANSFORMATION to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameters

- **m**: The 2D (3 × 3) homogeneous transformation matrix to use in the composition. This matrix is a Pmatrix type, defined in phigs.h as follows:
  ```c
typedef Pfloat Pmatrix[3][3];
```

- **pt**: A pointer to a Ppoint structure containing the x and y coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point. Ppoint is defined in phigs.h as follows:
  ```c
typedef struct {
    Pfloat x; /* x coordinate */
  } Ppoint;
```
C Output Parameters

  error_ind
  A pointer to the location to store the error number of any error detected by this function.

  result
  The resulting 2D (3 x 3) transformation matrix.

FORTRAN Input Parameters

  XFRMTI
  The 2D (3 x 3) homogeneous transformation matrix to use in the composition.

  X0, Y0
  The x and y coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point.

  DX, DY
  The x and y coordinates defining the shift (translation) to be added to the transformation.

  PHI
  The angle of rotation, in radians, to be added to the transformation. A positive angle is a counter-clockwise rotation. A negative angle is clockwise rotation.

  FX, FY
  The x and y values defining the scale factors to be applied to the transformation.

FORTRAN Output Parameters

  ERRIND
  The error number of any error detected by this function.

  XFRMTO
  The resulting 2D (3 x 3) transformation matrix.

Execution

COMPOSE TRANSFORMATION MATRIX returns the matrix that performs the transformation resulting from the composition of the input matrix and the matrix specified by the input parameters shift, angle, and scale. Rotation and scaling are calculated relative to the fixed point.

The composition is performed as:

  result = a x b
Where $b$ is the input matrix and $a$ is the matrix built from the other input parameters. The order of operations used to build the transformation matrix $b$ is:

- scale
- rotate
- shift (translate)

**ERRORS**

002 Ignoring function, function requires state ($\text{PHOP}$, $\ast$, $\ast$, $\ast$)

**SEE ALSO**

- COMPOSE TRANSFORMATION MATRIX 3 ($3P$)
- COMPOSE MATRIX ($3P$)
- BUILD TRANSFORMATION MATRIX ($3P$)
- SET LOCAL TRANSFORMATION ($3P$)
- SET GLOBAL TRANSFORMATION ($3P$)
COMPOSE TRANSFORMATION MATRIX 3 – compose a 3D transformation matrix, which is the composition of a specified matrix and a transformation matrix defined by a fixed point, shift vector, rotation angle, and scale factors.

**SYNOPSIS**

**C Syntax**
```c
void pcompose_tran_matrix3 ( m, pt, shift, x_ang, y_ang, z_ang, scale, error_ind, result )
```

- **m**
  - Pmatrix3 type, defined in phigs.h as follows:
  ```c
typedef float Pmatrix3[4][4];
```
- **pt**
  - A pointer to a Ppoint3 structure containing the x, y, and z coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed.

**FORTRAN Syntax**
```fortran
SUBROUTINE pcomt3 ( XFRMTI, X0, Y0, Z0, DX, DY, DZ, PHIX, PHIY, PHIZ, FX, FY, FZ, ERRIND, XFRMTO )
```

- **XFRMTI(4, 4)**
  - transformation matrix
- **X0, Y0, Z0**
  - fixed point
- **DX, DY, DZ**
  - shift vector
- **PHIX, PHIY, PHIZ**
  - rotation angles (radians)
- **FX, FY, FZ**
  - scale factor vector
- **ERRIND**
  - OUT error indicator
- **XFRMTO(4, 4)**
  - OUT transformation matrix

**DESCRIPTION**

**Purpose**
Use COMPOSE TRANSFORMATION MATRIX 3 to generate a 3D (4 x 4) matrix that composes an existing 4 x 4 matrix with a transformation specified by 3D values for scaling, rotation, and translation about a fixed point.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation applied to output primitives during traversal.

**C Input Parameters**
- **m**
  - The 3D (4 x 4) homogeneous transformation matrix to use in the composition.
  - This matrix is a Pmatrix3 type, defined in phigs.h as follows:
  ```c
typedef float Pmatrix3[4][4];
```
- **pt**
  - A pointer to a Ppoint3 structure containing the x, y, and z coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed.
relative to this fixed point. Ppoint3 is defined in phigs.h as follows:

typedef struct {
    float  x;  // x coordinate
    float  y;  // y coordinate
    float  z;  // z coordinate
} Ppoint3;

shift  A pointer to a Pvec3 structure containing x, y, and z coordinates defining the shift (translation) to be added to the transformation. Pvec3 is defined in phigs.h as follows:

typedef struct {
    float  delta_x;  // x magnitude
    float  delta_y;  // y magnitude
    float  delta_z;  // z magnitude
} Pvec3;

x_ang, y_ang, z_ang  
The angles of rotation about the indicated axes, in radians, to be added to the transformation. A positive angle is a counterclockwise rotation. A negative angle is clockwise rotation.

cscale  A pointer to a Pvec3 structure containing x, y, and z values defining the scale factors to be applied to the transformation.

C Output Parameters

error_ind  A pointer to the location to store the error number of any error detected by this function.

result  The resulting 3D (4 x 4) transformation matrix.

FORTRAN Input Parameters

XFRMTI  The 3D (4 x 4) homogeneous transformation matrix to use in the composition.

X0, Y0, Z0  
The x, y, and z coordinates of a fixed point in Modelling Coordinates (MC). Scaling and rotation are performed relative to this fixed point.

DX, DY, DZ  
The x, y, and z coordinates defining the shift (translation) to be added to the transformation.

PHIX, PHIY, PHIZ  
The angles of rotation about the indicated axes, in radians, to be added to the transformation. A positive angle is a counterclockwise rotation. A negative angle is clockwise rotation.

FX, FY, FZ  
The x, y, and z values defining the scale factors to be applied to the transformation.
FORTRAN Output

Parameters

ERRIND
The error number of any error detected by this function.

XFRMTO
The resulting 3D (4 × 4) transformation matrix.

Execution

COMPOSE TRANSFORMATION MATRIX 3 returns the matrix that performs the transformation resulting from the composition of the input matrix and the matrix specified by the input parameters shift, angle, and scale. Rotation and scaling are calculated relative to the fixed point.

The composition is performed as:

\[ \text{result} = a \times b \]

Where \( b \) is the input matrix and \( a \) is the matrix built from the other input parameters.

The order of operations used to build the transformation matrix \( b \) is:

- scale
- rotate
- shift (translate)

ERRORS

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

SEE ALSO

COMPOSE TRANSFORMATION MATRIX (3P)
BUILD TRANSFORMATION MATRIX 3 (3P)
COMPOSE MATRIX 3 (3P)
SET LOCAL TRANSFORMATION 3 (3P)
SET GLOBAL TRANSFORMATION 3 (3P)
NAME
COPY ALL ELEMENTS FROM STRUCTURE – copy elements of specified structure into open structure

SYNOPSIS
C Syntax
void
pcopy_all_elems_struct ( struct_id )

int struct_id;  // structure identifier

FORTRAN Syntax
SUBROUTINE pcelst ( STRID )

INTEGER STRID  // source structure identifier

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
COPY ALL ELEMENTS FROM STRUCTURE copies the structure elements of the source structure identifier into the currently open structure. The elements are copied into the open structure after the element pointed to by the element pointer.

C Input Parameter
struct_id
Identifies the structure from which to copy the elements.

FORTRAN Input Parameter
STRID  // Identifies the structure from which to copy the elements.

Execution
The COPY ALL ELEMENTS FROM STRUCTURE subroutine copies all the structure elements in the source structure identifier and inserts them into the currently open structure after the element pointed to by the element pointer. The element pointer is then updated to point to the last element inserted. The current edit mode has no effect on this subroutine.

If source structure identifier is the open structure, then its contents are copied into itself after the element pointed to by the element pointer.

If source structure identifier refers to an empty structure or to a structure that does not exist, then the subroutine does not perform any action and does not generate an error.

ERRORS
005  // Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
CHANGE STRUCTURE IDENTIFIER (3P)
COPY ELEMENTS BETWEEN LABELS (3PP)
COPY ELEMENT RANGE (3PP)
EXECUTE STRUCTURE (3P)
**NAME**
CREATE STORE – creates a Store, and returns a handle to it

**SYNOPSIS**

C Syntax
void
pcreate_store ( err, store )
Pint *err; /* OUT error */
Pstore *store; /* OUT handle to store object */

**Required PHIGS Operating States**
(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**
CREATE STORE creates a Store object and returns the handle to it. Store objects are used to hold the return values of complex inquiry functions.

**C Output Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>err</td>
<td>The status of the creation of the new Store.</td>
</tr>
<tr>
<td>store</td>
<td>The handle to the newly created Store.</td>
</tr>
</tbody>
</table>

**Execution**

CREATE STORE creates a new Store and returns a handle to it via the `store` parameter. The parameter `err` returns the error number of any error detected. If a zero is returned the creation was successful.

A Store is an opaque pointer that is passed as a parameter to a function returning complex data. A Store contains the memory to hold the results of a complex inquiry. A Store may be passed to or used by more than one inquiry function, but at any one time a Store only holds the result of the last inquiry function that used it as an argument. A Store continues to hold the information from the last inquiry function until it is freed by DELETE STORE, or until the Store is used as an argument to a subsequent inquiry function, in which case the information is overwritten.

**ERRORS**

900 Storage overflow has occurred in PHIGS

**SEE ALSO**

DELETE STORE (3P)

*modified 2 April 1993*
### NAME
DELETE ALL STRUCTURES – remove all existing structures from the Central Structure Store (CSS)

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pdas
```

### Required PHIGS Operating States

(PHOP, *, *, *)

### DESCRIPTION

**Purpose**

Use DELETE ALL STRUCTURES to remove all the existing structures from the CSS.

**C Input Parameters**

None.

**FORTRAN Input Parameters**

None.

**Execution**

The **DELETE ALL STRUCTURES** subroutine removes all the existing structures from the CSS. To do this, **DELETE ALL STRUCTURES** executes the **DELETE STRUCTURE** function for each existing structure. All structure identifiers and structure elements are deleted, with the exception of an open structure.

If a structure is open, **DELETE ALL STRUCTURES** retains the structure with its current structure identifier, but removes all the structure elements from it. The result is to replace the open structure with an empty structure.

Any structure that is posted to a workstation is unposted.

### ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
</tbody>
</table>

### SEE ALSO

- **DELETE STRUCTURE (3P)**
- **DELETE STRUCTURE NETWORK (3P)**
- **DELETE ALL STRUCTURES FROM ARCHIVE (3P)**
DELETE ALL STRUCTURES FROM ARCHIVE – delete all structures from an archive file

**NAME**

DELETE ALL STRUCTURES FROM ARCHIVE

**SYNOPSIS**

**C Syntax**

```c
void
pdel_all_structs_ar ( archive_id )
Pint   archive_id;   archive identifier
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pdasar ( AFID )
INTEGER   AFID   archive file identifier
```

**Required PHIGS Operating States**

(PHOP, *, *, AROP)

**DESCRIPTION**

**Purpose**

Use DELETE ALL STRUCTURES FROM ARCHIVE to delete all structures in the specified open archive file.

**C Input Parameter**

`archive_id`

The archive identifier specifying the open archive file to delete from.

**FORTRAN Input Parameter**

`AFID`

The archive identifier specifying the open archive file to delete from.

**Execution**

All structures in the specified archive file are deleted and the archive file is left in the same state as if it had just been created.

**ERRORS**

007 Ignoring function, function requires state (PHOP, *, *, AROP)

404 Ignoring function, the specified archive file is not open

**SEE ALSO**

DELETE ALL STRUCTURES (3P)
DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
DELETE STRUCTURES FROM ARCHIVE (3P)
<table>
<thead>
<tr>
<th>NAME</th>
<th>DELETE ELEMENT – delete structure element</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td></td>
</tr>
<tr>
<td>C Syntax</td>
<td>void pdel_elem ( )</td>
</tr>
<tr>
<td>FORTRAN Syntax</td>
<td>SUBROUTINE pdel</td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td>DELETE ELEMENT deletes the structure element currently pointed to by the element pointer in the open structure.</td>
</tr>
<tr>
<td>C Input Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>FORTRAN Input Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Execution</td>
<td>The DELETE ELEMENT subroutine removes the structure element currently pointed to by the element pointer in the open structure and renumbers the remaining elements in the structure to maintain a consecutive, ascending order in the structure. When the element is deleted, the element pointer is set to point to the element preceding the deleted element in the structure. If the element pointer is 0 (pointing to the position in the structure before any elements) when DELETE ELEMENT is called, the subroutine will not delete any element and will not adjust the element pointer.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>005 Ignoring function, function requires state (PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>SET EDIT MODE (3P)</td>
</tr>
<tr>
<td></td>
<td>EMPTY STRUCTURE (3P)</td>
</tr>
<tr>
<td></td>
<td>DELETE ELEMENT RANGE (3P)</td>
</tr>
<tr>
<td></td>
<td>DELETE ELEMENTS BETWEEN LABELS (3P)</td>
</tr>
</tbody>
</table>

modified 2 April 1993
NAME
DELETE ELEMENT RANGE – delete a block of elements in a structure

SYNOPSIS
C Syntax

void
del_elem_range ( ep1_value, ep2_value )

Pint ep1_value; element pointer 1 value
Pint ep2_value; element pointer 2 value

FORTRAN Syntax
SUBROUTINE pdelra ( EP1, EP2 )

INTEGER EP1, EP2 element pointer range

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
Use DELETE ELEMENT RANGE to delete all structure elements in the open structure between, and including, two specified element numbers.

C Input Parameters
ep1_value
Specifies the beginning of the element range to be deleted.

ep2_value
Specifies the end of the element range to be deleted.

FORTRAN Input Parameters
EP1
Specifies the beginning of the element range to be deleted.

EP2
Specifies the end of the element range to be deleted.

Execution
The DELETE ELEMENT RANGE subroutine removes structure elements from the open structure between and including the lower and the higher of the element positions specified by element pointer 1 value and element pointer 2 value. The remaining elements in the structure are renumbered and the element pointer is updated to point to the element preceding the deleted elements.

If an element position is less than 1, the range of elements to be deleted will start from element position 0. If an element position is greater than the number of elements in the open structure, the range of elements to be deleted will end at the last element of the open structure.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
DELETE ELEMENT (3P)
DELETE ELEMENTS BETWEEN LABELS (3P)
EMPTY STRUCTURE (3P)

modified 2 April 1993
### NAME
DELETE ELEMENTS BETWEEN LABELS – delete all elements in the open structure between specified labels

### SYNOPSIS
#### C Syntax
```c
void pdel_elems_labels ( label1_id, label2_id )
Pint label1_id;  // label 1 identifier
Pint label2_id;  // label 2 identifier
```

#### FORTRAN Syntax
```fortran
SUBROUTINE pdellb ( LABEL1, LABEL2 )
INTEGER LABEL1, LABEL2  // label range
```

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION
#### Purpose
Use DELETE ELEMENTS BETWEEN LABELS to delete structure elements in the open structure between two specified labels. The elements containing the labels are not deleted.

#### C Input Parameters
- `label1_id`
  The subroutine begins deleting structures with the structure element immediately following this label.
- `label2_id`
  The subroutine ends deleting structures with the structure element immediately preceding this label.

#### FORTRAN Input Parameters
- `LABEL1`
  The subroutine begins deleting structures with the structure element immediately following this label.
- `LABEL2`
  The subroutine ends deleting structures with the structure element immediately preceding this label.

#### Execution
The DELETE ELEMENTS BETWEEN LABELS subroutine removes all the elements in the open structure between the two specified labels. The remaining elements are renumbered and the element pointer is updated to point to label 1 identifier.

The subroutine searches for the specified labels only from the current position of the element pointer in the structure to the end of the structure. The next occurrence of an element containing label 1 identifier is first found; then, starting from the element containing label 1 identifier, the next occurrence of an element containing label 2 identifier is found.
If either of the label identifiers cannot be found between the current position of the element pointer and the end of the structure, no elements are deleted and an error is generated.

**ERRORS**

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>Ignoring function, function requires state (PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>206</td>
<td>Ignoring function, one or both of the labels does not exist in the open structure between the element pointer and the end of the structure</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- **DELETE ELEMENT (3P)**
- **DELETE ELEMENT RANGE (3P)**
- **EMPTY STRUCTURE (3P)**
NAME
DELETE STORE – destroy a Store object

SYNOPSIS
C Syntax
void
pdel_store ( store )

Pstore store; handle to store object

Required PHIGS
Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
DELETE STORE destroys the specified Store object and frees all memory associated with it.

C Input Parameter
store The Store to be deleted.

Execution
DELETE STORE frees all memory associated with a Store. A Store is an opaque pointer that is passed as a parameter to a function returning complex data. A Store contains the memory to hold the results of a complex inquiry. A Store may be passed to or used by more than one inquiry function, but at any one time a Store only holds the result of the last inquiry function that used it as an argument. A Store continues to hold the information from the last inquiry function until it is freed by DELETE STORE, or until the Store is used as an argument to a subsequent inquiry function, in which case the information is overwritten.

ERRORS
None.

SEE ALSO
CREATE STORE (3P)
### NAME
DELETE STRUCTURE – remove specified structure

### SYNOPSIS

**C Syntax**
```c
void pdel_struct ( struct_id )
Pint struct_id;  structure identifier
```

**FORTRAN Syntax**
```fortran
SUBROUTINE pdst ( STRID )
INTEGER STRID  structure identifier
```

**Required PHIGS Operating States**
(PHOP, *, *, *)

### DESCRIPTION

**Purpose**
DELETE STRUCTURE removes the specified structure from the Central Structure Store (CSS), along with all references to the structure and all postings of the structure to workstations.

- If the specified structure is closed, the subroutine removes the structure elements, structure identifier, and any references to the structure.
- If the structure is the currently open structure, the structure identifier is retained and remains open, but the structure contents and any references to the structure are deleted.

**C Input Parameter**
- `struct_id`
  Specifies the structure to be deleted.

**FORTRAN Input Parameter**
- `STRID`
  Specifies the structure to be deleted.

**Execution**
The effect of DELETE STRUCTURE when the specified structure is closed at the time the subroutine is called is different from the effect when structure identifier is the open structure. The two situations are described below.

### Deleting a Closed Structure
If structure identifier is not the currently open structure, the DELETE STRUCTURE subroutine removes the specified structure from the PHIGS CSS. The subroutine deletes the structure identifier, structure contents, and all references to structure identifier contained in other structures.

- If the currently open structure contains an element referencing structure identifier, the element in the open structure is deleted and the remaining elements renumbered. If the element pointer is pointing to the deleted element, the pointer is set to the preceding element. If the element pointer is pointing to an element following the deleted element, the element pointer is updated such that it still refers to the same element.

modified 2 April 1993
Deleting the Open Structure

If structure identifier is the currently open structure, the result of DELETE STRUCTURE is to replace the open structure with an empty, unreferenced structure. The effect is the same as calling the following subroutines in sequence:

- CLOSE STRUCTURE
- DELETE STRUCTURE
- OPEN STRUCTURE

ERRORS

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

SEE ALSO

DELETE ALL STRUCTURES (3P)
DELETE STRUCTURE NETWORK (3P)
DELETE STRUCTURES FROM ARCHIVE (3P)
EMPTY STRUCTURE (3P)
NAME
DELETE STRUCTURE NETWORK – delete network of structures from central structure store

SYNOPSIS
C Syntax
void
pdel_struct_net ( struct_id, ref_flag )

Pint struct_id; // structure identifier
Pref_flag ref_flag; // reference handling flag

FORTRAN Syntax
SUBROUTINE pdsn ( STRID, REFHNF )

INTEGER STRID // structure identifier
INTEGER REFHNF // reference handling mode (PDELE, PKEEP)

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use DELETE STRUCTURE NETWORK to remove from the Central Structure Store (CSS) a specified structure and all structures referenced directly or indirectly by that structure.

A structure is referenced from within another structure by an EXECUTE STRUCTURE element created by the EXECUTE STRUCTURE subroutine. A structure network is a hierarchical arrangement of structures in which the original structure references subordinate structures which, in turn, reference other structures, and so on.

More than one EXECUTE STRUCTURE element can refer to the same structure, so that a structure referenced in the network to be deleted can also be referenced by structures in other networks. DELETE STRUCTURE NETWORK includes a reference handling flag parameter which controls whether to delete or keep these structures.

C Input Parameters

struct_id
Specifies the structure which is the beginning of the structure network to be deleted.

ref_flag
Controls how structures in the selected network that are also referenced by other structures outside the network are to be handled. This is an enumerated type defined in phigs.h as follows:

PFLAG_DEL // Delete all structures
PFLAG_KEEP // Keep structures referenced outside the network

The use of these values is described in the Execution section below.

FORTRAN Input Parameters

STRID
Specifies the structure which is the beginning of the structure network to be deleted.

REFHNF
Controls how structures in the selected network that are also referenced by other structures outside the network are to be handled. Values for this parameter are

modified 2 April 1993
defined in phigs77.h as follows:

PDELETE Delete
PKEEP Keep

The use of these values is described in the Execution section below.

**Execution**

The DELETE STRUCTURE NETWORK subroutine deletes the structure specified by structure identifier and the structure network beginning with the specified structure. The structure network is the chain of structures referenced by EXECUTE STRUCTURE elements in the structure identifier or in descendent structures.

If the structure identifier does not exist, the subroutine takes no action and does not return an error.

The reference handling flag parameter controls how structures that are referenced both by this structure network and by another structure outside the network are handled.

- If reference handling flag is DELETE, DELETE STRUCTURE NETWORK removes all structures referenced in the structure network beginning with the structure identifier even if those structures are also referenced by another structure outside this network.
- If reference handling flag is KEEP, DELETE STRUCTURE NETWORK will not remove structures referenced in this network which are also referenced by another structure outside this network. However, the structure specified by the structure identifier is always deleted.

All structures that are actually deleted are deleted as if DELETE STRUCTURE were called for each individual structure; that is, the structure identifier, its contents, and all references to it are removed.

**ERRORS**

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

**SEE ALSO**

DELETE ALL STRUCTURES (3P)
DELETE STRUCTURE (3P)
DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)

modified 2 April 1993
NAME
DELETE STRUCTURE NETWORKS FROM ARCHIVE – delete specified structure networks from an archive file

SYNOPSIS
C Syntax
void pdel_struct_nets_ar ( archive_id, struct_ids )
Pint archive_id; /* archive identifier */
*struct_ids; /* list of structure identifiers */

FORTRAN Syntax
SUBROUTINE pdsnar ( AFID, N, LSTRID )
INTEGER AFID; /* archive file identifier */
INTEGER N; /* number of structure identifiers in the list */
INTEGER LSTRID(N); /* list of structure identifiers */

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use DELETE STRUCTURE NETWORKS FROM ARCHIVE to delete a list of structure networks from the specified open archive file.

C Input Parameters
archive_id
The archive identifier specifying the open archive file to delete from.

struct_ids
A pointer to a Pint_list structure containing the list of the root structure identifiers of the archived structure networks to be deleted. The Pint_list structure is defined in phigs.h as follows:

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

The num_ints component specifies the number of structure identifiers in the list. The ints component is a pointer to a list, num_ints long, of the structure identifiers.

FORTRAN Input Parameters
AFID The archive identifier specifying the open archive file to delete from.
N The number of archived structure networks to be deleted.
LSTRID An array of integers containing the root structure identifiers of the networks to be deleted.

Execution
The structures belonging to the specified networks are deleted from the archive file. No attempt is made to insure that the deleted structures are not referenced from other structures in the archive file.

modified 2 April 1993
ERRORS
007 Ignoring function, function requires state (PHOP, *, *, AROP)
404 Ignoring function, the specified archive file is not open
407 Warning, some of the specified structures do not exist on the archive file

SEE ALSO
DELETE ALL STRUCTURES FROM ARCHIVE (3P)
DELETE STRUCTURES FROM ARCHIVE (3P)
DELETE STRUCTURE NETWORK (3P)
**NAME**
DELETE STRUCTURES FROM ARCHIVE – delete specified structures from an archive file

**SYNOPSIS**
C Syntax

`void pdel_structs_ar ( archive_id, struct_ids )`

<table>
<thead>
<tr>
<th>Pint</th>
<th>archive_id;</th>
<th>archive identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pint_list</td>
<td>*struct_ids;</td>
<td>list of structure identifiers</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

`SUBROUTINE pdstar ( AFID, N, LSTRID )`

<table>
<thead>
<tr>
<th>INTEGER AFID</th>
<th>archive file identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER N</td>
<td>number of structure identifiers in the list</td>
</tr>
<tr>
<td>INTEGER LSTRID(N)</td>
<td>list of structure identifiers</td>
</tr>
</tbody>
</table>

**Required PHIGS Operating States**

( PHOP, *, *, AROP )

**DESCRIPTION**

**Purpose**
Use DELETE STRUCTURES FROM ARCHIVE to delete a list of structures from the specified open archive file.

**C Input Parameters**

- `archive_id`
The archive identifier specifying the open archive file to delete from.

- `struct_ids`
A pointer to a Pint_list structure containing the list of structure identifiers to be deleted. The Pint_list structure is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of structure identifiers in the list. The `ints` component is a pointer to a list, `num_ints` long, of the structure identifiers.

**FORTRAN Input Parameters**

- `AFID` The archive identifier specifying the open archive file to delete from.
- `N` The number of structure identifiers to be deleted.
- `LSTRID` An array of integers containing the structure identifiers to be deleted.

**Execution**
The specified structures are deleted from the archive file. No attempt is made to insure that the deleted structures are not referenced from other structures in the archive file.

modified 2 April 1993
| ERRORS | 007 | Ignoring function, function requires state (PHOP, *, *, AROP) |
|        | 404 | Ignoring function, the specified archive file is not open   |
|        | 407 | Warning, some of the specified structures do not exist on the archive file |

**SEE ALSO**

- DELETE ALL STRUCTURES FROM ARCHIVE (3P)
- DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
- DELETE STRUCTURE (3P)
NAME
ELEMENT SEARCH – search for next matching element in specified structure

SYNOPSIS
C Syntax

```
void pelem_search ( struct_id, start_el, dir, incl, excl, error_ind, status, found_el )
```

```
Pint struct_id;       /* structure identifier */
Pint start_el;       /* starting element pointer */
Psearch_dir dir;     /* search direction */
Pelem_type_list *incl; /* element incl. list */
Pelem_type_list *excl; /* element excl. list */
Pint *error_ind;     /* OUT error indicator */
Psearch_status *status; /* OUT search status */
Pint *found_el;      /* OUT found element pointer */
```

FORTRAN Syntax

```
SUBROUTINE pels ( STRID, STRTEP, SRCDIR, EISN, EIS, EESN, EES, ERRIND, STATUS, FNDEP )
```

```
INTEGER STRID       /* structure identifier */
INTEGER STRTEP      /* start element pointer */
INTEGER SRCDIR      /* search direction (PBWD, PFWD) */
INTEGER EISN        /* number of elements in element inclusion set */
INTEGER EIS(EISN)   /* element inclusion set */
INTEGER EESN        /* number of elements in element exclusion set */
INTEGER EES(EESN)   /* element exclusion set */
INTEGER ERRIND      /* OUT error indicator */
INTEGER STATUS      /* OUT status indicator (PFAIL, PSUCC) */
INTEGER FNDEP       /* OUT found element pointer */
```

Required PHIGS Operating States

```
(PHOP, *, *, *)
```

DESCRIPTION

Purpose

ELEMENT SEARCH searches the specified structure for the next element matching the search criteria. The search starts at the specified element (or the limit of the structure) and proceeds in the direction specified. The search terminates if an element is found, or if the opposite limit of the structure is reached.

The search criteria is based on element types. An element satisfies the criteria if its element type is in the inclusion set, and not in the exclusion set.

C Input Parameters

```
struct_id
identifier of the structure to search.
```

```
start_el
position in the specified structure of the first element tested against the search criteria.
```

```
dir
search direction. Psearch_dir is an enumerated type defined in phigs.h to have
```

modified 2 April 1993
the following values:

- PDIR_BACKWARD
- PDIR_FORWARD

incl  A pointer to a Pelem_type_list structure containing the list of element types in the inclusion set.

The Pelem_type_list structure and its sub-types are defined in phigs.h as:

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

The Pelem_type enumeration lists all the supported element types. It is enumerated in phigs.h as follows:

typedef enum {
    PELEM_ALL,
    PELEM_NIL,
    PELEM_POLYLINE3,
    PELEM_POLYLINE,
    PELEM_POLYMARKER3,
    PELEM_POLYMARKER,
    PELEM_TEXT3,
    PELEM_TEXT,
    PELEM_ANNO_TEXT_REL3,
    PELEM_ANNO_TEXT_REL,
    PELEM_FILL_AREA3,
    PELEM_FILL_AREA,
    PELEM_FILL_AREA_SET3,
    PELEM_FILL_AREA_SET,
    PELEM_CELL_ARRAY3,
    PELEM_CELL_ARRAY,
    PELEM_GDP3,
    PELEM_GDP,
    PELEM_LINE_IND,
    PELEM_MARKER_IND,
    PELEM_TEXT_IND,
    PELEM_INT_IND,
    PELEM_EDGE_IND,
    PELEM_LINETYPE,
    PELEM_LINEWIDTH,
    PELEM_LINE_COLR_IND,
    PELEM_MARKER_TYPE,
    PELEM_MARKER_SIZE,
    PELEM_MARKER_COLR_IND,

modified 2 April 1993
```c
PELEM_TEXT_FONT,
PELEM_TEXT_PREC,
PELEM_CHAR_EXPAN,
PELEM_CHAR_SPACE,
PELEM_TEXT_COLR_IND,
PELEM_CHAR_HT,
PELEM_CHAR_UP_VEC,
PELEM_TEXTUREPATH,
PELEM_TEXT_ALIGN,
PELEM_ANNO_CHAR_HT,
PELEM_ANNO_CHAR_UP_VEC,
PELEM_ANNO_PATH,
PELEM_ANNO_ALIGN,
PELEM_ANNO_STYLE,
PELEM_INT_STYLE,
PELEM_INT_STYLE_IND,
PELEM_INT_COLR_IND,
PELEM_EDGETYPE,
PELEM_EDGETEXT,
PELEM_EDGETEXTIND,
PELEM_EDGETEXTCOLR_IND,
PELEM_EDGE_FLAGS,
PELEM_EDGE_WIDTH,
PELEM_EDGE_COLR_IND,
PELEM_ADD_NAMES_SET,
PELEM_REMOVE_NAMES_SET,
PELEM_INDIV_ASF,
PELEM_HLHSR_ID,
PELEM_LOCAL_MODEL_TRAN3,
PELEM_LOCAL_MODEL_TRAN,
PELEM_GLOBAL_MODEL_TRAN3,
PELEM_GLOBAL_MODEL_TRAN,
PELEM_MODEL_CLIP_VOL3,
PELEM_MODEL_CLIP_VOL,
PELEM_MODEL_CLIP_IND,
PELEM_RESTORE_MODEL_CLIP_VOL,
PELEM_VIEW_IND,
PELEM_EXEC_STRUCT,
PELEM_LABEL,
PELEM_APPLDATA,
PELEM_GSE,
PELEM_PICK_ID,
PELEM_ALL,
PELEM_POLYLINE_SET3_DATA,
```
PELEM_FILL_AREA_SET3_DATA,
PELEM_TRI_STRIP3_DATA,
PELEM_QUAD_MESH3_DATA,
PELEM_SET_OF_FILL_AREA_SET3_DATA,
PELEM_NUNI_BSP_CURVE,
PELEM_NUNI_BSP_SURF,
PELEM_CELL_ARRAY3_PLUS,
PELEM_TEXT_COLR,
PELEM_MARKER_COLR,
PELEM_EDGE_COLR,
PELEM_LINE_COLR,
PELEM_CURVE_APPROX_CRIT,
PELEM_LINE_SHAD_METH,
PELEM_INT_COLR,
PELEM_BACK_INT_COLR,
PELEM_BACK_INT_STYLE,
PELEM_BACK_INT_STYLE_IND,
PELEM_REFL_PROPS,
PELEM_BACK_REFL_PROPS,
PELEM_INT_SHAD_METH,
PELEM_BACK_INT_SHAD_METH,
PELEM_INT_REFL_EQN,
PELEM_BACK_INT_REFL_EQN,
PELEM_SURF_APPROX_CRIT,
PELEM_FACE_DISTING_MODE,
PELEM_FACE_CULL_MODE,
PELEM_LIGHT_SRC_STATE,
PELEM_DCUE_IND,
PELEM_COLR_MAPPING_IND,
PELEM_RENDERING_COLR_MODEL,
PELEM_NUM_EL_TYPES

} Pelem_type;

excl A pointer to a Pelem_type_list structure containing the list of element types in the exclusion set. Pelem_type_list is defined above. Excluding all element types will result in failure to find matching elements.

C Output Parameters

error_ind A pointer to the location to store the error number of any error detected by this function.

status A pointer to the location to store the search status. Psearch_status is an enumerated type defined in phigs.h to have the following values:

PSEARCH_STATUS_FAILURE
PSEARCH_STATUS_SUCCESS

modified 2 April 1993
found_el
A pointer to the location to store the position of the element found. If status is PSEARCH_STATUS_FAILURE, found_el is undefined.

FORTRAN Input Parameters

STRID  Identifier of the structure to search.

STRTEP  Sequence number in the specified structure of the first element tested against the search criteria.

SRCDIR  Search direction. Valid values defined in phigs77.h are PBWD (backward) and PFWD (forward).

EISN  Number of element types in the inclusion set.

EIS(EISN)
Array of element types in the inclusion set. The special value PEALL indicates that all element types are to be included (including PENIL), so an exhaustive list of all element types need not be generated. The exclusion list could then be used to exclude certain element types. The mapping from the FORTRAN constants in phigs77.h to the actual element type names is given below for supported elements:

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEALL</td>
<td>Include all elements</td>
</tr>
<tr>
<td>PENIL</td>
<td>NIL</td>
</tr>
<tr>
<td>PEPL3</td>
<td>POLYLINE 3</td>
</tr>
<tr>
<td>PEPL</td>
<td>POLYLINE</td>
</tr>
<tr>
<td>PEPM3</td>
<td>POLYMARKER 3</td>
</tr>
<tr>
<td>PEPM</td>
<td>POLYMARKER</td>
</tr>
<tr>
<td>PETX3</td>
<td>TEXT 3</td>
</tr>
<tr>
<td>PETX</td>
<td>TEXT</td>
</tr>
<tr>
<td>PEATR3</td>
<td>ANNOTATION TEXT RELATIVE 3</td>
</tr>
<tr>
<td>PEATR</td>
<td>ANNOTATION TEXT RELATIVE</td>
</tr>
<tr>
<td>PEFA3</td>
<td>FILL AREA 3</td>
</tr>
<tr>
<td>PEFA</td>
<td>FILL AREA</td>
</tr>
<tr>
<td>PEFAS3</td>
<td>FILL AREA SET 3</td>
</tr>
<tr>
<td>PEFAS</td>
<td>FILL AREA SET</td>
</tr>
<tr>
<td>PECA3</td>
<td>CELL ARRAY 3</td>
</tr>
<tr>
<td>PECA</td>
<td>CELL ARRAY</td>
</tr>
<tr>
<td>PEGDP3</td>
<td>GDP 3</td>
</tr>
<tr>
<td>PEGDP</td>
<td>GDP</td>
</tr>
<tr>
<td>PEPLI</td>
<td>POLYLINE INDEX</td>
</tr>
<tr>
<td>PEPMI</td>
<td>POLYMARKER INDEX</td>
</tr>
<tr>
<td>PETXI</td>
<td>TEXT INDEX</td>
</tr>
<tr>
<td>PEII</td>
<td>INTERIOR INDEX</td>
</tr>
<tr>
<td>PEEDI</td>
<td>EDGE INDEX</td>
</tr>
<tr>
<td>PELN</td>
<td>LINETYPE</td>
</tr>
</tbody>
</table>

modified 2 April 1993
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PELWSC</td>
<td>LINEWIDTH SCALE FACTOR</td>
</tr>
<tr>
<td>PEPLCI</td>
<td>POLYLINE COLOUR INDEX</td>
</tr>
<tr>
<td>PEMK</td>
<td>MARKER TYPE</td>
</tr>
<tr>
<td>PEMKSC</td>
<td>MARKER SIZE SCALE FACTOR</td>
</tr>
<tr>
<td>PEMPCI</td>
<td>MARKER COLOUR INDEX</td>
</tr>
<tr>
<td>PETXFN</td>
<td>TEXT FONT</td>
</tr>
<tr>
<td>PETXPR</td>
<td>TEXT PRECISION</td>
</tr>
<tr>
<td>PECHXP</td>
<td>CHARACTER EXPANSION FACTOR</td>
</tr>
<tr>
<td>PECHSP</td>
<td>CHARACTER SPACING</td>
</tr>
<tr>
<td>PETXC1</td>
<td>TEXT COLOUR INDEX</td>
</tr>
<tr>
<td>PECHH</td>
<td>CHARACTER HEIGHT</td>
</tr>
<tr>
<td>PECHUP</td>
<td>CHARACTER UP VECTOR</td>
</tr>
<tr>
<td>PETXP</td>
<td>TEXT PATH</td>
</tr>
<tr>
<td>PETXAL</td>
<td>TEXT ALIGNMENT</td>
</tr>
<tr>
<td>PEATCH</td>
<td>ANNOTATION TEXT CHARACTER HEIGHT</td>
</tr>
<tr>
<td>PEATCU</td>
<td>ANNOTATION TEXT CHARACTER UP VECTOR</td>
</tr>
<tr>
<td>PEATP</td>
<td>ANNOTATION TEXT PATH</td>
</tr>
<tr>
<td>PEATAL</td>
<td>ANNOTATION TEXT ALIGNMENT</td>
</tr>
<tr>
<td>PEANST</td>
<td>ANNOTATION STYLE</td>
</tr>
<tr>
<td>PEIS</td>
<td>INTERIOR STYLE</td>
</tr>
<tr>
<td>PEISI</td>
<td>INTERIOR STYLE INDEX</td>
</tr>
<tr>
<td>PEICI</td>
<td>INTERIOR COLOUR INDEX</td>
</tr>
<tr>
<td>PEEDFG</td>
<td>EDGE FLAG</td>
</tr>
<tr>
<td>PEEDT</td>
<td>EDGETYPE</td>
</tr>
<tr>
<td>PEIEWSC</td>
<td>EDGWIDTH SCALE FACTOR</td>
</tr>
<tr>
<td>PEEDCI</td>
<td>EDGE COLOUR INDEX</td>
</tr>
<tr>
<td>PEPA</td>
<td>PATTERN SIZE</td>
</tr>
<tr>
<td>PEPRPV</td>
<td>PATTERN REFERENCE POINT AND VECTORS</td>
</tr>
<tr>
<td>PEPARF</td>
<td>PATTERN REFERENCE POINT</td>
</tr>
<tr>
<td>PEADS</td>
<td>ADD NAMES TO SET</td>
</tr>
<tr>
<td>PERES</td>
<td>REMOVE NAMES FROM SET</td>
</tr>
<tr>
<td>PEIASF</td>
<td>INDIVIDUAL ASF</td>
</tr>
<tr>
<td>PEHRID</td>
<td>HLHSR IDENTIFIER</td>
</tr>
<tr>
<td>PELMT3</td>
<td>LOCAL MODELLING TRANSFORMATION 3</td>
</tr>
<tr>
<td>PELMT</td>
<td>LOCAL MODELLING TRANSFORMATION</td>
</tr>
<tr>
<td>PEGMT3</td>
<td>GLOBAL MODELLING TRANSFORMATION 3</td>
</tr>
<tr>
<td>PEGMT</td>
<td>GLOBAL MODELLING TRANSFORMATION</td>
</tr>
<tr>
<td>PEMCV3</td>
<td>MODELLING CLIPPING VOLUME 3</td>
</tr>
<tr>
<td>PEMCV</td>
<td>MODELLING CLIPPING VOLUME</td>
</tr>
<tr>
<td>PEMCLI</td>
<td>MODELLING CLIPPING INDICATOR</td>
</tr>
<tr>
<td>PERMCL</td>
<td>RESTORE MODELLING CLIPPING VOLUME</td>
</tr>
<tr>
<td>PEVWI1</td>
<td>VIEW INDEX</td>
</tr>
<tr>
<td>PEEXST</td>
<td>EXECUTE STRUCTURE</td>
</tr>
<tr>
<td>PELB</td>
<td>LABEL</td>
</tr>
</tbody>
</table>

modified 2 April 1993
PEAP      APPLICATION DATA
PEGSE     GSE
PEPKID    PICK ID
PEPSD3    POLYLINE SET 3 WITH DATA†
PEFSD3    FILL AREA SET 3 WITH DATA†
PETRSD    TRIANGLE STRIP 3 WITH DATA†
PEQMD3    QUADRILATERAL MESH 3 WITH DATA†
PESFS3    SET OF FILL AREA SET 3 WITH DATA†
PENBSC    NON-UNIFORM B-SPLINE CURVE†
PENBSS    NON-UNIFORM B-SPLINE SURFACE†
PEECA     CELL ARRAY 3 PLUS†
PETXC     TEXT COLOUR†
PEPMC     POLYMARKER COLOUR†
PEECD     EDGE COLOUR†
PEPLC     POLYLINE COLOUR†
PECAC     CURVE APPROXIMATION CRITERIA†
PEPLSM    POLYLINE SHADING METHOD†
PEIC      INTERIOR COLOUR†
PEBIC     BACK INTERIOR COLOUR†
PEBIS     BACK INTERIOR STYLE†
PEBISI    BACK INTERIOR STYLE INDEX†
PEAPR     REFLECTANCE PROPERTIES†
PEBAPR    BACK REFLECTANCE PROPERTIES†
PEISM     INTERIOR SHADING METHOD†
PEBISM    BACK INTERIOR SHADING METHOD†
PEIRE     REFLECTANCE EQUATION†
PEBIRE    BACK INTERIOR REFLECTANCE EQUATION†
PESAC     SURFACE APPROXIMATION CRITERIA†
PEFDM     FACE DISTINGUISHING MODE†
PEFCM     FACE CULLING MODE†
PELSS     LIGHT SOURCE STATE†
PEDCIN     DEPTH CUE INDEX†
PECMI     COLOUR MAPPING INDEX†
PERCM     RENDERING COLOUR MODEL†
† This is a SunPHIGS Extension based on PHIGS PLUS and is not part of the PHIGS standard.

EESN     Number of element types in the exclusion set.
EES(EESN) Array of element types in the exclusion set. Valid element type constants are in phigs77.h. The mapping from the FORTRAN constants to the actual element type names is given above. Excluding all element types will result in failure to find matching elements.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>FORTRAN Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERRIND</strong></td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td><strong>STATUS</strong></td>
<td>The search status indicator. Valid values defined in <code>phigs77.h</code> are PFAIL (failure) and PSUCC (success).</td>
</tr>
<tr>
<td><strong>FNDEP</strong></td>
<td>The sequence number of the element found. If status is failure, <code>FNDEP</code> is undefined.</td>
</tr>
</tbody>
</table>

**Execution**

If the start element position is less than zero, the search begins at element position zero, whose element type is considered NIL. If the start element position is greater than the number of elements in the structure, the search begins at the last element in the open structure. Otherwise, the search starts at the specified element.

If this element satisfies the search criteria, the element position is returned along with a status indicator of success. Otherwise the next element in the direction specified is considered. The search will continue until either an element fulfills the search criteria or the far end of the structure is reached. If the search is unsuccessful, then a status indicator of failure is returned.

The current element pointer and currently open structure (if any) are not used, and are unaffected.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>201</td>
<td>Ignoring function, the specified structure does not exist</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- LABEL (3P)
- INQUIRE ELEMENT POINTER (3P)
- INCREMENTAL SPATIAL SEARCH (3P)
### NAME
EMERGENCY CLOSE PHIGS – perform an emergency close of PHIGS

### SYNOPSIS

**C Syntax**

```c
void pemergency_close_phigs ( )
```

**FORTRAN Syntax**

`SUBROUTINE peclph`

### Required PHIGS Operating States

`(*, *, *, *)`

### DESCRIPTION

**Purpose**

Use EMERGENCY CLOSE PHIGS to close the PHIGS system even if the error state is ON. The purpose of this function is to close PHIGS and save as much graphical information as possible, in the event of errors that the application program cannot recover from.

**Execution**

When EMERGENCY CLOSE PHIGS is called, the following actions are performed:

- The open structure (if any) is closed.
- Any open archive files are closed.
- Any open workstations are updated and closed.
- PHIGS is closed.

If PHIGS is already closed when EMERGENCY CLOSE PHIGS is called, no action is taken.

### ERRORS

No Error

### SEE ALSO

CLOSE PHIGS (3P)

modified 2 April 1993
NAME
EMPTY STRUCTURE – remove all elements from structure

SYNOPSIS
C Syntax

```c
void pempty_struct ( struct_id )
Pint struct_id;  structure id
```

FORTRAN Syntax

```fortran
SUBROUTINE pemst ( STRID )
INTEGER STRID  structure identifier
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION
Purpose

Use EMPTY STRUCTURE to remove all the structure elements from a specified structure. The structure continues to exist as an empty structure even if there are no references to it in other structures. Any references that do exist are not changed by the EMPTY STRUCTURE subroutine.

C Input Parameter

```c
struct_id
```

Specifies the structure from which all structure elements should be removed.

FORTRAN Input Parameter

```fortran
STRID
```

Specifies the structure from which all structure elements should be removed.

Execution

The EMPTY STRUCTURE subroutine removes all the structure elements from structure identifier. The structure specified may be the open structure, another existing structure, or a non-existent structure.

If structure identifier is the currently-open structure or an existing structure, EMPTY STRUCTURE removes all elements from the structure and sets the element pointer to 0. Any references to this structure in other structures are not changed.

If the structure specified by structure identifier does not exist, EMPTY STRUCTURE creates a new empty structure.

ERRORS

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

SEE ALSO

DELETE ELEMENT RANGE (3P)
DELETE ELEMENTS BETWEEN LABELS (3P)
DELETE STRUCTURE (3P)
NAME

ERROR HANDLING – default PHIGS error handling routine

SYNOPSIS

C Syntax

void

perr_hand ( errnum, funcnum, fname )

Pint errnum; error number
Pint funcnum; number of function that detected the error
char *fname; name of error file

FORTRAN Syntax

SUBROUTINE perhnd ( ERRNR, FCTID, ERRFIL )

INTEGER ERRNR error number
INTEGER FCTID function identification
INTEGER ERRFIL error file

Required PHIGS Operating States

(∗, ∗, ∗, ∗)

DESCRIPTION

Purpose

When the error handling mode is ON, ERROR HANDLING is called by PHIGS functions that
detect an error in order to transfer the error information to the ERROR LOGGING subroutine
and then to the error file.

Because the error handling routine is separate from the error logging routine, the
application may write its own ERROR HANDLING procedure to specify different responses
to certain errors, and still use the ERROR LOGGING subroutine to record the source of the
errors.

C Input Parameters

errnum The number of the error detected. The error numbers and corresponding error
messages are listed with each function description in this reference.

funcnum The identification of the function that detected the error.

fname A pointer to a character string containing the name of the error file. The error file
is specified in the call to OPEN PHIGS.

FORTRAN Input Parameters

ERRNR The number of the error detected. The error numbers and corresponding error
messages for each subroutine are listed with each subroutine description in this reference.

FCTID The identification of the subroutine which detected the error. This number is
interpreted by the ERROR LOGGING subroutine in order to write the subroutine
name on the error file.

ERRFIL The FORTRAN Logical Unit Number (LUN) of the file on which to write the error
information. The error file is specified in the OPEN PHIGS subroutine.
**Execution**

When ERROR HANDLING is called by a PHIGS subroutine that has detected an error, it receives the error number, function number, and error file from the calling subroutine, and passes these parameters on in a call to ERROR LOGGING.

The ERROR LOGGING subroutine then writes the corresponding error message and function name onto the error file and returns to the calling function.

**ERRORS**

No Error

**SEE ALSO**

- ERROR LOGGING (3P)
- SET ERROR HANDLING MODE (3P)
- ESCAPE -1 (3P)
**NAME**  
ERROR LOGGING – log SunPHIGS errors on error file

**SYNOPSIS**

**C Syntax**

```c
void perr_log ( errnum, funcnum, fname )
Pint errnum; error number
Pint funcnum; number of function that detected the error
char *fname; name of error file
```

**FORTRAN Syntax**

```fortran
SUBROUTINE perlog ( ERRNR, FCTID, ERRFIL )
INTEGER ERRNR error number
INTEGER FCTID function identification
INTEGER ERRFIL error file
```

**Required PHIGS Operating States**

`(*, *, *, *)`

**DESCRIPTION**

**Purpose**

ERROR LOGGING is called to write an error message and the identification of the PHIGS function detecting the error to the *error file*. The error file is specified in the OPEN PHIGS function. See OPEN PHIGS for a description of the PHIGS interaction with the error file. ERROR LOGGING is always called by ERROR HANDLING but may also be called by the application.

**C Input Parameters**

- **errnum**  
The number of the error detected. The error numbers and corresponding error messages for each function are listed with each function description in this reference.

- **funcnum**  
The identification of the function which detected the error.

- **fname**  
A pointer to the character string containing the name of the error file specified in the OPEN PHIGS function.

**FORTRAN Input Parameters**

- **ERRNR**  
The number of the error detected. The error numbers and corresponding error messages for each function are listed with each function description in this reference.

- **FCTID**  
The identification of the function which detected the error.

- **ERRFIL**  
The FORTRAN Logical Unit Number (LUN) of the error file specified in the OPEN PHIGS function. See OPEN PHIGS for a description of valid values.

**Execution**  
When ERROR LOGGING is called it writes to the error file: the error number, a description of the error, and the function number of the function that detected the error.

modified 2 April 1993
ERROR LOGGING produces error messages of the form:

SunPHIGS error \texttt{<number>} in \texttt{<function name>}: \texttt{<error message text>}

ERRORS
No Error

SEE ALSO
ERROR HANDLING (3P)
SET ERROR HANDLING MODE (3P)
ESCAPE -1 (3P)
OPEN PHIGS (3P)
**NAME**

EVALUATE VIEW MAPPING MATRIX – generate transformation matrix to map 2D VRC window to 2D NPC viewport

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pevmm ( VWWNLM, PJVPLM, ERRIND, VWMPMT )
REAL VWWNLM(4) /* window limits (VRC) */
REAL PJVPLM(4) /* projection viewport limits (NPC) */
INTEGER ERRIND /* OUT error indicator */
REAL VWMPMT(3, 3) /* OUT view mapping matrix */
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use EVALUATE VIEW MAPPING MATRIX to calculate the viewing transformation matrix that transforms a specified window in View Reference Coordinates (VRC) to a specified viewport in Normalized Projection Coordinates (NPC).

The view mapping matrix returned by this function may be used as an argument to the SET VIEW REPRESENTATION function to specify the mapping from VRC to NPC. For more information, see the descriptions for the functions SET VIEW REPRESENTATION and EVALUATE VIEW ORIENTATION MATRIX in this function reference.

**C Input Parameters**

*map* A pointer to a Pview_map structure defining the view mapping. This structure is defined in phigs.h as follows:

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

*win* is a Plimit structure containing u and v VRC values defining the window to be mapped to NPC. Plimit is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* x minimum */
    Pfloat x_max; /* x maximum */
    Pfloat y_min; /* y minimum */
    Pfloat y_max; /* y maximum */
} Plimit;
```
The fields in Plimit define the VRC window as follows:

- Plimit.x_min = minimum u coordinate value
- Plimit.x_max = maximum u coordinate value
- Plimit.y_min = minimum v coordinate value
- Plimit.y_max = maximum v coordinate value

`proj_vp` is a Plimit structure containing the definition of the NPC viewport.

### C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error detected by this function.
- `m` A `Pmatrix` structure containing the 2D (3 × 3) transformation matrix that performs the specified view mapping. `Pmatrix` is defined in `phigs.h` as follows:
  ```
  typedef float Pmatrix[3][3];
  ```

### FORTRAN Input Parameters

- `VWWNLM` A four element array containing u and v VRC values defining the window limits. The array positions correspond to the window definition as follows:
  ```
  VWWNLM(1) = minimum u coordinate value
  VWWNLM(2) = maximum u coordinate value
  VWWNLM(3) = minimum v coordinate value
  VWWNLM(4) = maximum v coordinate value
  ```

- `PJVPLM` A four element array containing the definition of the NPC viewport. The array positions correspond to the viewport definition as follows:
  ```
  PJVPLM(1) = minimum x NPC coordinate value
  PJVPLM(2) = maximum x NPC coordinate value
  PJVPLM(3) = minimum y NPC coordinate value
  PJVPLM(4) = maximum y NPC coordinate value
  ```

### FORTRAN Output Parameters

- `ERRIND` The error number of any error detected by this function.
- `VWMPMT` The 2D (3 × 3) transformation matrix that performs the specified view mapping.

### Execution

If the input parameters are properly defined, `EVALUATE VIEW MAPPING MATRIX` returns a 2D (3 × 3) transformation matrix in the output parameter `view mapping matrix`. This transformation matrix performs the specified mapping from the VRC window to the NPC viewport.

The following restrictions apply to the viewport definition:

- u minimum < u maximum
- v minimum < v maximum

The following restrictions apply to the window definition:

- 0 <= x minimum < x maximum <= 1
- 0 <= y minimum < y maximum <= 1

110 modified 2 April 1993
Where zero and 1 are the NPC limits.
The range of VRC coordinate units is determined by the view orientation transformation.
The window to be mapped is specified in VRC by the window input parameter. It is a rectangle located on the \( n = 0 \) plane whose sides are parallel to the \( u \) and \( v \) axes. The rectangle is defined by minimum \( u \) and \( v \) values at the lower left-hand corner and by the maximum \( u \) and \( v \) values at the upper right-hand corner.
The viewport to which the window is mapped is specified in NPC by the viewport input parameter. It is a rectangle located in NPC space on the \( z = 0 \) plane whose sides are parallel to the \( x \) and \( y \) axes. The rectangle is defined by the \( x \) and \( y \) coordinates of the lower left corner and by the \( x \) and \( y \) coordinates of the upper right corner. The coordinate values must be within the closed unit square range of NPC space, \([0,1] \times [0,1] \).

Due to computational limitations, primitives may be rendered incorrectly when a perspective view with extreme clip limits is used. Specifically, when the ratio:

\[
\frac{\text{back}_{-}\text{plane} - \text{prp.z}}{\text{front}_{-}\text{plane} - \text{prp.z}}
\]

is greater than 1000, \( z \) buffering may be incorrect. When this ratio is even higher, the position and clippings of the primitive may also be incorrect. Correct this problem by moving the front plane further from the prp.

**ERRORS**

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **151** Ignoring function, invalid window; \( \text{XMIN} \geq \text{XMAX}, \text{YMIN} \geq \text{YMAX} \), or \( \text{ZMIN} > \text{ZMAX} \)
- **152** Ignoring function, invalid viewport; \( \text{XMIN} \geq \text{XMAX}, \text{YMIN} \geq \text{YMAX} \), or \( \text{ZMIN} > \text{ZMAX} \)
- **155** Ignoring function, the projection viewport limits are not within NPC range

**SEE ALSO**

EVALUATE VIEW MAPPING MATRIX 3 (3P)
**NAME**
EVALUATE VIEW MAPPING MATRIX 3 – generate transformation matrix to map 3D VRC window to 3D NPC viewport

**SYNOPSIS**

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

**FORTRAN Syntax**
```fortran
SUBROUTINE pevmm3 ( VWWNL, PJVPLM, PJTYPE, PJRX, PJRY, PJRZ, VPLD, BPLD, FPLD, ERRIND, VWMPMT )
REAL VWWNL(4) window limits (VRC)
REAL PJVPLM(6) projection viewport limits (NPC)
INTEGER PJTYPE projection type (PPARL, PPERS)
REAL PJRX, PJRY, PJRZ projection reference point (VRC)
REAL VPLD view plane distance (VRC)
REAL BPLD back plane distance (VRC)
REAL FPLD front plane distance (VRC)
INTEGER ERRIND /* OUT error indicator */
REAL VWMPMT(4, 4) OUT view mapping matrix
```

**DESCRIPTION**

**Purpose**

Use EVALUATE VIEW MAPPING MATRIX 3 to calculate the viewing transformation matrix that transforms a specified 3D window in View Reference Coordinates (VRC) to a specified 3D viewport in Normalized Projection Coordinates (NPC). The viewing transformation may be either a parallel or perspective transformation.

The view mapping matrix returned by this function may be used as an argument to the SET VIEW REPRESENTATION 3 function. For more information, see the descriptions of the functions SET VIEW REPRESENTATION 3 and EVALUATE VIEW ORIENTATION MATRIX 3.

**C Input Parameters**

*map* A pointer to a Pview_map3 structure defining the view mapping. This structure is defined in phigs.h as follows:
```c
typedef struct {
    Plimit win; /* window limits */
    Plimit3 proj_vp; /* viewport limits */
    Pproj_type proj_type; /* projection type */
    Ppoint3 proj_ref_point; /* proj. ref. point */
    Pfloat view_plane; /* view plane distance */
    Pfloat back_plane; /* back plane distance */
} Pview_map3;
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**modified 2 April 1993**
Pfloat front_plane; /* front plane distance */
}
Pview_map3;

*front plane distance*

\( Pview_map3 \)

\( \text{win} \) is a Plimit structure containing \( u \) and \( v \) VRC values defining the window to be mapped to NPC. Plimit is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min; /* x minimum */
    Pfloat x_max; /* x maximum */
    Pfloat y_min; /* y minimum */
    Pfloat y_max; /* y maximum */
} Plimit;

The fields in Plimit define the VRC window as follows:

\( \text{Plimit.x_min} = \text{minimum } u \text{ coordinate value} \)
\( \text{Plimit.x_max} = \text{maximum } u \text{ coordinate value} \)
\( \text{Plimit.y_min} = \text{minimum } v \text{ coordinate value} \)
\( \text{Plimit.y_max} = \text{maximum } v \text{ coordinate value} \)

\( \text{proj_vp} \) is a Plimit3 structure containing the definition of the NPC viewport. The min and max fields correspond to the back lower left and front upper right coordinates of the viewport volume, respectively. Plimit3 is defined in phigs.h as follows:

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

\( \text{proj_type} \) is the view project type. Pproj_type is defined in phigs.h as follows:

typedef enum {
    PTYPE_PARAL, /* Parallel projection */
    PTYPE_PERSPECT, /* Perspective projection */
} Proj_type;

\( \text{proj_ref_point} \) is the project reference point, defined in VRC. Ppoint3 is defined in phigs.h as follows:

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

modified 2 April 1993
Pfloat  z;      /* z coordinate */
} Ppoint3;

view_plane
is the VRC location of the view plane on the n axis of the VRC coordination system.

back_plane
is the VRC location of the back plane on the n axis of the VRC coordination system.

front_plane
is the VRC location of the front plane on the n axis of the VRC coordination system.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

m
A Pmatrix structure containing the 3D (3×3) transformation matrix that performs the specified view mapping. Pmatrix3 is defined in phigs.h as follows:
typedef Pfloat Pmatrix3[4][4];

FORTRAN Input Parameters

VWWNLM  A four-element array containing the u and v VRC values defining the window limits. The array positions correspond to the window definition as follows:

VWWNLM(1) = minimum u coordinate value
VWWNLM(2) = maximum u coordinate value
VWWNLM(3) = minimum v coordinate value
VWWNLM(4) = maximum v coordinate value

PJVPLM  A six element array containing the definition of the NPC viewport. The array positions correspond to the viewport definition as follows:

PJVPLM(1) = minimum x NPC coordinate value
PJVPLM(2) = maximum x NPC coordinate value
PJVPLM(3) = minimum y NPC coordinate value
PJVPLM(4) = maximum y NPC coordinate value
PJVPLM(5) = minimum z NPC coordinate value
PJVPLM(6) = maximum z NPC coordinate value

Minimum and maximum correspond to the front-lower-left and back-upper-right coordinates of the viewport volume, respectively.

PJTYPE  The projection type. Valid values as defined in phigs77.h are:

- PPARL  Parallel
- PPER  Perspective

PJRX, PJRY, PJRZ
The Projection Reference Point coordinates, defined in VRC.
**FORTRAN Output Parameters**

- **VPLD** The VRC location of the view plane on the $n$ axis of the VRC coordinate system.
- **BPLD** The VRC location of the back plane on the $n$ axis of the VRC coordinate system.
- **FPLD** The VRC location of the front plane on the $n$ axis of the VRC coordinate system.

**ERRIND** The error number of any error detected by this function.

**VWMPMT** The 3D ($4 \times 4$) transformation matrix that performs the specified view mapping.

**Execution**

If the input parameters are properly defined, `EVALUATE VIEW MAPPING MATRIX 3` returns a 3D ($4 \times 4$) transformation matrix in the output parameter view mapping matrix. This transformation matrix performs the specified mapping from the VRC window to the NPC viewport.

The front plane, back plane, and view plane all define planes in VRC space parallel to the $uv$ plane of the VRC system. The location of front and back along the $n$ axis of VRC defines the front and back of the volume of VRC that will be mapped to NPC. The view plane locates the view window along the VRC $n$ axis, and window defines the size of the view window by specifying maximum and minimum $u$ and $v$ values that establish the edges of the window. These values taken together establish the volume of VRC space that is mapped into the NPC viewport.

The type of projection may be parallel or perspective. The *projection reference point* orients the projectors defining the surfaces of the view volume. If the projection type is parallel, the projectors are all parallel to the vector joining the projection reference point and the center of the view window (located on the view plane). If the projection type is perspective, the projectors all converge at the projection reference point. Thus, the view volume is a parallelepiped for parallel views, and a portion of a double rectangular cone for perspective views.

Due to computational limitations, primitives may be rendered incorrectly when a perspective view with extreme clip limits is used. Specifically, when the ratio:

$\frac{(\text{back\_plane} - \text{prp\_z})}{(\text{front\_plane} - \text{prp\_z})}$

is greater than 1000, $z$ buffering may be incorrect. When this ratio is even higher, the position and clipping of the primitive may also be incorrect. Correct this problem by moving the front plane further from the prp.

**ERRORS**

- **002** Ignoring function, function requires state ($\text{PHOP}, *, *, *$)
- **151** Ignoring function, invalid window; $\text{XMIN} \geq \text{XMAX}, \text{YMIN} \geq \text{YMAX},$ or $\text{ZMIN} > \text{ZMAX}$
- **152** Ignoring function, invalid viewport; $\text{XMIN} \geq \text{XMAX}, \text{YMIN} \geq \text{YMAX},$ or $\text{ZMIN} > \text{ZMAX}$
- **158** Ignoring function, front plane and back plane distances are equal when $z$-extent of the projection viewport is zero
- **162** Ignoring function, the projection reference point is between the front and back planes

modified 2 April 1993 115
163 Ignoring function, the projection reference point cannot be positioned on the view plane
164 Ignoring function, the back plane is in front of the front plane
155 Ignoring function, the projection viewport limits are not within NPC range

SEE ALSO EVALUATE VIEW MAPPING MATRIX (3P)
NAME
EVALUATE VIEW ORIENTATION MATRIX – calculate transformation matrix that orients View Reference Coordinate (VRC) system in the World Coordinate (WC) system

SYNOPSIS

C Syntax

```c
void peval_view_ori_matrix ( vrp, vup, error_ind, m )
Ppoint *vrp;
Pvec *vup;
Pint *error_ind;
matrix m;
```

FORTRAN Syntax

```fortran
SUBROUTINE pevom ( VWRX, VWRY, VUPX, VUPY, ERRIND, VWORMT )
REAL VWRX, VWRY
REAL VUPX, VUPY
INTEGER ERRIND
REAL VWORMT(3, 3)
```

Required PHIGS Operating States

(_PHOP, *, *, * )

DESCRIPTION

Purpose

Use EVALUATE VIEW ORIENTATION MATRIX to calculate a view orientation matrix, used to transform World Coordinates (WC) to View Reference Coordinates (VRC). This matrix establishes the u, v, and n VRC axes in relation to WC space. The view orientation matrix calculated by this function can be passed as an argument to SET VIEW REPRESENTATION. See the descriptions of the functions SET VIEW REPRESENTATION and EVALUATE VIEW MAPPING MATRIX in this function reference for more information.

C Input Parameters

* `vrp` A pointer to a Ppoint structure containing x and y WC values that specify the view reference point. Ppoint is defined in phigs.h as follows:

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

* `vup` A pointer to a Pvec structure containing x and y WC values that specify the view up vector. Pvec is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat delta_x;    /* x magnitude */
    Pfloat delta_y;    /* y magnitude */
} Pvec;
```
C Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_ind</td>
<td>A pointer to the location to store the error number of any error detected by this function.</td>
</tr>
<tr>
<td>m</td>
<td>The 2D homogeneous (3 × 3) view orientation matrix. Pmatrix is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef float Pmatrix[3][3];</td>
</tr>
</tbody>
</table>

FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VWRX, VWRY</td>
<td>The x and y WC values that specify the view reference point.</td>
</tr>
<tr>
<td>VUPX, VUPY</td>
<td>The x and y WC values that specify the view up vector.</td>
</tr>
</tbody>
</table>

FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td>VWORMT</td>
<td>The 2D homogeneous (3 × 3) view orientation matrix.</td>
</tr>
</tbody>
</table>

Execution

If the input parameters are consistent and well-defined, EVALUATE VIEW ORIENTATION MATRIX returns a 2D homogeneous (3 × 3) transformation matrix in the output parameter view orientation matrix. This matrix transforms WC to VRC according to the values of the input parameters.

The view reference point defines the point in WC space that is to be used as the origin of the VRC system. It is specified as a 2D point in the z = 0 plane of the WC system and is usually a point on or near the object to be viewed.

The view up vector defines the up direction of the VRC system. It is specified as a 2D vector relative to the view reference point. This vector defines a direction in the WC z = 0 plane. This direction becomes the v axis of the VRC system. The n axis of VRC is parallel to the z axis of the WC system, and the u axis is determined so that u, v, and n axes form a right-handed coordinate system.

ERRORS

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

160 Ignoring function, the view up vector has length zero

SEE ALSO

EVALUATE VIEW ORIENTATION MATRIX 3 (3P)
NAME
EVALUATE VIEW ORIENTATION MATRIX 3 – calculate transformation matrix that orients the View Reference Coordinate (VRC) system in the World Coordinate (WC) system

SYNOPSIS
C Syntax
void
peval_view_ori_matrix3 ( vrp, vpn, vup, error_ind, m )

Ppoint3 *vrp; view reference point
Pvec3 *vpn; view plane normal
Pvec3 *vup; view up vector
Pint *error_ind; OUT error indicator
Pmatrix3 m; OUT view orientation matrix

FORTRAN Syntax
SUBROUTINE pevom3 ( VWRX, VWRY, VWRZ, VPNX, VPNY, VPNZ, VUPX, VUPY, VUPZ, ERRIND, VWORMT )
REAL VWRX, VWRY, VWRZ view reference point (WC)
REAL VPNX, VPNY, VPNZ view plane normal (WC)
REAL VUPX, VUPY, VUPZ view up vector (WC)
INTEGER ERRIND OUT error indicator
REAL VWORMT(4, 4) OUT view orientation matrix

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use EVALUATE VIEW ORIENTATION MATRIX 3 to calculate a view orientation matrix, used to transform World Coordinates (WC) to View Reference Coordinates (VRC). This matrix establishes the u, v and n VRC axes in relation to World Coordinate Space. The view orientation matrix calculated by this function can be passed as an argument to SET VIEW REPRESENTATION 3.

See the descriptions of the functions SET VIEW REPRESENTATION 3 and EVALUATE VIEW MAPPING MATRIX 3 in this function reference for more information.

C Input Parameters
vrp A pointer to a Ppoint3 structure containing x, y and z world coordinates that specify the view reference point. Ppoint3 is defined in phigs.h as follows:

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

modified 2 April 1993
A pointer to a Pvec3 structure containing x, y, and z World Coordinate values that specify the view plane normal vector. Pvec3 is defined in phigs.h as follows:

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

A pointer to a Pvec3 structure containing x, y, and z world coordinate values that specify the view up vector. Pvec3 is defined above.

### C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error detected by this function.

- **m**
  The 3D homogeneous (4 × 4) view orientation matrix. Pmatrix3 is defined in phigs.h as follows:

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

### FORTRAN Input Parameters

- **VWRX, VWRY, VWRZ**
  The x, y, and z world coordinate values that specify the view reference point.

- **VPNX, VPNY, VPNZ**
  The x, y and z world coordinate values that specify the view plane normal vector.

- **VUPX, VUPY, VUPZ**
  The x, y and z world coordinate values that specify the view up vector.

### FORTRAN Output Parameters

- **ERRIND**
  The error number of any error detected by this function.

- **VWORMT**
  The 3D homogeneous (4 × 4) view orientation matrix.

### Execution

If the input parameters are consistent and well-defined, `EVALUATE VIEW ORIENTATION MATRIX 3` returns a 3D homogeneous (4 × 4) transformation matrix in the output parameter `view orientation matrix`. This matrix transforms World Coordinates to View Reference Coordinates according to the values of the input parameters.

The **view reference point** defines the point in World Coordinate Space that is to be used as the origin of the View Reference Coordinate system. It is specified as a 3D world coordinate point and is usually a point on or near the object to be viewed.

The **view plane normal** is a 3D vector defined in World Coordinates relative to the view reference point. It defines the n axis of the VRC system. The plane in WC that contains the view reference point and is perpendicular to the view plane normal is called the **view reference plane**.
The **view up vector** is a 3D vector defined in World Coordinates relative to the view reference point. The projection of this vector onto the view reference plane parallel to the **view plane normal** determines the \( v \) axis of the VRC system.

The \( u \) axis of VRC is determined so that the \( uvn \) axes form a right-handed coordinate system.

**ERRORS**

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

159 Ignoring function, the view plane normal vector has length zero

160 Ignoring function, the view up vector has length zero

161 Ignoring function, the view up and view plane normal vectors are parallel thus the viewing coordinate system cannot be established

**SEE ALSO**

EVALUATE VIEW ORIENTATION MATRIX (3P)
### NAME
EXECUTE STRUCTURE – create structure element to invoke another structure

### SYNOPSIS

#### C Syntax
void
pexec_struct ( struct_id )

Pint struct_id;  //structure identifier

#### FORTRAN Syntax
SUBROUTINE pexst ( STRID )

INTEGER STRID  //structure identifier

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION

#### Purpose
EXECUTE STRUCTURE puts a structure element containing the `structure identifier` into the currently-open structure according to the current edit mode. If no structure exists with the `structure identifier`, then an empty structure is created with that identifier. EXECUTE STRUCTURE elements are used to define hierarchical structure networks by invoking one structure from within another during structure traversal.

If the current edit mode is INSERT, the EXECUTE STRUCTURE element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the EXECUTE STRUCTURE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new EXECUTE STRUCTURE element.

#### C Input Parameter
`struct_id`
Identifies structure to be invoked from the currently-open structure.

#### FORTRAN Input Parameter
`STRID` Identifies structure to be invoked from the currently-open structure.

#### Execution
When an EXECUTE STRUCTURE element is traversed, the traversal performs the following in order:

1. Suspends the traversal of the current structure.
2. Saves the current states of:
   - All primitive attributes.
   - Local and global modelling transformations.
   - The view index.
   - HLHSR and PICK identifiers.
   - The name set.
   - Modelling clipping limits and clipping indicator.
3. Sets the global modelling transformation to the current composite modelling transformation. Sets the local modelling transformation to an identity matrix.
4. Traverses the structure specified by *structure identifier*. This process will be repeated for any EXECUTE STRUCTURE elements encountered in the structure with the *structure identifier* and its subordinate structures before returning to the current structure.

5. Restores all the saved states.

6. Resumes traversal of the current structure.

If *structure identifier* itself contains an EXECUTE STRUCTURE element, traversal of the structure with the *structure identifier* is also suspended, and traversal continues with that structure invocation. At the end of a structure (that is, after its last element), traversal returns to the invoking structure, restoring the saved state, and continuing with later elements in the invoking structure, until its end. This continues until the entire structure network is traversed (that is, until traversal of the posted structure is completed).

The application must avoid recursive structure networks: no EXECUTE STRUCTURE element may invoke any ancestor structure. The PHIGS implementation is not required to check for this problem, and SunPHIGS does not.

Each subordinate structure inherits the current attribute values from its parent structure, except for the modelling transformations, as described in step 3 above. Unless the attributes are explicitly changed with the appropriate SET element, they retain the last value set in a structure higher in the structure network hierarchy. On the other hand, when traversal of a structure is suspended, the current attribute values are saved and later restored when traversal of the structure resumes. This means that structures can only affect the display of structures subordinate to them in the network; no structure can affect any attributes of its parent.

Implementations vary in the depth of structure execution supported, and in the efficiency of EXECUTE STRUCTURE traversal. SunPHIGS is limited only by available virtual memory, and EXECUTE STRUCTURE is quite efficient.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

POST STRUCTURE (3P)
DELETE STRUCTURE NETWORK (3P)
CHANGE STRUCTURE REFERENCES (3P)
CHANGE STRUCTURE IDENTIFIER AND REFERENCES (3P)
INQUIRE PATHS TO ANCESTORS (3P)
INQUIRE PATHS TO DESCENDANTS (3P)
**NAME**

FILL AREA – create structure element specifying 2D fill area primitive

**SYNOPSIS**

**C Syntax**

```c
void pfill_area ( point_list )
Ppoint_list *point_list;  /* array of points */
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pfa ( N, PXA, PYA )
INTEGER N  /* number of points */
REAL PXA(N), PYA(N)  /* coordinates of points (MC) */
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

The FILL AREA function puts a structure element containing the 2D specification of a FILL AREA primitive into the currently-open structure. The FILL AREA primitive is a closed polygonal area defined by a series of two-dimensional points in Modelling Coordinates. The z coordinate is assumed to be 0.

**C Input Parameters**

`point_list`

A pointer to a list `num_points` long of Ppoint structures containing the x and y coordinates for each point used to define the FILL AREA polygon. The Ppoint_list structure is defined in phigs.h as follows:

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

Ppoint is defined as follows:

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

**FORTRAN Input Parameters**

`N`

The number of points used to define the FILL AREA polygon. You must specify at least three points; a FILL AREA element that has less than three points will be ignored when the structure is traversed.

`PXA` An array of `N` real values containing the x coordinates of the FILL AREA polygon.

`PYA` An array of `N` real values containing the y coordinates of the FILL AREA polygon.
When the structure is traversed, the FILL AREA element will draw a closed polygonal area. The appearance of the interior of the defined area is determined by the attributes listed below. These attributes control the type of fill (SOLID, EMPTY, HOLLOW, HATCH, and PATTERN) and the colour with which the primitive is drawn.

The points are specified in Modelling Coordinates. These may be any coordinate units that are convenient to the application. At traversal, these coordinate values are transformed by the current Local and Global Modelling Transformations, the View Representation selected by the current view index, and the Workstation Transformation current on the workstation to which the structure is posted.

The attributes listed below are used to display the FILL AREA primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>interior colour</td>
<td>interior colour index ASF</td>
</tr>
<tr>
<td>back interior colour</td>
<td>back interior colour ASF</td>
</tr>
<tr>
<td>interior style</td>
<td>interior style ASF</td>
</tr>
<tr>
<td>back interior style</td>
<td>back interior style ASF</td>
</tr>
<tr>
<td>interior style index</td>
<td>interior style index ASF</td>
</tr>
<tr>
<td>back interior style index</td>
<td>back interior style index ASF</td>
</tr>
<tr>
<td>interior shading method</td>
<td>interior shading method ASF</td>
</tr>
<tr>
<td>back interior shading method</td>
<td>back interior shading method ASF</td>
</tr>
<tr>
<td>interior reflectance equation</td>
<td>interior reflectance equation ASF</td>
</tr>
<tr>
<td>back interior reflectance equation</td>
<td>back interior reflectance equation ASF</td>
</tr>
<tr>
<td>reflectance properties</td>
<td>reflectance properties ASF</td>
</tr>
<tr>
<td>back reflectance properties</td>
<td>back reflectance properties ASF</td>
</tr>
<tr>
<td>interior index</td>
<td></td>
</tr>
<tr>
<td>face distinguishing mode</td>
<td></td>
</tr>
<tr>
<td>face culling mode</td>
<td></td>
</tr>
<tr>
<td>depth cue index</td>
<td></td>
</tr>
<tr>
<td>light source state</td>
<td></td>
</tr>
<tr>
<td>name set</td>
<td></td>
</tr>
</tbody>
</table>

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

FILL AREA 3 (3P)
FILL AREA SET (3P)
FILL AREA SET 3 WITH DATA (3PP)

modified 2 April 1993
NAME
FILL AREA 3 – create structure element specifying 3D fill area primitive

SYNOPSIS
C Syntax
void
pfill_area3 ( point_list )

FORTRAN Syntax
SUBROUTINE pfa3 ( N, PXA, PYA, PZA )

Description
Purpose
The FILL AREA 3 function puts a structure element containing the 3D specification of a FILL AREA 3 primitive into the currently-open structure. The FILL AREA 3 primitive is a closed polygonal area defined by a series of three-dimensional Modelling Coordinate points.

Note: The FILL AREA 3 primitive INTERIOR attributes control the representation of the interior of the area defined by the specified coordinate points. FILL AREA 3 does not have separate edge attributes. See the FILL AREA SET and FILL AREA SET 3 primitives for independent control of the edge properties.

If the current edit mode is INSERT, the structure element created by the FILL AREA 3 subroutine is inserted into the open structure after the element pointed to by the structure’s element pointer. If the edit mode is REPLACE, the FILL AREA 3 element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new FILL AREA 3 element.

C Input Parameters
point_list
A pointer to a list num_points long of Ppoint3 structures containing the x, y, and z coordinates for each point used to define the FILL AREA 3 polygon. The Ppoint_list3 structure is defined in phigs.h as follows:

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

Ppoint3 is defined as follows:

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

Parameters

\( N \)  
The number of points used to define the FILL AREA 3 polygon. You must specify at least three points; a FILL AREA 3 element that has less than three points will be ignored when the structure is traversed.

\( PXA \)  
An array of \( N \) real values containing the \( x \) coordinates of the FILL AREA 3 polygon.

\( PYA \)  
An array of \( N \) real values containing the \( y \) coordinates of the FILL AREA 3 polygon.

\( PZA \)  
An array of \( N \) real values containing the \( z \) coordinates of the FILL AREA 3 polygon.

Execution

When the structure is traversed, the FILL AREA 3 element will draw a closed polygonal area. The appearance of the interior of the defined area is determined by the attributes listed below. These attributes control the type of fill (SOLID, EMPTY, HOLLOW, HATCH, and PATTERN) and the colour in which the primitive is drawn.

The coordinate points are specified in Modelling Coordinate points. Modelling Coordinate units may be any that are convenient to the application. At traversal, these coordinate values are transformed by the current Local and Global Modelling Transformations, the View Representation selected by the current view index, and the Workstation Transformation current on the workstation to which the structure is posted.

Attributes Applied

The attributes listed below are used to display the FILL AREA 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- interior colour
- back interior colour
- interior style
- back interior style
- interior style index
- back interior style index
- interior shading method
- back interior shading method
- interior reflectance equation
- back interior reflectance equation
- reflectance properties
- back reflectance properties
- interior index
- face distinguishing mode
- face culling mode

modified 2 April 1993
depth cue index  
light source state  
name set  

**ERRORS**  
005 Ignoring function, function requires state (PHOP, *, STOP, *)  

**SEE ALSO**  
FILL AREA (3P)  
FILL AREA SET 3 (3P)  
FILL AREA SET 3 WITH DATA (3PP)
NAME
FILL AREA SET – create structure element specifying 2D fill area set primitive

SYNOPSIS
C Syntax

```c
void pfill_area_set ( point_list_list )

Ppoint_list_list *point_list_list;  // list of point lists
```

FORTRAN Syntax

```fortran
SUBROUTINE pfas ( NPL, IXA, PXA, PYA )

INTEGER NPL  // number of point lists
INTEGER IXA(NPL)  // array of end indices for the point lists
REAL PXA(*), PYA(*)  // coordinates of points (MC)
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION
Purpose

The FILL AREA SET function puts a structure element containing the 2D specification of a FILL AREA SET primitive into the currently-open structure. The FILL AREA SET primitive is a group of implicitly closed polygonal areas. This allows for specifying areas with holes or disjoint regions. The subroutine parameters specify the number of fill areas to be drawn, the number of points used to define each, and the Modelling coordinates of each point. The z coordinates are assumed to be zeroes. Each set of points defines a separate closed area. All of the areas specified in a single function call are drawn using the current values of the FILL AREA SET attributes listed below.

If the current edit mode is INSERT, the structure element created by the FILL AREA SET subroutine is inserted into the open structure after the element pointed to by the structure’s element pointer. If the edit mode is REPLACE, the new FILL AREA SET element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new FILL AREA SET element.

C Input Parameters

```
point_list_lists
```

A pointer to a Ppoint_list_list structure, which is defined in phigs.h as follows:

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

The `num_point_lists` component specifies the number of fill areas in the set. The `point_lists` component is a pointer to a list of Ppoint_list structures, each of which defines one of the fill areas in the set. Ppoint_list is defined in phigs.h as follows:

```c
typedef struct {
    Pint num_points;  // number of Ppoint structures in the list */
    Ppoint *points;  // list of points */
```
The `num_points` component specifies the number of points used to define the fill area. The `points` component is a pointer to a list, `num_points` long, of `Ppoint` structures that contain the Modelling Coordinates of each vertex of the fill area. `Ppoint` is defined in `phigs.h` as follows:

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

### FORTRAN Input Parameters

- **NPL** The number of point lists, or sets of points, to be specified. Each set of points defines a separate closed area.
- **IXA** An array of `NPL` integers containing the end indices in the `PXA` and `PYA` arrays for each set of points.
- **PXA** An array of `N` real values containing the `x` coordinates of the FILL AREA SET.
- **PYA** An array of `N` real values containing the `y` coordinates of the FILL AREA SET.

The first fill area in the set is defined by the points from 1 to `IXA(1)` in the `PXA` and `PYA` arrays, the second fill area is defined by the points from `IXA(1) + 1` to `IXA(2)`, and so on.

### Execution

When the structure is traversed, the FILL AREA SET element draws the specified number of closed polygonal areas using the given points. Each fill area forms a boundary by connecting the specified points in order. SunPHIGS will implicitly close each area by extending the boundary of the fill area from the last point to the first.

The points are specified in Modelling Coordinates. These may be any coordinate units that are convenient to the application. At traversal, these coordinates are transformed by the current Local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

All the specified areas are drawn using the current values of the primitive attributes listed below. The edge attributes, `EDGE FLAG`, `EDGETYPE`, `EDGEBASE SCALE FACTOR` and `EDGE COLOUR INDEX` allow you to control whether, and how, the edges of the areas are displayed independently of the interior representation.

### Using Edge Attributes With Interior Style

If edges are not displayed, and the INTERIOR STYLE is HOLLOW, only the boundaries of the fill areas are drawn using the interior colour. If edges are displayed when the INTERIOR STYLE is HOLLOW, the edges will overlay the boundaries using the edge colour. If the `EDGETYPE` is a broken line, the boundaries will show through the breaks in the edge representation.

### Attributes Applied

The attributes listed below are used to display the FILL AREA SET primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they...
can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- interior colour
- back interior colour
- interior style
- back interior style
- interior style index
- back interior style index
- interior shading method
- back interior shading method
- interior reflectance equation
- back interior reflectance equation
- reflectance properties
- back reflectance properties
- interior index
- edge colour
- edge flag
- edgetype
- edgewidth scale factor
- edge index
- face distinguishing mode
- face culling mode
- depth cue index
- light source state
- name set

**ERRORS** 005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**
- FILL AREA (3P)
- FILL AREA SET 3 (3P)
- FILL AREA SET 3 WITH DATA (3PP)

modified 2 April 1993
NAME
FILL AREA SET 3 – create structure element specifying 3D fill area set primitive

SYNOPSIS
C Syntax
void
pfill_area_set3 ( point_list_list )

FORTRAN Syntax
SUBROUTINE pfas3 ( NPL, IXA, PXA, PYA, PZA )

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
The FILL AREA SET 3 function puts a structure element containing the 3D specification of a
FILL AREA SET 3 primitive into the currently open structure. The FILL AREA SET 3 primitive
is a group of implicitly closed polygonal areas. This allows for specifying areas with
holes or disjoint coplanar regions. The subroutine parameters specify the number of
areas to be drawn, the number of points used to define each area, and the coordinates of
each point. Each set of points defines a separate closed area. All of the areas specified in
a single function call are drawn using the current values of the FILL AREA SET 3 attributes
listed below.

If the current edit mode is INSERT, the FILL AREA SET 3 element is inserted into the open
structure after the element pointed to by the structure’s element pointer. If the edit mode
is REPLACE, the FILL AREA SET 3 element replaces the element pointed to by the element
pointer. In either case, the element pointer is updated to point to the new FILL AREA SET 3
element.

C Input Parameters
point_list_lists
A pointer to a Ppoint_list_list3 structure, which is defined in phigs.h as follows:

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

The num_point_lists component specifies the number of fill areas in the set. The
point_lists component is a pointer to a list of Ppoint_list3 structures, each of which
defines one of the fill areas in the set. Ppoint_list3 is defined in phigs.h as
follows:

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

modified 2 April 1993
Ppoint3 *points; /* list of points */
} Ppoint_list3;

The `num_points` component specifies the number of points used to define the fill area. The `points` component is a pointer to a list, `num_points` long, of `Ppoint3` structures containing the coordinates of the fill area vertices in Modelling Coordinates. `Ppoint3` is defined in `phigs.h` as follows:

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

### FORTRAN Input Parameters

- **NPL** The number of point lists, or sets of points, to be specified. Each set of points defines a separate closed area.
- **IXA** An array of `NPL` integers containing the end indices in the `PXA`, `PYA`, and `PZA` arrays for each set of points.
- **PXA** An array of `N` real values containing the `x` coordinates of the FILL AREA SET.
- **PYA** An array of `N` real values containing the `y` coordinates of the FILL AREA SET.
- **PZA** An array of `N` real values containing the `z` coordinates of the FILL AREA SET.

The first fill area in the set is defined by the points from 1 to `IXA(1)` in the `PXA`, `PYA`, and `PZA` arrays, the second fill area is defined by the points from `IXA(1) + 1` to `IXA(2)`, and so on.

### Execution

When the structure is traversed, the FILL AREA SET 3 element draws the specified number of closed polygonal areas using the given points. Each fill area forms a boundary by connecting the specified points in order. SunPHIGS will implicitly close each area by extending the boundary of the fill area from the last point to the first.

The entire fill area set must be coplanar.

The points are specified in Modelling Coordinates. These may be any coordinate units that are convenient to the application. At traversal, these coordinates are transformed by the current Local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

All the specified areas are drawn using the current values of the primitive attributes listed below. The edge attributes, `EDGE FLAG`, `EDGETYPE`, `EDGEWIDTH SCALE FACTOR` and `EDGE COLOUR INDEX`, allow you to control whether, and how, the edge of the areas is displayed independently of the interior representation.
Using Edge Attributes with Interior Style

If edges are not displayed, and the INTERIOR STYLE is HOLLOW, only the boundaries of the fill areas are drawn using the interior colour. If the edges are displayed when the INTERIOR STYLE is HOLLOW, the edges will overlay the boundaries using the edge colour. If the EDGETYPE is a broken line, the boundaries will show through the breaks in the edge representation.

Attributes Applied

The attributes listed below are used to display the FILL AREA SET 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- interior colour
- back interior colour
- interior style
- back interior style
- interior style index
- back interior style index
- interior shading method
- back interior shading method
- interior reflectance equation
- back interior reflectance equation
- reflectance properties
- back reflectance properties
- interior index
- edge colour
- edge flag
- edgetype
- edgewidth scale factor
- edge index
- face distinguishing mode
- face culling mode
- depth cue index
- light source state
- name set

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

FILL AREA 3 (3P)
FILL AREA SET (3P)
FILL AREA SET 3 WITH DATA (3PP)

modified 2 April 1993
NAME
FLUSH DEVICE EVENTS – remove all entries in input queue generated by specified logical input device

SYNOPSIS
C Syntax
void
pflush_events ( ws, class, dev )

Pint ws;   // workstation identifier
Pin_class class; // device class
Pint dev;   // logical input device number

FORTRAN Syntax
SUBROUTINE pflush ( WKID, ICL, IDNR )
INTEGER WKID  // workstation identifier
INTEGER ICL   // input class
INTEGER IDNR  // logical input device number

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use FLUSH DEVICE EVENTS to remove all the events in the input queue that were generated by a specified input device.

C Input Parameters
ws
The workstation identifier of the workstation associated with the input device.

class
The input class of the input device. Pin_class is an enumerated type defined in phigs.h that may take the following values:

PIN_NONE
PIN_LOC
PIN_STROKE
PIN_VAL
PIN_CHOICE
PIN_PICK
PIN_STRING
dev
The device number of the input device.

FORTRAN Input Parameters
WKID
The workstation identifier of the workstation associated with the input device.

ICL
The class of the input device. Valid classes as defined in phigs77.h are as follows:

PLOCAT Locator
PSTROK Stroke
PVVALUA Valuator
PCHOIC Choice
PPICK Pick
PSTRIN String
<table>
<thead>
<tr>
<th>IDNR</th>
<th>The device number of the input device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORS</td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>256</td>
<td>Warning, the input queue has overflowed</td>
</tr>
</tbody>
</table>

SEE ALSO

AWAIT EVENT (3P)
NAME
GET CHOICE – retrieve the CHOICE measure from the PHIGS current event report

SYNOPSIS
C Syntax
void
pget_choice ( in_status, choice )
Pin_status *in_status; OUT choice status
Pint *choice; OUT choice

FORTRAN Syntax
SUBROUTINE pgtch ( STAT, CHNR )
INTEGER STAT OUT status (POK, PNCHOI)
INTEGER CHNR OUT choice number

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use GET CHOICE to retrieve the measure (logical input value) of the Choice device from the current event report in the PHIGS state list.

A CHOICE device measure consists of a status and a choice number. Status indicates whether or not one of the possible values on the device was selected. The choice number indicates the value selected, if any.

C Output Parameters
in_status
PHIGS sets the variable pointed to by in_status to indicate whether or not one of the possible choices of the CHOICE device was selected. A value of PIN_STATUS_OK indicates that one of the possible values was selected. A value of PIN_STATUS_NO_IN indicates that none of the possible values was selected. Pin_status is defined in phigs.h as follows:

typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;

This function does not return the value PIN_STATUS_NONE, which is for use with REQUEST CHOICE.

choice
If the status returned is PIN_STATUS_OK, PHIGS sets the variable pointed to by choice to the value selected by the operator. The variable is not set if the status returned is PIN_STATUS_NO_IN.

FORTRAN Output Parameters
STAT
The measure’s choice status. Valid values as defined in phigs77.h are:
POK OK
PNCHOI No choice

modified 2 April 1993
**CHNR**  The measure’s choice number. This value is undefined if the status returned is PNCHOI.

**Execution**  The GET CHOICE function retrieves the measure of a CHOICE device from the current event report in the PHIGS state list. If *No Choice* is returned as the status, then choice number is undefined. If OK is returned as the status, then *choice number* contains the measure’s choice number.

When an input device that is set to Event mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a Choice event, then GET CHOICE is used to retrieve the measure from the current event report.

See INITIALIZE CHOICE 3 for a description of the available CHOICE devices and how their measure values are determined.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)

259  Ignoring function, the input device class of the current input report does not match the class being requested

**SEE ALSO**

INITIALIZE CHOICE (3P)

INITIALIZE CHOICE 3 (3P)

AWAIT EVENT (3P)
NAME
GET ITEM TYPE FROM METAFILE – get type and length of current item from metafile

SYNOPSIS
C Syntax

```c
void
pget_item_type ( ws, type, length )
Pint  ws;        workstation identifier
Pint  *type;     OUT item type
Pint  *length;   OUT item data record length
```

FORTRAN Syntax

```fortran
SUBROUTINE pgtitm ( WKID, TYPE, IDRL )
INTEGER WKID    workstation identifier
INTEGER TYPE    OUT item type
INTEGER IDRL    OUT item data record length (to be received from PRDITM)
```

Required PHIGS Operating States

( PHOP, WSOP, *, * )

DESCRIPTION

Note: This function has C and FORTRAN bindings, but its functionality is not implemented.

ERRORS

003 Ignoring function, function requires state ( PHOP, WSOP, *, * )
054 Ignoring function, the specified workstation is not open
058 Ignoring function, specified workstation is not of category MI
302 Ignoring function, no item is left in metafile input
303 Ignoring function, metafile item is invalid

modified 2 April 1993
**NAME**

GET LOCATOR – retrieve the 2D LOCATOR measure from the PHIGS current event report.

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pgtlc ( VIEWI, LPX, LPY )
INTEGER VIEWI OUT view index
REAL LPX, LPY OUT locator position in world coordinates
```

**Required PHIGS Operating States**

(HPHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

Use GET LOCATOR to retrieve the 2D components of the measure (logical input value) of the LOCATOR device from the current event report in the PHIGS state list.

A LOCATOR device measure consists of a position and a view index. Position is the World Coordinate (WC) point corresponding to the position on the workstation selected by the operator. The view index is the index of the view representation used to transform position from Normalized Projection Coordinates (NPC) to WC. See INITIALIZE LOCATOR and SET VIEW TRANSFORMATION INPUT PRIORITY for a description of how this view representation is determined. The workstation transform is used to transform the Device Coordinate (DC) position to a NPC position.

**C Output Parameters**

- **view_ind**
  - PHIGS sets the variable pointed to by `view_ind` to the index of the view representation used to transform the NPC locator position to a WC position.

- **loc_pos**
  - PHIGS sets the variable pointed to by `loc_pos` to the 2D world coordinate locator position corresponding to the device coordinate position selected by the operator. `Ppoint` is defined in phigs.h as follows:

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

**FORTRAN Output Parameters**

- **VIEWI**
  - The index of the view representation used to transform the NPC position to a WC position.

- **LPX, LPY**
  - The 2D locator position in WC.

140 modified 2 April 1993
Execution

GET LOCATOR retrieves the 2D components of the LOCATOR measure from the current event report in the PHIGS state list. The \( x \) and \( y \) components are returned; the \( z \) component is discarded.

When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a LOCATOR event, then GET LOCATOR is used to retrieve the measure from the current event report.

See INITIALIZE LOCATOR 3 for a description of the available LOCATOR devices and how their measure values are determined.

ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>259</td>
<td>Ignoring function, the input device class of the current input report does not match the class being requested</td>
</tr>
</tbody>
</table>

SEE ALSO

- INITIALIZE LOCATOR (3P)
- AWAIT EVENT (3P)
- GET LOCATOR 3 (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)

modified 2 April 1993
| NAME | GET LOCATOR 3 – retrieve the 3D LOCATOR measure from the PHIGS current event report |
| SYNOPSIS |  |
| C Syntax | void pget_loc3 (view_ind, loc_pos) |
| | Pint *view_ind; OUT view index |
| | Ppoint3 *loc_pos; OUT locator position |
| FORTRAN Syntax | SUBROUTINE pgtlc3 (VIEWI, LPX, LPY, LPZ) |
| | INTEGER VIEWI OUT view index |
| | REAL LPX, LPY, LPZ OUT locator position in world coordinates |
| Required PHIGS Operating States | (PHOP, WSOP, *, *) |
| DESCRIPTION Purpose | Use GET LOCATOR 3 to retrieve the measure (logical input value) of the LOCATOR device from the current event report in the PHIGS state list. |
| A LOCATOR device measure consists of a position and a view index. Position is the World Coordinate (WC) point corresponding to the position on the workstation selected by the operator. The view index is the index of the view used to transform position from Device Coordinates (DC) to WC. See INITIALIZE LOCATOR 3 and SET VIEW TRANSFORMATION INPUT PRIORITY for a description of how this view representation is determined. The workstation transform is used to transform the DC position to a NPC position. |
| C Output Parameters | view_ind |
| | PHIGS sets the variable pointed to by view_ind to the index of the view representation used to transform the NPC locator position to a WC position. |
| loc_pos | PHIGS sets the variable pointed to by loc_pos to the 3D world coordinate locator position corresponding to the device coordinate position selected by the operator. Ppoint3 is defined in phigs.h as follows: |
| | typedef struct { |
| | Pfloat x; /* x coordinate */ |
| | Pfloat y; /* y coordinate */ |
| | Pfloat z; /* z coordinate */ |
| } Ppoint3; |
| FORTRAN Output Parameters | VIEWI |
| | The index of the view representation used to transform the NPC position to a WC position. |
| LPX, LPY, LPZ | The 3D locator position in WC. |
**Execution**

GET LOCATOR 3 retrieves the LOCATOR measure from the current event report in the PHIGS state list.

When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a LOCATOR event, GET LOCATOR 3 is used to retrieve the measure from the current event report.

See INITIALIZE LOCATOR 3 for a description of the available LOCATOR devices and how their measure values are determined.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>259</td>
<td>Ignoring function, the input device class of the current input report does not match the class being requested</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INITIALIZE LOCATOR 3 (3P)
- AWAIT EVENT (3P)
- GET LOCATOR (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
NAME
GET PICK – retrieve the PICK measure from the PHIGS current event report

SYNOPSIS
C Syntax
void pget_pick ( depth, in_status, rpick )

Pint depth; depth of pick path to return
Pin_status *in_status; OUT pick status
Ppick_path *rpick; OUT pick path

FORTRAN Syntax
SUBROUTINE pgtpk ( IPPD, STAT, PPD, PP )
INTEGER IPPD depth of pick path to return
INTEGER STAT OUT status (POK, PNPICK)
INTEGER PPD OUT depth of actual pick path
INTEGER PP(3, IPPD) OUT pick path

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use GET PICK to retrieve the measure (logical input value) of the PICK device from
the current event report in the PHIGS state list.

A PICK device measure consists of a status and a pick path. Status indicates whether
a pick by the operator was successfully resolved. Pick path describes the location of
the picked primitive, if any, in the Central Structure Store (CSS). The pick filter of
a PICK device controls which output primitives on the device’s workstation are
pickable. By default, no output primitives are pickable. See SET PICK FILTER for
more information on the pick filter.

C Input Parameter
depth The maximum number of pick path elements to return. This may be more or less
than the actual path depth in the current event report.

C Output Parameters
in_status
PHIGS sets the variable pointed to by in_status to the event’s pick status. A value
of PIN_STATUS_OK indicates that an output primitive was successfully selected
by the operator. A value of PIN_STATUS_NO_IN indicates that a pick was attempted,
but no primitive was selected. Pin_status is defined in phigs.h as follows:

typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;

This function does not return the value PIN_STATUS_NONE, which is for use with REQUEST
PICK.

rpick PHIGS sets the variable pointed to by rpick to the event’s pick path. Ppick_path is

modified 2 April 1993
defined in phigs.h as follows:

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

A pick path is returned only if the status is PIN_STATUS_OK. The contents of rpick are not changed if the status is not PIN_STATUS_OK. The pick path depth indicates the number of elements in the event’s pick path as stored in the current event report. It is not affected by the depth input parameter; therefore, the number of elements returned in path_list may be less than depth.

path_list is an array of references defining the location of the picked primitive in the CSS.

Note: This array must be allocated by the calling program and the array pointer assigned to this field before calling this function.

The array must be at least long enough to hold the number of path elements indicated by the depth input parameter. Ppick_path_elem is defined in phigs.h as:

typedef struct {
    Pint struct_id;     /* structure identifier */
    Pint pick_id;     /* hierarchical pick identifier */
    Pint elem_pos;     /* element sequence number */
} Ppick_path_elem;

The struct_id, pick_id, and elem_pos are the structure identifier, pick id, and element number, respectively, of each element in the path. Each element but the last indicates the location and current pick id of an EXECUTE STRUCTURE structure element in the path to the selected primitive. The last element in the path indicates the location and current pick id of the selected output primitive.

FORTRAN Input Parameter

IPPD The maximum number of path elements to return. This may be more or less than the actual path depth in the current event report.

FORTRAN Output Parameters

STAT The measure’s pick status. Valid values as defined in phigs77.h are:

    POK    OK
    PNPICK No pick

PPD The number of elements in the measure’s path. This value is undefined if the status returned is PNPICK. This is the depth value contained in the measure of the current event report, and is not affected by the maximum depth to return parameter, IPPD. Thus, the number of elements returned in PP may be less than PPD.

PP An array in which to store the measure’s pick path. The contents of this array are undefined if the status returned is PNPICK. This is the two-dimensional array of
GET PICK (3P)

path elements defining the location of the picked primitive in the CSS. Each row of the array contains the structure identifier, pick id, and element number, respectively, of each element in the path. The array must be at least of dimension (3,IPPD).

**Execution**

The GET PICK function retrieves the measure of a PICK device from the current event report in the PHIGS state list. If NO PICK is returned as the status, pick path is undefined. If OK is returned as the status, pick path contains the portion of the measure's pick path requested, as specified by maximum depth to return. The path will be in either top-first or bottom-first order depending upon the value specified when the device was initialized.

When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a PICK event, GET PICK is used to retrieve the measure from the current event report.

See INITIALIZE PICK 3 for a description of the available PICK devices and how their measure values are determined.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>259</td>
<td>Ignoring function, the input device class of the current input report does not match the class being requested</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INITIALIZE PICK (3P)
- INITIALIZE PICK 3 (3P)
- AWAIT EVENT (3P)

modified 2 April 1993
NAME
GET STRING—retrieve the STRING measure from the PHIGS current event report

SYNOPSIS
C Syntax
void pget_string ( string )
char *string; /*OUT pointer to string of buffer size for the device + 1 for the terminator

FORTRAN Syntax
SUBROUTINE pgtst ( LOSTR, STR, STR_LEN )
INTEGER LOSTR /*OUT number of characters returned
CHARACTER*80 STR /*OUT string
INTEGER STR_LEN

FORTRAN Subset Syntax
SUBROUTINE pgtst ( LOSTR, STR )
INTEGER LOSTR /*OUT number of characters returned
CHARACTER*80 STR /*OUT string
INTEGER STR_LEN

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use GET STRING to retrieve the measure (logical input value) of the STRING device from the current event report in the PHIGS state list.
A STRING device measure consists of a character string.

C Output Parameter
string PHIGS copies the event’s contents to the array pointed to by string. The string is null terminated. The array must be at least as large as the buffer of the STRING device that generated the event. This buffer size is set when the device is initialized.

FORTRAN Output Parameters
LOSTR The number of characters returned in STR.
STR An array in which to store the STRING measure. The array must be at least as large as the buffer of the STRING device that generated the event. This buffer size is set when the device is initialized.
The FORTRAN subset version of this function will return no more than 80 of the characters in the measure.
STR_LEN String length.

Execution
The GET STRING function retrieves the measure of a STRING device from the current event report in the PHIGS state list.

modified 2 April 1993
When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. **AWAIT EVENT** moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a STRING event, **GET STRING** is used to retrieve the measure from the current event report.

See **INITIALIZE STRING 3** for a description of the available STRING devices and how their measure values are determined.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>259</td>
<td>Ignoring function, the input device class of the current input report does not match the class being requested</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INITIALIZE STRING (3P)
- INITIALIZE STRING 3 (3P)
- AWAIT EVENT (3P)
NAME
GET STROKE – retrieve the 2D STROKE measure from the PHIGS current event report

SYNOPSIS
C Syntax

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

FORTRAN Syntax

```
SUBROUTINE pgtsk ( N, VIEWI, NP, PXA, PYA )
INTEGER N                             maximum number of points
INTEGER VIEWI                          OUT view index
INTEGER NP                             OUT number of points
REAL PXA(N), PYA(N)                    OUT points in stroke in world coordinates
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use GET STROKE to retrieve the 2D components of the measure (logical input value) of the
STROKE device from the current event report in the PHIGS state list.

A STROKE device measure consists of a list of World Coordinate (WC) points and a
view_index. The points correspond to positions on the workstation selected by the
operator. The view_index is the index of the view used to transform these positions from
Device Coordinates (DC) to WC.

C Output Parameters

```c
view_ind
```
PHIGS sets the variable pointed to by view_ind to the index of the view
representation used to transform the NPC stroke positions to WC points.

```c
stroke
```
PHIGS copies the number and list of stroke points selected by the operator to the
variable pointed to by stroke. Ppoint_list is defined in phigs.h as follows:

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

num_points is the number of points in points.
points is the array of 2D WC points.

Note: This array must be allocated by the calling program and the array pointer
assigned to this field before calling this function.

The array must be at least as large as the buffer of the STROKE device that
generated the event. This buffer size is set when the device is initialized. Ppoint
is defined in phigs.h as follows:
typedef struct {
    Pfloat x; /* x coordinate */
    Pfloat y; /* y coordinate */
} Ppoint;

### FORTRAN Input Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>The maximum number of points to store in ( PXA ) and ( PYA ).</td>
</tr>
</tbody>
</table>

### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( VIEWI )</td>
<td>The view index used to transform the DC positions to WC points.</td>
</tr>
<tr>
<td>( NP )</td>
<td>The number of points in the measure.</td>
</tr>
<tr>
<td>( PXA, PYA )</td>
<td>The arrays in which to store the points in WC. The arrays must be at least as large as ( NP ).</td>
</tr>
</tbody>
</table>

### Execution

GET STROKE retrieves the 2D components of the STROKE measure from the current event report in the PHIGS state list. The \( x \) and \( y \) components are returned. The \( z \) component is discarded.

When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. **Await Event** moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a STROKE event, **GET STROKE** is used to retrieve the measure from the current event report.

See **INITIALIZE STROKE 3** for a description of the available STROKE devices and how their measure values are determined.

### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>259</td>
<td>Ignoring function, the input device class of the current input report does not match the class being requested</td>
</tr>
</tbody>
</table>

### SEE ALSO

- **INITIALIZE STROKE (3P)**
- **AWAIT EVENT (3P)**
- **SET VIEW TRANSFORMATION INPUT PRIORITY (3P)**
- **GET STROKE 3 (3P)**
NAME
GET STROKE 3 – retrieve the STROKE measure from the PHIGS current event report

SYNOPSIS
C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pgtsk3 ( N, VIEWI, NP, PXA, PYA, PZA )
INTEGER N maximum number of points
INTEGER VIEWI OUT view index
INTEGER NP OUT number of points
REAL PXA(N), PYA(N), PZA(N) OUT points in stroke in world coordinates
```

Required PHIGS Operating States

( PHOP, WSOP, *, * )

DESCRIPTION
Purpose
Use GET STROKE 3 to retrieve the measure (logical input value) of the STROKE device from the current event report in the PHIGS state list.

A STROKE device measure consists of a list of World Coordinate (WC) points and a view index. The points correspond to positions on the workstation selected by the operator. The view index is the index of the view used to transform these positions from Device Coordinates (DC) to WC.

C Output Parameters

- **view_ind**
  PHIGS sets the variable pointed to by *view_ind* to the index of the view representation used to transform the NPC stroke positions to WC points.

- **stroke**
  PHIGS copies the number and list of stroke points selected by the operator to the variable pointed to by *stroke*. `Ppoint_list3` is defined in `phigs.h` as follows:

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

`num_points` is the number of points in `points`.

`points` is the array of Ppoint3 structures specifying the points in WC.

**Note:** This array must be allocated by the calling program and the array pointer assigned to this field before calling this function.

The array must be at least as large as the buffer of the STROKE device that generated the event. This buffer size is set when the device is initialized. `Ppoint3` is defined in `phigs.h` as follows:

modified 2 April 1993
typedef struct {
    Pfloat x; /* x coordinate */
    Pfloat y; /* y coordinate */
    Pfloat z; /* z coordinate */
} Ppoint3;

FORTRAN Input Parameter

$N$ The maximum number of points to store in $PXA$, $PYA$, and $PZA$.

FORTRAN Output Parameters

$VIEWI$ The view index used to transform the DC positions to WC points.

$NP$ The number of points in the measure.

$PXA$, $PYA$, $PZA$ The arrays in which to store the points in WC. The arrays must be at least as large as $NP$.

Execution

GET STROKE 3 retrieves the STROKE measure from the current event report in the PHIGS state list.

When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a STROKE event, GET STROKE 3 is used to retrieve the measure from the current event report.

See INITIALIZE STROKE 3 for a description of the available STROKE devices and how their measure values are determined.

ERRORS

003 Ignoring function, function requires state ($PHOP$, $WSOP$, $*$, $*$)

259 Ignoring function, the input device class of the current input report does not match the class being requested

SEE ALSO

INITIALIZE STROKE 3 (3P)
AWAIT EVENT (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
GET STROKE (3P)
NAME
GET VALUATOR – retrieve the VALUATOR measure from the PHIGS current event report

SYNOPSIS
C Syntax

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

FORTRAN Syntax
SUBROUTINE pgtvl ( VAL )
REAL  VAL  OUT value

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use GET VALUATOR to retrieve the measure (logical input value) of the VALUATOR device from the current event report in the PHIGS state list.
A VALUATOR device measure consists of a floating point number.

C Output Parameter
valuator
PHIGS copies the event’s contents to the variable pointed to by valuator.

FORTRAN Output Parameter
VAL  The VALUATOR measure.

Execution
The GET VALUATOR function retrieves the measure of a VALUATOR device from the current event report in the PHIGS state list.
When an input device that is set to EVENT mode is triggered by the operator, an event report is added to the input event queue. The event report contains the identity and current measure of the device. AWAIT EVENT moves the measure of the oldest event in the input queue to the current event report in the PHIGS state list. If the event was a VALUATOR event, GET VALUATOR is used to retrieve the measure from the current event report.
See INITIALIZE VALUATOR 3 for a description of the available VALUATOR devices and how their measure values are determined.

ERRORS
003  Ignoring function, function requires state (PHOP, WSOP, *, *)
259  Ignoring function, the input device class of the current input report does not match the class being requested

SEE ALSO
INITIALIZE VALUATOR (3P)
INITIALIZE VALUATOR 3 (3P)
AWAIT EVENT (3P)

modified 2 April 1993
NAME
INCREMENTAL SPATIAL SEARCH – search the structure network for the next occurrence of a graphical output structure element meeting the specified search criteria.

SYNOPSIS
C Syntax

```c
void pincr_spa_search ( ref, dist, sp, mclip_flag, ceil, norm, inv, len, st, error_ind, fp, tot_len )
```

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

FORTRAN Syntax

```fortran
SUBROUTINE pss ( SRPX, SRPY, SDIST, SPTHSZ, SPATH, MCLIPF, SRCHCI,
                 NFLN, NFLISX, NFLIS, NFLESX, NFLES, IFLN, IFLISX, IFLIS, IFLESX,
                 IFLES, IPTHSZ, ERRIND, FPTHSZ, FPATH )
```

- `REAL SRPX, SRPY`  search reference point (WC)
- `REAL SDIST`  search distance
- `INTEGER SPTHSZ`  number of elements in starting path
- `INTEGER SPATH(2, SPTHSZ)`  starting path
- `INTEGER MCLIPF`  modelling clip flag
- `INTEGER SRCHCI`  search ceiling index
- `INTEGER NFLN`  number of normal filters
- `INTEGER NFLISX(NFLN)`  array of end indices of normal filter inclusion sets
- `INTEGER NFLIS(*)`  normal filter inclusion sets
- `INTEGER NFLESX(NFLN)`  array of end indices of normal filter exclusion sets
- `INTEGER NFLES(*)`  normal filter exclusion sets
- `INTEGER IFLN`  number of inverted filters
- `INTEGER IFLISX(IFLN)`  array of end indices of inverted filter inclusion sets
- `INTEGER IFLIS(*)`  inverted filter inclusion sets
- `INTEGER IFLESX(IFLN)`  array of end indices of inverted filter exclusion sets
- `INTEGER IFLES(*)`  inverted filter exclusion sets
- `INTEGER IPTHSZ`  size of found path array

modified 2 April 1993
### INCREMENTAL SPATIAL SEARCH (3P)

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

INCREMENTAL SPATIAL SEARCH searches a structure network for the next occurrence of a graphical output structure element that satisfies the specified search criteria.

**C Input Parameters**

- **ref**
  A pointer to a `Ppoint` structure that specifies the `x` and `y` coordinates, in World Coordinates (WC), of the search reference point. The `z` coordinate is assumed to be zero. The `Ppoint` structure is defined in `phigs.h` as:

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

- **dist**
  A real value specifying the maximum distance that a selected primitive may be from the search reference point.

- **sp**
  A pointer to a `Pelem_ref_list` structure that contains the starting search path. `Pelem_ref_list` is defined in `phigs.h` as:

  ```c
  typedef struct {
    Pint numElemRefs; /* number of execute references */
    Pelem_ref *elemRefs; /* list of execute references */
  } Pelem_ref_list;
  ```

  `elem_refs` is a pointer to a list of `numElemRefs` long of `Pelem_ref` structures containing the structure identifier and element number of each execute reference structure element in the execute reference list. `Pelem_ref` is defined in `phigs.h` as:

  ```c
  typedef struct {
    Pint structId; /* structure identifier */
    Pint elemPos; /* element number */
  } Pelem_ref;
  ```

- **mclip_flag**
  Indicates whether model clipping should be performed during the incremental spatial search. `Pclip_ind` is defined in `phigs.h` as follows:

  ```c
  typedef enum {
    PIND_NO_CLIP, /* Do not perform model clipping */
    PIND_CLIP /* Perform model clipping */
  } Pclip_ind;
  ```

**modified 2 April 1993 155**
The search ceiling index defines a position in the starting path list. The structure identifier at this position of the list defines a ceiling for the search.

A pointer to a Pfilter_list structure containing a set of normal filter lists. Pfilter_list is defined below.

A pointer to a Pfilter_list structure containing a set of inverted filter lists. Pfilter_list is defined in phigs.h as:

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

`filters` is a pointer to an array (num_filters) of Pfilter structures, that contains an inclusion set and an exclusion set of names.

Pfilter is defined in phigs.h as:

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

The incl_set contains a set of names for the inclusion set. The excl_set contains a set of names for the exclusion set. Pint_list is defined in phigs.h as:

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

The maximum length of the portion of the found path to be returned.

The starting position of the portion of the found path to be returned. The complete found path is returned if st is set to zero.

**C Output Parameters**

- **error_ind**
  
  A pointer to the location to store the error number of any error detected by this function.

- **fp**
  
  PHIGS stores the found path in the variable pointed to by this parameter. Pelem_ref_list is defined in phigs.h as:

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

  `num_elem_refs` specifies the length of the structure path returned.
elem_refs is a pointer to an array of Pelem_ref structures num_elem_refs long. All but the last of these structures contain the structure identifier and element number of each execute reference structure element in the path to the selected output primitive.

The application must allocate memory for len elements in the list of elem_refs. If the length of the found path is greater than the len parameter, then no found path is returned. In this case, tot_len will be set to indicate the total length of the found path.

tot_len A pointer to an integer to contain the total length of the found path.

FORTRAN Input Parameters
- SRPX The x coordinate in WC for the search reference point.
- SRPY The y coordinate in WC for the search reference point.
- SDIST The search distance.
- SPATHSZ The number of elements in the starting path.
- SPATH An array of integers containing the structure identifier and element position in the starting path.
  \[ \text{SPATH}(1, *) = \text{Structure Identifier} \]
  \[ \text{SPATH}(2, *) = \text{Element Position} \]
- MCLIPF Indicates whether model clipping should be performed during the incremental spatial search. This should be one of:
  - PNCLIP Do not perform model clipping
  - PCLIP Perform model clipping
- SRCHCI The search ceiling index defines a position in the list of the starting path. The structure identifier at this position of the list defines a ceiling for the search.
- NFLN The number of normal filters.
- NFLISX An array of integers containing the end indices into the NFLIS array for each inclusion set in the normal filter.
- NFLIS An array of integers containing the names for the inclusion sets in the normal filter.
- NFLESX An array of integers containing the end indices into the NFLES array for each exclusion set in the normal filter.
- NFLES An array of integers containing the names for the exclusion sets in the normal filter.
- IFLN The number of inverted filters.

modified 2 April 1993
INCREMENTAL SPATIAL SEARCH (3P) SunPHIGS Release 3.0

IFLISX An array of integers containing the end indices into the IFLIS array for each inclusion set in the inverted filter.

IFLIS An array of integers containing the names for the inclusion sets in the inverted filter.

IFLESX An array of integers containing the end indices into the IFLES array for each exclusion set in the inverted filter.

IFLES An array of integers containing the names for the exclusion sets in the inverted filter.

IFTHSZ Size of the FPATH array in which the returned structure path data will be stored. If this value is smaller than the actual size of the structure path (FPTHSZ), then no data will be returned in the FPATH array, but FPTHSZ will be set to indicate the array size required. If this function is called with an array size of zero, FPTHSZ is returned with the required array size. Error 2001 is returned if IPTHSZ is too small, but not if it is zero.

FORTRAN Output Parameters

ERRIND The error number of any error detected by this function.

FPTHSZ The number of structure path elements returned in FPATH.

FPATH A 2 x IPTHSZ integer array containing the found path, where the (1,*) components contain the structure identifiers, and the (2,*) components contain the element sequence numbers.

FPATH(1,∗) = Structure Identifier
FPATH(2,∗) = Element Position

Execution

When INCREMENTAL SPATIAL SEARCH is called, the search begins at the element following the position specified by the starting path. The search is conceptually a traversal with structure elements being examined sequentially and matched against the search criteria. Element position zero is permitted so that the search may start at the first element of the structure. Search filters are applied to control which output primitives structure elements in the structures searched are considered. The filters are organized into two lists, the normal and the inverted filter lists, which operate in the opposite senses. A structure element is said to be accepted if it is declared eligible when NAME SET is applied to the filters. To be accepted by a filter, NAME SET must have at least one name in common with the inclusion set and no names in common with the exclusion set. For a structure element to be considered a candidate for the spatial search, it must be accepted by each of the filters in the normal filter list and rejected by each of the filters in the inverted filter list. If the normal filter list is empty, then all structure elements satisfy the acceptance criteria for normal filters. If the inverted filter list is empty, all structure elements satisfy the rejection criteria for inverted filters. Graphical output structure elements, which satisfy the search filters are checked for proximity to the reference point.

modified 2 April 1993
For **TEXT** elements, the spatial extent is the enclosing rectangle calculated using the values of the geometric attributes (character height, character up vector, text path, and text alignment) arising from traversal, together with the following values for the workstation-dependent attributes: text font, 1; text precision, \textit{STROKE}; character expansion factor, 1; and character spacing, 0. For **ANNOTATION TEXT RELATIVE** elements, the only proximity relationship used is closeness to the annotation reference point.

For **NON-UNIFORM B-SPLINE CURVE** and **NON-UNIFORM B-SPLINE SURFACE**, **INCREMENTAL SPATIAL SEARCH** uses the following values as the workstation-dependent attributes: curve approximation type, \textit{WORKSTATION_DEPENDENT}; curve approximation value, 1; surface approximation type, \textit{WORKSTATION_DEPENDENT}; surface approximation value for \textit{u} dimension, 1; and surface approximation value for \textit{v} dimension, 1.

Search will continue until either a graphical output structure element matches the search criteria or the end of the structure identified by the search ceiling is reached. If a search is successful, the complete search path is returned as a found path. An unsuccessful search returns a null found path.

The function is incremental in that, having found a match, the search may be continued by invoking **INCREMENTAL SPATIAL SEARCH** again, with the found path as the next starting path. This allows all elements matching the search criteria to be found for a given portion of a structure network.

**ERRORS**

002 Ignoring function, function requires state (\textit{PHOP, *, *, *})

203 Ignoring function, specified starting path not found in CSS

204 Ignoring function, specified search ceiling index out of range

**SEE ALSO**

\texttt{ADD NAMES TO SET (3P)}

\texttt{REMOVE NAMES FROM SET (3P)}

\texttt{INCREMENTAL SPATIAL SEARCH 3 (3P)}
NAME

INCREMENTAL SPATIAL SEARCH 3 – search the structure network for the next occurrence of a graphical output structure element meeting the specified search criteria

SYNOPSIS

C Syntax

```c
void pincr_spa_search3 ( ref, dist, sp, mclip_flag, ceil, norm, inv, len, st, error_ind, fp, 
                          tot_len )
```

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

FORTRAN Syntax

```fortran
SUBROUTINE piss3 ( SRPX, SRPY, SRPZ, SDIST, SPTHSZ, SPATH, MCLIPF, 
                    SRCHCI, NFLN, NFLISX, NFLIS, NFLESX, NFLES, IFLN, IFLISX, IFLIS, 
                    IFLESX, IFLES, IPTHSZ, ERRIND, FPTHSZ, FPATH )
```

- `REAL SRPX, SRPY, SRPZ` search reference point (WC)
- `REAL SDIST` search distance
- `INTEGER SPTHSZ` number of elements in starting path
- `INTEGER SPATH(2, SPTHSZ)` starting path
- `INTEGER MCLIPF` modelling clip flag
- `INTEGER SRCHCI` search ceiling index
- `INTEGER NFLN` number of normal filters
- `INTEGER NFLISX(NFLN)` array of end indices of normal filter inclusion sets
- `INTEGER NFLIS(*)` normal filter inclusion sets
- `INTEGER NFLESX(NFLN)` array of end indices of normal filter exclusion sets
- `INTEGER NFLES(*)` normal filter exclusion sets
- `INTEGER IFLN` number of inverted filters
- `INTEGER IFLISX(IFLN)` array of end indices of inverted filter inclusion sets
- `INTEGER IFLIS(*)` inverted filter inclusion sets
- `INTEGER IFLESX(IFLN)` array of end indices of inverted filter exclusion sets
- `INTEGER IFLES(*)` inverted filter exclusion sets
- `INTEGER IPTHSZ` size of found path array

160 modified 2 April 1993
INTEGRAL SPATIAL SEARCH 3 searches a structure network for the next occurrence of a graphical output structure element that satisfies the specified search criteria.

**Purpose**

**C Input Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref</td>
<td>A pointer to a Ppoint structure that specifies the x, y, and z coordinates, in World Coordinates (WC), of the search reference point. The Ppoint3 structure is defined in phigs.h as:</td>
</tr>
<tr>
<td>dist</td>
<td>A real value specifying the maximum distance that a selected primitive may be from the search reference point.</td>
</tr>
<tr>
<td>sp</td>
<td>A pointer to a Pelem_ref_list structure that contains the starting search path.</td>
</tr>
<tr>
<td>elem_refs</td>
<td>A pointer to a list of num_elem_refs long of Pelem_ref structures containing the structure identifier and element number of each execute reference structure element in the execute reference list.</td>
</tr>
<tr>
<td>mclip_flag</td>
<td>Indicates whether model clipping should be performed during the incremental spatial search.</td>
</tr>
</tbody>
</table>

**Required PHIGS Operating States**

(PHOP, *, *, *)

**modified 2 April 1993**
PIND_CLIP  Perform model clipping

} Pclip_ind;

ceil  The search ceiling index defines a position in the starting path list. The structure
identifier at this position of the list defines a ceiling for the search.

norm  A pointer to a Pfilter_list structure containing a set of normal filter lists.
Pfilter_list is defined below.

inv   A pointer to a Pfilter_list structure containing a set of inverted filter lists.
Pfilter_list is defined in phigs.h as:

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

Pfilter is defined in phigs.h as:

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

The incl_set contains a set of names for the inclusion set. The excl_set contains a
set of names for the exclusion set. Pint_list is defined in phigs.h as:

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

len   The maximum length of the portion of the found path to be returned.

st    The starting position of the portion of the found path to be returned. If st is set to
zero, then the complete found path is returned.

C Output Parameters

error_ind  A pointer to the location to store the error number of any error detected by this
function.

fp    A pointer to a Pelem_ref_list structure containing the found path. Pelem_ref_list
is defined in phigs.h as:

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

modified 2 April 1993
num_elem_refs specifies the length of the structure path returned.

elem_refs is a pointer to an array of Pelem_ref structures num_elem_refs long. All but the last of these structures contain the structure identifier and element number of each execute reference structure element in the path to the selected output primitive.

The application must allocate memory for len elements in the list of elem_refs. If the length of the found path is greater than the len parameter, then no found path is returned. In this case, tot_len will be set to indicate the total length of the found path.

tot_len A pointer to an integer that returns the total length of the found path.

FORTRAN Input Parameters

SRPX The x coordinate in WC for the search reference point.
SRPY The y coordinate in WC for the search reference point.
SRPZ The z coordinate in WC for the search reference point.
SDIST The search distance.
SPTHSZ The number of elements in the starting path.
SPATH An array of integers containing the structure identifier and element position in the starting path.

SPATH(1,∗) = Structure Identifier
SPATH(2,∗) = Element Position

MCLIPF Indicates whether model clipping should be performed during the incremental spatial search. This should be one of:

PNCLIP Do not perform model clipping
PCLIP Perform model clipping

SRCHCI The search ceiling index defines a position in the list of the starting path. The structure identifier at this position of the list defines a ceiling for the search.

NFLN The number of normal filters.

NFLISX An array of integers containing the end indices into the NFLIS array for each inclusion set in the normal filter.
NFLIS An array of integers containing the names for the inclusion sets in the normal filter.

NFLESX An array of integers containing the end indices into the NFLES array for each exclusion set in the normal filter.
NFLES An array of integers containing the names for the exclusion sets in the normal
filter.

**IFLN** The number of inverted filters.

**IFLISX** An array of integers containing the end indices into the **IFLIS** array for each inclusion set in the inverted filter.

**IFLIS** An array of integers containing the names for the inclusion sets in the inverted filter.

**IFLESX** An array of integers containing the end indices into the **IFLES** array for each exclusion set in the inverted filter.

**IFLES** An array of integers containing the names for the exclusion sets in the inverted filter.

**IPTHSZ** Size of the **FPATH** array in which the returned structure path data will be stored. If this value is smaller than the actual size of the structure path (**FPTHSZ**), no data will be returned in the **FPATH** array, but **FPTHSZ** will be set to indicate the array size required. If this function is called with an array size of zero, **FPTHSZ** is returned with the required array size. Error 2001 is returned if **IPTHSZ** is too small, but not if it is zero.

**FORTRAN Output Parameters**

**ERRIND** The error number of any error detected by this function.

**FPTHSZ** The number of structure path elements returned in **FPATH**.

**FPATH** A 2 x **IPTHSZ** integer array containing the found path, where the (1,*) components contain the structure identifiers, and the (2,*) components contain the element sequence numbers.

\[
\text{FPATH}(1,*) = \text{Structure Identifier} \\
\text{FPATH}(2,*) = \text{Element Position}
\]

**Execution**

When **INCREMENTAL SPATIAL SEARCH 3** is called, the search begins at the element following the position specified by the starting path. The search is conceptually a traversal with structure elements being examined sequentially and matched against the search criteria. Element position zero is permitted so that the search may start at the first element of the structure. Search filters are applied to control which output primitives structure elements in the structures searched are considered. The filters are organized into two lists, the normal and the inverted filter lists, which operate in the opposite senses. A structure element is said to be **accepted** if it is declared eligible when **NAME SET** is applied to the filters. To be accepted by a filter, **NAME SET** must have at least one name in common with the inclusion set and no names in common with the exclusion set. For a structure element to be considered a candidate for the spatial search, it must be accepted by each of the filters in the normal filter list and rejected by each of the filters in the inverted filter list. If the normal filter list is empty, then all structure elements satisfy the acceptance criteria for normal filters. If the inverted filter list is empty, all structure elements satisfy...
the rejection criteria for inverted filters. Graphical output structure elements that satisfy
the search filters are checked for proximity to the reference point.

For TEXT elements, the spatial extent is the enclosing rectangle calculated using the values
of the geometric attributes (character height, character up vector, text path, and text
alignment) arising from traversal together with the following values for the workstation-
dependent attributes: text font, 1; text precision, STROKE; character expansion factor, 1;
and character spacing, 0. For ANNOTATION TEXT RELATIVE elements, the only proximity
relationship used is closeness to the annotation reference point.

For NON-UNIFORM B-SPLINE CURVE and NON-UNIFORM B-SPLINE SURFACE, INCREMENTAL
SPATIAL SEARCH 3 uses the following values as the workstation-dependent attributes:
curve approximation type, WORKSTATION_DEPENDENT; curve approximation value, 1;
surface approximation type, WORKSTATION_DEPENDENT; surface approximation value for
u dimension, 1; and surface approximation value for v dimension, 1.

Search will continue until either a graphical output structure element matches the search
criteria or the end of the structure identified by the search ceiling is reached. If a search is
successful, the complete search path is returned as a found path. An unsuccessful search
returns a null found path.

The function is incremental in that, having found a match, the search may be continued
by invoking INCREMENTAL SPATIAL SEARCH 3 again with the found path as the next starting
path. This allows all elements matching the search criteria to be found for a given
portion of a structure network.

ERRORS
002 Ignoring function, function requires state (PHOP, *, *, *)
203 Ignoring function, specified starting path not found in CSS
204 Ignoring function, specified search ceiling index out of range

SEE ALSO
ADD NAMES TO SET (3P)
REMOVE NAMES FROM SET (3P)
INCREMENTAL SPATIAL SEARCH (3P)
NAME

INITIALIZE CHOICE – initialize a CHOICE input device using 2D data

SYNOPSIS

C Syntax

```c
void pinit_choice ( ws, dev, istat, init, pet, echo_area, record )
```

- `Pint ws;` workstation identifier
- `Pint dev;` choice device number
- `Pin_status istat;` initial choice status
- `Pint init;` initial choice
- `Pint pet;` prompt and echo type
- `Plimit *echo_area;` echo area pointer
- `Pchoice_data *record;` data record pointer

FORTRAN Syntax

```fortran
SUBROUTINE pinch ( WKID, CHDNR, ISTAT, ICHNR, PET, XMIN, XMAX, YMIN, YMAX, LDR, DATREC )
```

- `INTEGER WKID` workstation identifier
- `INTEGER CHDNR` choice device number
- `INTEGER ISTAT` initial status (POK, PNCHOS)
- `INTEGER ICHNR` initial choice number
- `INTEGER PET` prompt/echo type
- `REAL XMIN, XMAX, YMIN, YMAX` echo area in Device Coordinates
- `INTEGER LDR` dimension of data record array
- `CHARACTER*80 DATREC(LDR)` data record

Required PHIGS Operating States

(HPH, WSOP, *, *)

DESCRIPTION

Purpose

INITIALIZE CHOICE sets the initialization parameters of a CHOICE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

- `ws` The workstation identifier of the workstation associated with the device.
- `dev` The device number of the CHOICE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
- `istant` The CHOICE status of the initial measure. Valid values from the Pin_status enumerated type defined in phigs.h are:
  - `PIN_STATUS_OK`
  - `PIN_STATUS_NO_IN`
- `init` The initial choice number. PHIGS ignores this value if `istant` is not `PIN_STATUS_OK`. 

modified 2 April 1993
pet The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

echo_area A pointer to a Plimit structure defining the x and y components of the echo volume, in Device Coordinates. The z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min;  /* minimum x coordinate value */
    Pfloat x_max;  /* maximum x coordinate value */
    Pfloat y_min;  /* minimum y coordinate value */
    Pfloat y_max;  /* maximum y coordinate value */
} Plimit;
```

record A pointer to a Pchoice_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Pchoice_data structure for prompt/echo type 1 is pet_r1.

Pchoice_data is defined in phigs.h as:

```c
typedef struct {
    union Pchoice_pets {
        struct Pchoice_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pchoice_pet_r2 {
            Pint num_prompts; /* number of alternatives*/
            Ppr_switch *prompts; /* array of prompts*/
        } pet_r2;
        struct Pchoice_pet_r3 {
            Pint num_strings; /* number of choice strings*/
            char **strings; /* array of choice strings*/
        } pet_r3;
        struct Pchoice_pet_r4 {
            Pint num_strings; /* number of alternatives*/
            char **strings; /* array of strings*/
        } pet_r4;
        struct Pchoice_pet_r5 {
            Pint struct_id; /* struct identifier*/
            Pint num_pick_ids; /* number of alternatives*/
            Pint *pick_ids; /* array of pick identifiers*/
        } pet_r5;
    }
} Pchoice_data;
```
For some prompt/echo types the data record is not used; however, the record parameter must still be supplied.

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>WKID</th>
<th>The workstation identifier of the workstation associated with the device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHDNR</td>
<td>The device number of the CHOICE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.</td>
</tr>
<tr>
<td>ISTAT</td>
<td>The choice status of the initial measure. Valid values as defined in phigs77.h are: OK, OK, PNCHOI, No choice</td>
</tr>
<tr>
<td>ICHNR</td>
<td>The initial choice number. This value is ignored if ISTAT is PNCHOI.</td>
</tr>
<tr>
<td>PET</td>
<td>The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.</td>
</tr>
<tr>
<td>XMIN, XMAX, YMIN, YMAX</td>
<td>The x and y components of the echo volume, in Device Coordinates. The z component in the workstation state list is left unchanged.</td>
</tr>
<tr>
<td>LDR</td>
<td>The dimension of the data record array.</td>
</tr>
<tr>
<td>DATREC</td>
<td>A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.</td>
</tr>
</tbody>
</table>

**Execution**

INITIALIZE CHOICE sets the initialization parameters of a CHOICE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure will retain this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A CHOICE device measure consists of a status and a choice number. Status indicates whether or not one of the possible choice values on the device was selected. (It’s possible to trigger some CHOICE devices without selecting one of the choices.) The choice number indicates the value selected, if any.
The prompt/echo type determines the display characteristics of the device, that is, how it will be presented to the operator and respond to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below.

All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS Standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume will restrict their display to this region. Some of these devices will still recognize operator input outside the region even though they do not display there. This function only specifies the $x$ and $y$ components of the echo volume. The existing $z$ component in the workstation state list is left unchanged.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be properly specified, or this function will generate an error. The ERRORS section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a CHOICE input device can be inquired with the function INQUIRE DEFAULT CHOICE DEVICE DATA. The current state of the device can be inquired for with the function INQUIRE CHOICE DEVICE STATE.

---

**AVAILABLE DEVICES**

**Devices 1, 2, 3 – OLIT**

**Flat Exclusives Widgets**

These devices are a collection of widgets from the OLIT widget set, triggered by pressing SELECT. The set consists of a shell widget, a scrolled window widget, and a flat exclusives widget. The flat exclusives widget is the item that the operator manipulates to change the choice value.

The shell widget is a pop-up window that appears when the choice device is active. To select a choice value, position the cursor over the desired item and press SELECT. Pressing SELECT on the highlighted item generates a status value of NO CHOICE.

If an initial choice is specified, the corresponding widget item is highlighted when the pop-up window appears.

The echo volume is not used by these devices. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** 1, 3

**PET 1**

Display a SunPHIGS-defined list of choice strings in the widget items. The choice strings are the integers 1 through 10.
The data record is not used for this PET.

**PET 3**
Display an application-specified list of choice strings in the widget items. The data record contains the number and array of choice strings.

**C Data Record:**
The pet_r3 member of the Pchoice_data structure, defined in phigs.h as:
```c
struct {
    Pint num_strings; /* number of choice strings */
    char **strings; /* array of choice strings */
} pet_r3;
```

**FORTRAN Data Record:**
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:
- **IL** The number of integers = 0.
- **RL** The number of real values = 0.
- **SL** The number of choice strings.
- **LSTR** An array of integers, each entry specifying the length of the corresponding character string in **STR**.
- **STR** The array of choice strings.

The widgets used and their names, in decreasing hierarchy, are:
- choice1 → popupWindowShellWidgetClass
- scrollwin → scrolledWindowWidgetClass
- list → flatExclusivesWidgetClass

Fallback resources for choice devices are:
- *choice1*background → grey
- *choice1*list.layout Type → fixedcols
- *choice1*list.measure → 10

The fully qualified name of all widgets is
- `<appl_name>.workstation<ws_id>.choice<dev_id>.<widget_name>`

where `<appl_name>` is the application name specified in the call to OPEN XPHIGS. (This will be "phigs" if OPEN XPHIGS was not called.)

For example, phigs.workstation1.choice1.scrollwin is the name of the scrolled window widget of choice device 1 on workstation 1.

**Device 4 – Mouse Buttons**
This device consists of the mouse buttons. The choice number is the id of the button pressed, possibly modified according to the state of the **Shift** and **Control** keys. The mouse buttons are numbered 1, 2, 3 from left to right. A mouse button pressed while the Shift or
Control key is depressed is distinct from the un-shift-ed and un-control-ed button.

**Prompt/echo types supported: 1**

**PET 1** These devices have no display components, that is, no prompt and no echo.

The data record is not used for this PET.

A shifted and control-ed button is identified by encoding in the value both the button number and the fact that it is shifted or control-ed. In addition, a mouse button up-stroke value is encoded to indicate that it was generated by an up stroke. The following C macros and FORTRAN functions, defined in phigs.h and phigs77.h, respectively, can be used to determine the coded value:

**C Macros**

```
C Macros

PBUTTON_VALUE( choice_value )
returns the button number of the mouse button, ignoring the Shift or Control key state.

PBUTTON_SHIFTED( key_id )
returns the choice value that is generated for the shifted mouse button.

PBUTTON_CTRLED( key_id )
returns the choice value that is generated for the control-ed mouse button.

PBUTTON_UP( key_id )
returns the choice value that is generated for the mouse button up stroke.

PBUTTON_IS_SHIFTED( choice_value )
returns 1 if the value is a shifted value, otherwise it returns 0.

PBUTTON_IS_CTRLED( choice_value )
returns 1 if the value is a control-ed value, otherwise it returns 0.

PBUTTON_IS_UP( choice_value )
returns 1 if the value was generated by an up stroke, otherwise it returns 0.
```

**FORTRAN Functions**

These functions correspond to the C macros described above.

```
FORTRAN Functions

pbuttonvalue( choice_value : integer )
pbuttonshifted( key_id : integer )
pbuttoncrtled( key_id : integer )
pbuttonup( key_id : integer )
pbuttonishifted( choice_value : integer )
pbuttoniscrtled( choice_value : integer )
pbuttoniusup( choice_value : integer )
```

modified 2 April 1993
All mouse button choice values are in one of the following ranges:

- **un-shifted/un-control-ed:** $1$ through $\text{number of choices}$
- **shifted:** $\text{PBUTTON_SHIFTED}(1)$ through $\text{PBUTTON_SHIFTED(\text{num choices})}$
- **control-ed:** $\text{PBUTTON_CTRLED}(1)$ through $\text{PBUTTON_CTRLED(\text{num choices})}$

These ranges are not contiguous; however, the button values within each range are. The **number of choices** field in the workstation description table reflects only the number of possible buttons; the shifted and control-ed values are not included in this number.

### Device 5 – Keyboard
This device consists of the keyboard keys. The choice number is the $X$ *keysym* for the key pressed by the operator, as defined in the include file `<x11/keysymdef.h>.

**Prompt/echo types supported:** 1

PET 1  These devices have no display components; that is, no prompt and no echo.
The data record is not used for this PET.

### Device 6 – Software Button Box
This device is a software simulation of a 32-key button box, triggered by pressing SELECT. It appears as a pop-up window when the device is active (that is, in EVENT or SAMPLE mode or waiting for a REQUEST to be satisfied).
The echo volume is not used by this device. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** 1, 3

PET 1  All 32 buttons are active while the device is active; the keys are labeled with default strings from $1$ to $32$.
The data record is not used for this PET.

PET 3  The buttons specified by the application are active while the device is active; a button is active if it has a non-null or non-blank label string. The data record contains the number and array of labels. If less than 32 values are specified, the first $n$ buttons will be set as specified, and the remainder will be inactive.

**C Data Record:**
The `pet_r3` member of the Pchoice_data structure, defined in `phigs.h` as:

```c
struct {
    Pint num_strings;  /* number of choice strings */
    char **strings;   /* array of choice strings */
} pet_r3;
```
FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type's data record should be:

\[IL\] The number of integers = 0.

\[RL\] The number of real values = 0.

\[SL\] The number of choice strings.

\[LSTR\] An array of integers, each entry specifying the length of the corresponding character string in \(STR\).

\[STR\] The array of choice strings.

Device 7 – Sun Buttons

This device is the Sun 32 key lighted button box, if one is attached. The buttons trigger only when the pointer is within the display surface of the associated workstation.

The echo volume is not used by this device.

Prompt/echo types supported: 1, 2

PET 1 All 32 buttons will be active and lit while the device is active.

The data record is not used for this PET.

PET 2 The buttons specified by the application are active and lit while the device is active. The data record contains the number and array of on/off values. If less than 32 values are specified, the first \(n\) buttons will be set as specified, and the remainder will be inactive and not lit.

C Data Record:
The pet_r2 member of the Pchoice_data structure, defined in phigs.h as:

```c
struct {
    Pint num_prompts; /* number of prompts */
    Ppr_switch *prompts; /* array of prompts */
} pet_r2;
```

Ppr_switch is an enumerated type defined in phigs.h; valid values for this type are PPR_ON (active and lit) and PPR_OFF (not active and not lit).

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type's data record should be:

\[IL\] The number of on/off values.

\[IA\] An array of on/off values specifying whether each button is active and lit (PON) or not active and not lit (POFF).

\[RL\] The number of real values = 0.

\[SL\] The number of strings = 0.
ASSOCIATIONS BETWEEN INPUT DEVICES - SunPHIGS Extension

All CHOICE devices can have associated with them a set of other input devices that will also generate input events when the CHOICE device is triggered. The association is made with the ESCAPE function, described in the ESCAPE reference manual page. The ESCAPE function accepts a triggering-device/triggered-device pair. This pair indicates an additional device to trigger (the slave device) when the specified triggering device (the master device) is triggered. Separate associations can be made with each possible value of the master device (that is, choice numbers), in which case selection of that value by the operator will trigger the devices associated with that value. This allows operator selection of a specific choice value on a specified CHOICE device to also trigger one or more other input devices.

When a device and its associated devices are triggered, a set of simultaneous events are generated, one event for each device.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation is neither of category INPUT nor of category OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
251 Ignoring function, the function requires the input device to be in REQUEST mode
253 Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
254 Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
255 Ignoring function, one of the echo area/volume boundary points is outside the range of the device
260 Ignoring function, one of the fields within the input device data record is in error
261 Ignoring function, initial value is invalid

SEE ALSO

ESCAPE -19 (3P)
SET CHOICE MODE (3P)
REQUEST CHOICE (3P)
SAMPLE CHOICE (3P)
GET CHOICE (3P)
INQUIRE CHOICE DEVICE STATE (3P)
INITIALIZE CHOICE 3 (3P)
NAME

INITIALIZE CHOICE 3 – initialize a CHOICE input device using 3D data

SYNOPSIS

C Syntax

```c
void pinit_choice3 ( ws, dev, istat, init, pet, echo_volume, record )
```

- `Pint ws;`  
  - workstation identifier
- `Pint dev;`  
  - choice device number
- `Pin_status istat;`  
  - initial choice status
- `Pint init;`  
  - initial choice
- `Pint pet;`  
  - prompt and echo type
- `Plimit3 *echo_volume;`  
  - echo volume pointer
- `Pchoice_data3 *record;`  
  - data record pointer

FORTRAN Syntax

```fortran
SUBROUTINE pinch3 ( WKID, CHDNR, ISTAT, ICHNR, PET, EVOL, LDR, DATREC )
```

- `INTEGER WKID`  
  - workstation identifier
- `INTEGER CHDNR`  
  - choice device number
- `INTEGER ISTAT`  
  - initial status (POK, PNCHOI)
- `INTEGER ICHNR`  
  - initial choice number
- `INTEGER PET`  
  - prompt/echo type
- `REAL EVOL(6)`  
  - echo volume in Device Coordinates
- `INTEGER LDR`  
  - dimension of data record array
- `CHARACTER*80 DATREC(LDR)`  
  - data record

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INITIALIZE CHOICE 3 sets the initialization parameters of a CHOICE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

- `ws`  
  - The workstation identifier of the workstation associated with the device.
- `dev`  
  - The device number of the CHOICE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
- `istat`  
  - The CHOICE status of the initial measure. Valid values from the `Pin_status` enumerated type defined in `phigs.h` are:
    - `PIN_STATUS_OK`
    - `PIN_STATUS_NO_IN`
- `init`  
  - The initial choice number. PHIGS ignores this value if `istat` is not `PIN_STATUS_OK`.

modified 2 April 1993
pet  The prompt/echo type desired. Those supported by each device are listed in the
AVAILABLE DEVICES section below.

echo_volume
A pointer to a Plimit3 structure defining the x, y, and z components of the echo
volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;

record
A pointer to a Pchoice_data3 structure containing the data record information.
The contents of the data record for each device and prompt/echo type are
described below in the AVAILABLE DEVICES section.
The members of the data record union correspond to the prompt/echo type
being used. As an example, the appropriate member of the Pchoice_data3
structure for prompt/echo type 1 is pet_r1.

For some prompt/echo types the data record is not used. The record parameter,
however, must still be supplied.
Pchoice_data3 is defined in phigs.h as:
typedef struct {
    union Pchoice_pets {
        struct Pchoice_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pchoice_pet_r2 {
            Pint num_prompts; /* number of alternatives*/
            Ppr_switch *prompts; /* array of prompts*/
        } pet_r2;
        struct Pchoice_pet_r3 {
            Pint num_strings; /* number of choice strings*/
            char **strings; /* array of choice strings*/
        } pet_r3;
        struct Pchoice_pet_r4 {
            Pint num_strings; /* number of alternatives*/
            char **strings; /* array of strings*/
        } pet_r4;
        struct Pchoice_pet_r5 {
            Pint struct_id; /* struct identifier*/
            Pint num_pick_ids; /* number of alternatives*/
        } pet_r5;
    }
} Pchoice_pets;
**FORTRAN Input Parameters**

- **WKID**  The workstation identifier of the workstation associated with the device.
- **CHDNR** The device number of the CHOICE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
- **ISTAT** The choice status of the initial measure. Valid values as defined in phigs77.h are:
  - **POK**
  - **PNCHOI**
- **ICHNR** The initial choice number. This value is ignored if ISTAT is PNCHOI.
- **PET** The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.
- **EVL** The \( x, y, \) and \( z \) limits of the echo volume, \( XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX \), in Device Coordinates.
- **LDR** The dimension of the data record array.
- **DATREC** A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

**Execution**

INITIALIZE CHOICE 3 sets the initialization parameters of a CHOICE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure will retain this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A CHOICE device measure consists of a status and a choice number. Status indicates whether or not one of the possible choice values on the device was selected. (It is possible to trigger some CHOICE devices without selecting one of the choices.) The choice number indicates the value selected, if any.

The prompt/echo type determines the display characteristics of the device, that is, how it will be presented to the operator and respond to his actions. Each device supports one or more prompt/echo type. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type
1. Positive prompt/echo types are defined by the PHIGS Standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The *echo volume* defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume will restrict their display to this region. Some of these devices will still recognize operator input outside the region even though they don’t display there.

The *input data record* contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the *AVAILABLE DEVICES* section below list the data record contents that each device recognizes.

All the initialization parameters must be properly specified, or this function will generate an error. The *ERRORS* section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a CHOICE input device can be inquired with the function INQUIRE DEFAULT CHOICE DEVICE DATA 3. The current state of the device can be inquired with the function INQUIRE CHOICE DEVICE STATE 3.

**AVAILABLE DEVICES**

<table>
<thead>
<tr>
<th>Devices 1, 2, 3 – OLIT Flat Exclusives Widgets</th>
</tr>
</thead>
</table>

These devices are a collection of widgets from the OLIT widget set, triggered by pressing SELECT. The set consists of a shell widget, a scrolled window widget, and a flat exclusives widget. The flat exclusives widget is the item that the operator manipulates to change the choice value.

The shell widget is a pop-up window that appears when the choice device is active. To select a choice value, position the cursor over the desired item and press SELECT. Pressing SELECT on the highlighted item generates a status value of NO CHOICE.

If an initial choice is specified, the corresponding widget item is highlighted when the pop-up window appears.

The echo volume is not used by these devices. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** 1, 3

**PET 1** Display a SunPHIGS-defined list of choice strings in the widget items. The choice strings are the integers 1 through 10.

The data record is not used for this PET.

**PET 3** Display an application-specified list of choice strings in the widget items. The data record contains the number and array of choice strings.
C Data Record:
The pet_r3 member of the Pchoice_data structure, defined in phigs.h as:

```c
struct {
    Pint    num_strings;    /* number of choice strings */
    char**  strings;   /* array of choice strings */
} pet_r3;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- `IL` The number of integers = 0.
- `RL` The number of real values = 0.
- `SL` The number of choice strings.
- `LSTR` An array of integers, each entry specifying the length of the corresponding character string in `STR`.
- `STR` The array of choice strings.

The widgets used and their names, in decreasing hierarchy, are:

- `choice1` → `popupWindowShellWidgetClass`
- `scrollwin` → `scrolledWindowWidgetClass`
- `list` → `flatExclusivesWidgetClass`

Fallback resources for choice devices are:

- `*choice1*background` → `grey`
- `*choice1*list.layout` → `fixedcols`
- `*choice1*list.measure` → `10`

The fully qualified name of all widgets is

```plaintext
<appl_name>.workstation<ws_id>.choice<dev_id>.<widget_name>
```

where `<appl_name>` is the application name specified in the call to `OPEN XPHIGS`. (This will be "phigs" if `OPEN XPHIGS` was not called.)

For example, `phigs.workstation1.choice1.scrollwin` is the name of the scrolled window widget of choice device 1 on workstation 1.

**Device 4 – Mouse Buttons**

This device consists of the mouse buttons. The choice number is the id of the button pressed, possibly modified according to the state of the `Shift` and `Control` keys. The mouse buttons are numbered 1, 2, 3 from left to right. A mouse button pressed while the Shift or Control key is depressed is distinct from the un-shifted and un-control-ed button.

**Prompt/echo types supported:** 1

- PET 1 These devices have no display components, that is, no prompt and no echo.
The data record is not used for this PET.
A shifted and control-ed button is identified by encoding in the value both the button number and the fact that it is shifted or control-ed. In addition, a mouse button up-stroke value is encoded to indicate that it was generated by an up stroke. The following C macros and FORTRAN functions, defined in phigs.h and phigs77.h, respectively, can be used to determine the coded value:

C Macros

\texttt{PBUTTON_VALUE( choice_value )}

returns the button number of the mouse button, ignoring the Shift or Control key state.

\texttt{PBUTTON_SHIFTED( key_id )}

returns the choice value that is generated for the shifted mouse button.

\texttt{PBUTTON_CTRLED( key_id )}

returns the choice value that is generated for the control-ed mouse button.

\texttt{PBUTTON_UP( key_id )}

returns the choice value that is generated for the mouse button up stroke.

\texttt{PBUTTON_IS_SHIFTED( choice_value )}

returns 1 if the value is a shifted value, otherwise it returns 0.

\texttt{PBUTTON_IS_CTRLED( choice_value )}

returns 1 if the value is a control-ed value, otherwise it returns 0.

\texttt{PBUTTON_IS_UP( choice_value )}

returns 1 if the value was generated by an up stroke, otherwise it returns 0.

FORTRAN Functions

These functions correspond to the C macros described above.

\texttt{pbuttonvalue( choice_value : integer )}

\texttt{pbuttonshifted( key_id : integer )}

\texttt{pbuttonctrled( key_id : integer )}

\texttt{pbuttonup( key_id : integer )}

\texttt{pbuttonisshifted( choice_value : integer )}

\texttt{pbuttoniscrled( choice_value : integer )}

\texttt{pbuttonisup( choice_value : integer )}

All mouse button choice values will be in one of the following ranges:

un-shifted/un-control-ed: 1 through number of choices
shifted: \texttt{PBUTTONSHIFTED(1)} through \texttt{PBUTTONSHIFTED(num choices)}
control-ed: \texttt{PBUTTON_CTRLED(1)} through \texttt{PBUTTON_CTRLED(num choices)}
These ranges are not contiguous; however, the button values within each range are. The number of choices field in the workstation description table reflects only the number of possible buttons; the shifted and control-ed values are not included in this number.

Device 5 – Keyboard

This device consists of the keyboard keys. The choice number is the x keysym for the key pressed by the operator, as defined in the include file <x11/keysymdef.h>.

Prompt/echo types supported: 1

PET 1 These devices have no display components; that is, no prompt and no echo.
The data record is not used for this PET.

Device 6 – Software Button Box

This device is a software simulation of a 32-key button box, triggered by pressing SELECT. It appears as a pop-up window when the device is active (that is, in EVENT or SAMPLE mode or waiting for a REQUEST to be satisfied).

The echo volume is not used by this device. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

Prompt/echo types supported: 1, 3

PET 1 All 32 buttons are active while the device is active; the keys are labeled with default strings from 1 to 32.
The data record is not used for this PET.

PET 3 The buttons specified by the application are active while the device is active; a button is active if it has a non-null or non-blank label string. The data record contains the number and array of labels. If less than 32 values are specified, the first n buttons will be set as specified, and the remainder will be inactive.

C Data Record:
The pet_r3 member of the Pchoice_data structure, defined in phigs.h as:

```c
struct {
    Pint num_strings; /* number of choice strings */
    char **strings; /* array of choice strings */
} pet_r3;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of integers = 0.
- **RL** The number of real values = 0.
- **SL** The number of choice strings.
- **LSTR** An array of integers, each entry specifying the length of the corresponding character string in STR.
**Device 7 – Sun Buttons**

This device is the Sun 32 key lighted button box, if one is attached. The buttons trigger only when the pointer is within the display surface of the associated workstation.

The echo volume is not used by this device.

**Prompt/echo types supported:** 1, 2

**PET 1**
All 32 buttons are active and lit while the device is active.

The data record is not used for this PET.

**PET 2**
The buttons specified by the application are active and lit while the device is active. The data record contains the number and array of on/off values. If less than 32 values are specified, the first $n$ buttons will be set as specified, and the remainder will be inactive and not lit.

**C Data Record:**
The pet_r2 member of the Pchoice_data structure, defined in phigs.h as:

```c
struct {
    Pint num_prompts; /* number of prompts */
    Ppr_switch *prompts; /* array of prompts */
} pet_r2;
```

Ppr_switch is an enumerated type defined in phigs.h; valid values for this type are PPR_ON (active and lit) and PPR_OFF (not active and not lit).

**FORTRAN Data Record:**
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of on/off values.
- **IA** An array of on/off values specifying whether each button is active and lit (PON) or not active and not lit (POFF).
- **RL** The number of real values = 0.
- **SL** The number of strings = 0.

**ASSOCIATIONS BETWEEN INPUT DEVICES - SunPHIGS Extension**

All CHOICE devices can have associated with them a set of other input devices that will also generate input events when the CHOICE device is triggered. The association is made with the ESCAPE function, described in the ESCAPE reference manual page. The ESCAPE function accepts a triggering-device/triggered-device pair. This pair indicates an additional device to trigger (the slave device) when the specified triggering device (the master device) is triggered. Separate associations can be made with each possible value of the master device (that is, choice numbers), in which case selection of that value by the operator will trigger the devices associated with that value. This allows operator selection of a specific choice value on a specified CHOICE device to also trigger one or more other input devices.

182 modified 2 April 1993
When a device and its associated devices are triggered, a set of simultaneous events are generated, one event for each device.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation is neither of category INPUT nor of category OUTIN
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode
253  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
254  Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
255  Ignoring function, one of the echo area/volume boundary points is outside the range of the device
260  Ignoring function, one of the fields within the input device data record is in error
261  Ignoring function, initial value is invalid

**SEE ALSO**

ESCAPE -19 (3P)
SET CHOICE MODE (3P)
REQUEST CHOICE (3P)
SAMPLE CHOICE (3P)
GET CHOICE (3P)
INQUIRE CHOICE DEVICE STATE 3 (3P)
INITIALIZE CHOICE (3P)
NAME

initialize a LOCATOR input device using 2D data

SYNOPSIS

\[
\text{void pinit_loc ( ws, dev, init_view_ind, init_loc_pos, pet, echo_area, record )}
\]

\[
P\text{int ws; \hspace{2cm} workstation identifier}
\]

\[
P\text{int dev; \hspace{2cm} locator device number}
\]

\[
P\text{int init_view_ind; \hspace{2cm} initial view indicator}
\]

\[
P\text{point *init_loc_pos; \hspace{2cm} initial locator pointer}
\]

\[
P\text{int pet; \hspace{2cm} prompt and echo type}
\]

\[
P\text{limit *echo_area; \hspace{2cm} echo area pointer}
\]

\[
P\text{loc_data *record; \hspace{2cm} data record pointer}
\]

FORTRAN Syntax

\[
\text{SUBROUTINE pinlc ( WKID, LCDNR, IVIEWI, IPX, IPY, PET, XMIN, XMAX, YMIN, YMAX, LDR, DATREC )}
\]

\[
\text{INTEGER WKID \hspace{2cm} workstation identifier}
\]

\[
\text{INTEGER LCDNR \hspace{2cm} locator device number}
\]

\[
\text{INTEGER IVIEWI \hspace{2cm} initial view index}
\]

\[
\text{REAL IPX, IPY \hspace{2cm} initial locator position in World Coordinates}
\]

\[
\text{INTEGER PET \hspace{2cm} prompt/echo type}
\]

\[
\text{REAL XMIN, XMAX, YMIN, YMAX \hspace{2cm} echo area in Device Coordinates}
\]

\[
\text{INTEGER LDR \hspace{2cm} dimension of data record array}
\]

\[
\text{CHARACTER*80 DATREC(LDR) \hspace{2cm} data record}
\]

Required PHIGS Operating States

\(\text{PHOP, WSOP, *, *}\)

DESCRIPTION

Purpose

Use INITIALIZE LOCATOR to set the initialization parameters of a LOCATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

\(\text{ws} \hspace{2cm} \text{The workstation identifier of the workstation associated with the device.}\)

\(\text{dev} \hspace{2cm} \text{The device number of the LOCATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.}\)

\(\text{init_view_ind} \hspace{2cm} \text{The index of the view representation in the workstation’s view table to use to map the specified initial position from World Coordinates (WC) to Normalized Projection Coordinates (NPC).}\)

\(\text{init_loc_pos} \hspace{2cm} \text{A Ppoint structure specifying the x and y WC coordinates of the initial locator}\)
position. Ppoint is defined in phigs.h as follows:

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

pet The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

echo_area A pointer to a Plimit structure defining the x and y components of the echo volume, in Device Coordinates (DC). The z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
} Plimit;

record A pointer to a Ploc_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Ploc_data structure for prompt/echo type 1 is pet_r1.

For some prompt/echo types the locator data record is not used; however, the record parameter must still be supplied.

FORTRAN Input Parameters

WKID The workstation identifier of the workstation associated with the device.

LCDNR The device number of the LOCATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

IVIEWI The view index of the view representation in the workstation’s view table to use to map the specified initial position from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

IPX, IPY The x and y WC of the initial locator position. The z component in the workstation state list is left unchanged.

PET The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

XMIN, XMAX, YMIN, YMAX The x and y components of the echo volume, in Device Coordinates (DC). The z
component in the workstation state list is left unchanged.

*LDR*  The dimension of the data record array.

*DATREC*  A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

**Execution**

INITIALIZE LOCATOR sets the initialization parameters of a LOCATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value to which the device will be set whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A LOCATOR device measure consists of a position and a view index. Position is the World Coordinate (WC) point corresponding to the position on the workstation selected by the operator. The view index is the index of the view representation used to transform the locator position from Normalized Projection Coordinates (NPC) to WC. This view representation is determined by selecting the highest priority representation that contains the locator position within its NPC limits. See SET VIEW TRANSFORMATION INPUT PRIORITY for more information. The workstation transform is used to transform the operator-selected position from Device Coordinates (DC) to NPC.

The initial locator position is transformed to DC by applying the view orientation and view mapping transforms of the specified view representation, then applying the workstation transformation. If the view index is invalid, an error is generated.

The prompt/echo type determines the display characteristics of the device—that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo type. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS Standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in DC. Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they don’t display there. This function only specifies the x and y components of the echo volume. The existing z component in the workstation state list is left unchanged.
The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be specified properly, or this function generates an error. The ERRORS section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a LOCATOR input device can be inquired for with the function INQUIRE DEFAULT LOCATOR DEVICE DATA. The current state of the device can be inquired for with the function INQUIRE LOCATOR DEVICE STATE.

### AVAILABLE DEVICES

**Devices 1 - 5**

Device 1: Cursor and Left Mouse Button  
Device 2: Cursor and Middle Mouse Button  
Device 3: Cursor and Right Mouse Button  
Device 4: Mouse Movement  
Device 5: Mouse Movement While Any Mouse Button Down (Drag)

These devices are associated with the X pointer device, usually the mouse and cursor. Devices 1, 2, and 3 are triggered by the left, middle, and right mouse button, respectively. Devices 4 and 5 are triggered by mouse movement. The locator position in the LOCATOR device’s measure is the WC position corresponding to the pointer event position. PHIGS computes the WC position from the two-dimensional pointer position by transforming the pointer position to a 2D NPC point, determining the highest priority view containing that point, setting the Z coordinate of the NPC point to the lower Z limit of that view, and transforming the NPC point to World Coordinates. This function returns only the X and Y coordinates of the resulting position.

When in EVENT mode, devices 4 and 5 can easily generate hundreds of events in a few seconds and quickly fill the input queue if it is not monitored continuously and the events removed immediately. These devices do not provide acknowledgement (cursor blink) to the operator when they place an event on the input queue, as the other devices do.

Not all prompt/echo types use the initial locator position. Only those that need an additional point to the current pointer position use the initial position, PETS −4 and −5, for instance. No prompt/echo type moves the pointer to the initial position when the device is enabled. The operator is in complete control of the pointer position.

The LOCATOR echo is removed from the workstation when the cursor leaves the echo area.

**Prompt/echo types supported:** 1, 2, 3, −2 (coloured crosshairs), −4 (rubber band line), −5 (rubber band rectangle)

PET 1  
Display the default cursor at the current locator position.  
The data record is not used for this PET.
PET 2  Display crosshairs that intersect at the current locator position and extend to the edges of the workstation’s display surface. The crosshair colour is that of colour index 1 in the workstation’s colour table. (PET –2 also uses crosshairs and allows their colour to be specified.)

The data record is not used for this PET.

PET 3  Display a cross cursor at the current locator position.

The data record is not used for this PET.

PET –2  Display crosshairs that intersect at the current locator position and extend to the edges of the workstation’s display surface. The crosshair colour index is specified in the data record.

C Data Record:
The pet_u2 member of the Ploc_data structure, defined in phigs.h as:

```c
struct {
    Pint    crosshair_colr; /* colour index */
} pet_u2;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

| IL  | The number of integers = 1. |
| IA  | Contains one integer value in position IA(1) specifying the colour of the crosshairs. |
| RL  | The number of real values = 0. |
| SL  | The number of strings = 0. |

PET –4  Display a rubber banding line connecting the initial locator position to the current locator position. The line attributes are specified in the data record.

C Data Record:
The pet_u4 member of the Ploc_data structure, defined in phigs.h as:

```c
struct {
    Pline_bundle line_bundle; /* line type, width, and colour index */
} pet_u4;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

| IL  | The number of integers = 2. |
| IA  | An array of integers with the following values: |
| IA(1) | The line type of the rubber band line. |
IA(2)  The colour of the rubber band line.

RL   The number of real values = 1.

RA   Contains one real value in position RA(1) specifying the line width scale factor of the rubber band line.

SL   The number of strings = 0.

PET −5 Display rubber banding edges of a rectangle, the diagonal of which connects the initial locator position to the current locator position. The line attributes of the edges are specified in the data record.

C Data Record:
The pet_u5 member of the Ploc_data structure, defined in phigs.h as:

```c
struct {
    Pline_bundle line_bundle; /* line type, width and colour index */
} pet_u5;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL   The number of integers = 2.

IA   An array of integers with the following values:

IA(1)  The line type of the rubber band rectangle.

IA(2)  The colour of the rubber band rectangle.

RL   The number of real values = 1.

RA   Contains one real value in position RA(1) specifying the line width scale factor of the rubber band rectangle.

SL   The number of strings = 0.

ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation category is not INPUT or OUTIN
114  Ignoring function, the view index value is less than zero
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode
253  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place

modified 2 April 1993
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>254</td>
<td>Ignoring function, invalid echo area/volume; $X_{MIN} \geq X_{MAX}$, $Y_{MIN} \geq Y_{MAX}$, or $Z_{MIN} &gt; Z_{MAX}$</td>
</tr>
<tr>
<td>255</td>
<td>Ignoring function, one of the echo area/volume boundary points is outside the range of the device</td>
</tr>
<tr>
<td>260</td>
<td>Ignoring function, one of the fields within the input device data record is in error</td>
</tr>
<tr>
<td>261</td>
<td>Ignoring function, initial value is invalid</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- ESCAPE -19 (3P)
- SET LOCATOR MODE (3P)
- REQUEST LOCATOR (3P)
- SAMPLE LOCATOR (3P)
- GET LOCATOR (3P)
- INQUIRE LOCATOR DEVICE STATE (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
- INITIALIZE LOCATOR 3 (3P)
NAME

INITIALIZE LOCATOR 3 – initialize a LOCATOR input device using 3D data

SYNOPSIS

C Syntax

```c
void pinit_loc3 ( ws, dev, init_view_ind, init_loc_pos, pet, echo_volume, record )
```

Pint ws;                 // workstation identifer
Pint dev;                // locator device number
Pint init_view_ind;      // initial view indicator
Ppoint3 *init_loc_pos;   // initial locator position
Pint pet;                // prompt and echo type
Plimit3 *echo_volume;    // echo volume pointer
Ploc_data3 *record;      // data record pointer

FORTRAN Syntax

```fortran
SUBROUTINE pinc3 ( WKID, LCDNR, IVIEWI, IPX, IPY, IPZ, PET, EVOL, LDR, DATREC )
```

INTEGER WKID             // workstation identifier
INTEGER LCDNR            // locator device number
INTEGER IVIEWI           // initial view index
REAL IPX, IPY, IPZ       // initial locator position in World Coordinates
INTEGER PET              // prompt/echo type
REAL EVOL(6)             // echo volume in Device Coordinates
INTEGER LDR              // dimension of data record array
CHARACTER*80 DATREC(LDR)  // data record

Required PHIGS Operating States

( PHOP, WSOP, *, * )

DESCRIPTION

Purpose

Use INITIALIZE LOCATOR 3 to set the initialization parameters of a LOCATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

`ws` The workstation identifier of the workstation associated with the device.

`dev` The device number of the LOCATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

`init_view_ind` The index of the view representation in the workstation’s view table to use to map the specified initial position from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

`init_loc_pos` A Ppoint3 structure specifying the initial locator position, in World Coordinates.

modified 2 April 1993
Ppoint3 is defined in phigs.h as follows:

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

`pet` The prompt/echo type desired. Those supported by each device are listed in the `AVAILABLE DEVICES` section below.

`echo_volume` A pointer to a Plimit3 structure specifying the echo volume, in Device Coordinates (DC). Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min;    /* minimum x coordinate value */
    Pfloat x_max;    /* maximum x coordinate value */
    Pfloat y_min;    /* minimum y coordinate value */
    Pfloat y_max;    /* maximum y coordinate value */
    Pfloat z_min;    /* minimum z coordinate value */
    Pfloat z_max;    /* maximum z coordinate value */
} Plimit3;
```

`record` A pointer to a Ploc_data3 structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the `AVAILABLE DEVICES` section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Ploc_data3 structure for prompt/echo type 1 is `pet_r1`.

For some prompt/echo types the locator data record is not used; however, the `record` parameter must still be supplied.

### FORTRAN Input Parameters

**WKID** The workstation identifier of the workstation associated with the device.

**LCDNR** The device number of the LOCATOR device to initialize. See the `AVAILABLE DEVICES` section below for a description of the available devices.

**IVIEWI** The view index of the view representation in the workstation’s view table to use to map the specified initial position from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

**IPX, IPY, IPZ** The x, y, and z WC of the initial locator position.

**PET** The prompt/echo type desired. Those supported by each device are listed in the `AVAILABLE DEVICES` section below.

**EVOL** The x, y, and z limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, in
Device Coordinates (DC).

**LDR** The dimension of the data record array.

**DATREC** A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

**Execution**

INITIALIZE LOCATOR 3 sets the initialization parameters of a LOCATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value to which the device is set whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A LOCATOR device measure consists of a position and a view index. Position is the World Coordinate (WC) point corresponding to the position on the workstation selected by the operator. The view index is the index of the view representation used to transform the locator position from Normalized Projection Coordinates (NPC) to WC. This view representation is determined by selecting the highest priority representation that contains the locator position within its NPC limits. See SET VIEW TRANSFORMATION INPUT PRIORITY for more information. The workstation transform is used to transform the operator-selected position from Device Coordinates (DC) to NPC.

The initial locator position is transformed to DC by applying the view orientation and view mapping transforms of the specified view representation, then applying the workstation transformation. If the view index is invalid, an error is generated.

The prompt/echo type determines the display characteristics of the device—that is, how it will be presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS Standard. Negative types are implementation-dependent. Most Sun PHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in DC. Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they don’t display there.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data...
record contents that each device recognizes. All the initialization parameters must be specified properly, or this function generates an error. The \texttt{ERRORS} section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a \texttt{LOCATOR} input device can be inquired with the function \texttt{INQUIRE DEFAULT LOCATOR DEVICE DATA 3}. The current state of the device can be inquired with the function \texttt{INQUIRE LOCATOR DEVICE STATE 3}.

\subsection*{AVAILABLE DEVICES}

\textbf{Devices 1 - 5}

- Device 1: Cursor and Left Mouse Button
- Device 2: Cursor and Middle Mouse Button
- Device 3: Cursor and Right Mouse Button
- Device 4: Mouse Movement
- Device 5: Mouse Movement While Any Mouse Button Down (Drag)

These devices are associated with the $x$ pointer device, usually the mouse and cursor. Devices 1, 2, and 3 are triggered by the left, middle, and right mouse button, respectively. Devices 4 and 5 are triggered by mouse movement. The locator position in the \texttt{LOCATOR} device’s measure is the WC position corresponding to the pointer event position. PHIGS computes the WC position from the two-dimensional pointer position by transforming the pointer position to a 2D NPC point, determining the highest priority view containing that point, setting the $z$ coordinate of the NPC point to the lower $z$ limit of that view, and transforming the NPC point to World Coordinates.

When in \texttt{EVENT} mode, devices 4 and 5 can easily generate hundreds of events in a few seconds and quickly fill the input queue if it is not monitored continuously and the events removed immediately. These devices do not provide acknowledgement (cursor blink) to the operator when they place an event on the input queue, as the other devices do.

Not all prompt/echo types use the initial locator position. Only those that need an additional point to the current pointer position use the initial position, \texttt{PETs -4 and -5} for instance. No prompt/echo type moves the pointer to the initial position when the device is enabled. The operator is in complete control of the pointer position.

The \texttt{LOCATOR} echo is removed from the workstation when the cursor leaves the echo volume. Only the $x$ and $y$ components of the echo volume are used. The $z$ component is ignored.

\textbf{Prompt/echo types supported:} 1, 2, 3, –2 (coloured crosshairs), –4 (rubber band line), –5 (rubber band rectangle)

\begin{itemize}
\item \textbf{PET 1} Display the default cursor at the current locator position.
  The data record is not used for this PET.
\item \textbf{PET 2} Display crosshairs that intersect at the current locator position and extend to the edges of the workstation’s display surface. The crosshair colour is that of colour
\end{itemize}
index 1 in the workstation’s colour table. (PET−2 also uses crosshairs and allows their colour to be specified.)

The data record is not used for this PET.

PET 3 Display a cross cursor at the current locator position.

The data record is not used for this PET.

PET −2 Display crosshairs that intersect at the current locator position and extend to the edges of the workstation’s display surface. The crosshair colour index is specified in the data record.

C Data Record:
The pet_u2 member of the Ploc_data3 structure, defined in phigs.h as:

```c
struct {
    Pint  crosshair_colr; /∗ colour index ∗/
} pet_u2;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 1.

IA Contains one integer value in position IA(1) specifying the colour of the crosshairs.

RL The number of real values = 0.

SL The number of strings = 0.

PET −4 Display a rubber banding line connecting the initial locator position to the current locator position. The line attributes are specified in the data record.

C Data Record:
The pet_u4 member of the Ploc_data3 structure, defined in phigs.h as:

```c
struct {
    Pline_bundle line_bundle; /∗ line type, width and colour index ∗/
} pet_u4;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 2.

IA An array of integers with the following values:

IA(1) The line type of the rubber band line.

IA(2) The colour of the rubber band line.
The number of real values = 1.
RA Contains one real value in position RA(1) specifying the line width scale factor of the rubber band line.
SL The number of strings = 0.

PET –5 Display rubber banding edges of a rectangle, the diagonal of which connects the initial locator position to the current locator position. The line attributes of the edges are specified in the data record.

C Data Record:
The pet_u5 member of the Ploc_data3 structure, defined in phigs.h as:

```c
struct {
    Pline_bundle line_bundle; /* line type, width and colour index */
} pet_u5;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 2.
IA An array of integers with the following values:
IA(1) The line type of the rubber band rectangle.
IA(2) The colour of the rubber band rectangle.
RL The number of real values = 1.
RA Contains one real value in position RA(1) specifying the line width scale factor of the rubber band rectangle.
SL The number of strings = 0.

ERRORS
003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation category is not INPUT or OUTIN
114 Ignoring function, the view index value is less than zero
250 Ignoring function, the specified device is not available on the specified workstation
251 Ignoring function, the function requires the input device to be in REQUEST mode
253 Warning, the specified prompt/echo type is not available on the specified workstation.
254 Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
255 Ignoring function, one of the echo area/volume boundary points is outside the
range of the device

260 Ignoring function, one of the fields within the input device data record is in error. Prompt/echo type one will be used in its place

261 Ignoring function, initial value is invalid

SEE ALSO

ESCAPE -19 (3P)
SET LOCATOR MODE (3P)
REQUEST LOCATOR 3 (3P)
SAMPLE LOCATOR 3 (3P)
GET LOCATOR 3 (3P)
INQUIRE LOCATOR DEVICE STATE 3 (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
INITIALIZE LOCATOR (3P)
INITIALIZE PICK – initialize a PICK input device using 2D data

C Syntax
void
pinit_pick ( ws, dev, istat, init, pet, echo_area, record, order )

Pint ws;  /* workstation identifier */
Pint dev;  /* pick device number */
Pin_status istat;  /* initial pick status */
Ppick_path *init;  /* initial pick pointer */
Pint pet;  /* prompt and echo type */
Plimit *echo_area;  /* echo area pointer */
Ppick_data *record;  /* data record pointer */
Ppath_order order;  /* pick path order */

FORTRAN Syntax
SUBROUTINE pinpk ( WKID, PKDNR, ISTAT, IPPD, PP, PET, XMIN, XMAX, 
YMIN, YMAX, LDR, DATREC, PPORDR )

INTEGER WKID  /* workstation identifier */
INTEGER PKDNR  /* pick device number */
INTEGER ISTAT  /* initial status (POK, PNPICK) */
INTEGER IPPD  /* depth of initial pick path */
INTEGER PP(3,IPPD)  /* pick path */
INTEGER PET  /* prompt/echo type */
REAL XMIN, XMAX, YMIN, YMAX  /* echo area in device coordinates */
INTEGER LDR  /* dimension of data record array */
CHARACTER*80 DATREC(LDR)  /* data record */
INTEGER PPORDR  /* pick path order */
(PPOTOP,PPOBOT)

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
INITIALIZE PICK sets the initialization parameters of a PICK device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

The pick filter of a PICK device controls which output primitives on the device’s workstation are pickable. By default no output primitives are pickable. See SET PICK FILTER for more information on the pick filter.

C Input Parameters

ws  The workstation identifier of the workstation associated with the device.

dev  The device number of the PICK device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

istat  The PICK status of the initial measure. Valid values from the Pin_status
enumerated type defined in phigs.h are:

- PIN_STATUS_OK
- PIN_STATUS_NO_IN

**init**
A pointer to a Ppick_path structure containing the initial pick path, if any. This value is ignored if status is not PIN_STATUS_OK. Ppick_path is defined in phigs.h as:

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

The `depth` indicates the number of elements in the path.

**path_list** is the array of path elements defining the location of the primitive in the CSS. Ppick_path_elem is defined in phigs.h as:

```c
typedef struct {
    Pint  struct_id;   /* structure identifier */
    Pint  pick_id;     /* hierarchical pick identifier */
    Pint  elem_pos;    /* element sequence number */
} Ppick_path_elem;
```

The `struct_id`, `pick_id`, and `elem_pos` are the structure identifier, pick id, and element number, respectively, of each element in the path.

**pet**
The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

**echo_area**
A pointer to a Plimit structure specifying the x and y components of the echo volume, in Device Coordinates. The z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min;       /* minimum x coordinate value */
    Pfloat x_max;       /* maximum x coordinate value */
    Pfloat y_min;       /* minimum y coordinate value */
    Pfloat y_max;       /* maximum y coordinate value */
} Plimit;
```

**record**
A pointer to a Ppick_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Ppick_data structure for prompt/echo type 1 is `pet_r1`.

For some prompt/echo types the data record is not used; however, the record

modified 2 April 1993
parameter must still be specified.

**order**
The pick path order of paths returned by REQUEST PICK, GET PICK, and SAMPLE PICK. `Ppath_order` is an enumerated type defined in phigs.h as follows:

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

If the order is `TOP FIRST`, the structure specified in any pick path element is a parent of the structure specified in the subsequent pick path element. If the order is `BOTTOM FIRST`, the structure specified in any pick path element is a child of the structure specified in the subsequent pick path element.

### FORTRAN Input Parameters

**WKID**
The workstation identifier of the workstation associated with the device.

**PKDNR**
The device number of the PICK device to initialize. See the `AVAILABLE DEVICES` section below for a description of the available devices.

**ISTAT**
The pick status of the initial measure. Valid values as defined in phigs77.h are:

- **POK**  OK
- **PNPICK**  No pick

**IPPD**
The number of elements in the measure’s path. This value is ignored if `ISTAT` is `PNPICK`.

**PP**
An array containing the measure’s pick path. This value is ignored if `ISTAT` is `PNPICK`. This is the 2D array of path elements defining the location of the picked primitive in the CSS. Each row of the array contains the structure identifier, pick id, and element number, respectively, of each element in the path.

**PET**
The prompt/echo type desired. Those supported by each device are listed in the `AVAILABLE DEVICES` section below.

**XMIN, XMAX, YMIN, YMAX**
The x and y components of the echo volume, in Device Coordinates. The z component in the workstation state list is left unchanged.

**LDR**
The dimension of the data record array.

**DATREC**
A packed data record, built by `PACK DATA RECORD`, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the `AVAILABLE DEVICES` section.

**PPORDR**
The pick path order of paths returned by REQUEST PICK, GET PICK, and SAMPLE PICK. Valid values as defined in phigs77.h are:

- **PPOTOP**  Top first
- **PPOBOT**  Bottom first
If the order is Top First, then the structure specified in any pick path element is a parent of the structure specified in the subsequent pick path element. If the order is Bottom First, then the structure specified in any pick path element is a child of the structure specified in the subsequent pick path element.

**Execution**

INITIALIZE PICK sets the initialization parameters of a PICK device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are the initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure will retain this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A PICK device measure consists of a status and a pick path. Status indicates whether an initial pick path is specified. Pick path describes the location of the picked primitive, if any, in the Central Structure Store (CSS).

The **pick filter** of a PICK device controls which output primitives on the device’s workstation are pickable. By default no output primitives are pickable. See SET PICK FILTER for more information on the pick filter.

The prompt/echo type determines the display characteristics of the device—that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo type. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they don’t display there. This function specifies only the x and y components of the echo volume. The existing z component in the workstation state list is left unchanged.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be specified properly or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a PICK input device can be inquired for with the function `INQUIRE DEFAULT PICK DEVICE DATA`. The current state of the device can be inquired for with the function `INQUIRE PICK DEVICE STATE`.

### AVAILABLE DEVICES

**Devices 1 - 6**

Device 1: Cursor and Left Mouse Button Down Stroke  
Device 2: Cursor and Middle Mouse Button Down Stroke  
Device 3: Cursor and Right Mouse Button Down Stroke  
Device 4: Cursor and Left Mouse Button Up Stroke  
Device 5: Cursor and Middle Mouse Button Up Stroke  
Device 6: Cursor and Right Mouse Button Up Stroke

These devices are all associated with the X pointer device, usually the mouse and cursor. Using devices 1, 2, and 3, the operator picks an output primitive by positioning the pointer on or near the primitive and depressing the appropriate mouse button. Using devices 4, 5, and 6, the operator moves the mouse around, holding the mouse button down; SunPHIGS searches for and echoes any pickable primitives near the pointer position. When the operator releases the button, the highlighted primitive is selected. Echoing is accomplished according to the prompt echo type.

SunPHIGS searches the workstation’s list of posted structures for visible and pickable output primitives that intersect a 3D region around the pointer. This region is called the pick aperture.

The workstation’s pick testing traverses posted structure networks in priority order; but instead of drawing, the workstation tests each visible, pickable primitive against the pick aperture. If the primitive intersects the pick aperture, then the primitive is considered to be picked. (If the workstation’s HLHSR mode is z buffer, then the primitive’s intersection with the pick aperture must also satisfy the z buffer by being as close or closer to the viewer than other pickable primitives at that pixel. The last such picked primitive is the only primitive that produces the pick path. See SET HLHSR MODE (3P) and SET HLHSR IDENTIFIER (3P) for more information on HLHSR processing.)

Because a structure can appear multiple times on the workstation, this appearance of the picked primitive requires a path through EXECUTE STRUCTURE elements, to the picked primitive element, in order to uniquely specify it. PHIGS calls this a pick path.

If no pickable primitive is found to intersect the aperture, the PICK device measure is set to NO PICK. The device’s pick filter determines which output primitives are pickable.

**Prompt/echo types supported:** 1, 2, 3, −1, −2, −3, −4

In the following PET descriptions, blinking and highlighting mean alternating the colour of the primitives involved between their intrinsic colour and the highlight colour. Like a quick update method (see SET DISPLAY UPDATE STATE), these colour manipulations disregard structure priority. Therefore, after echoing, an implicit regeneration may (depending on the display update state) occur to correct the display.

modified 2 April 1993
PET 1 Highlight the picked primitive by blinking it twice. The highlight colour is the background colour. The default aperture is a 10-pixel-square box in the X and Y dimensions, and the full depth of the output device in the Z dimension.

The data record is not used for this PET.

PET 2 Highlight all the primitives in the structure with the same pick identifier as the picked primitive. The primitives are blinked highlight count times and are left displayed in each colour for highlight duration seconds. Aperture size is specified as the half-width of a box in DC units for the X and Y dimensions; the Z dimension is the full depth of the output device.

If highlight_count is negative, the absolute value is used as the number of times to blink, and the primitives are left displayed in the highlight colour. A regeneration of the workstation will return them to their intrinsic colour.

C Data Record:
The pet_r2 member of the Ppick_data structure, defined in phigs.h as:

```c
struct {
    Pint highl_colr;
    Pint highl_count; /* number of times to blink */
    Pfloat highl_duration; /* seconds per half blink */
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_r2;
```

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 2.

IA Contains two integer values:

IA(1) The pick highlight colour.

IA(2) The highlight count.

RL The number of real values = 2.

RA An array of reals in the following order:

RA(1) The highlight duration, in seconds.

RA(2) The aperture size, as the half-width of a box in DC units.

SL The number of strings = 0.

PET 3 Highlight the whole posted structure network containing the picked primitive.

The data record description is the same as for PET 2. The C structure member of Ppick_data is pet_r3.

PET −1 Highlight the picked primitive.
The data record description is the same as for PET 2. The C structure member of Ppick_data is pet_u1.

**PET −2** Highlight the longest contiguous interval of primitives within the same structure, with the same pick identifier, and containing the picked primitive. 

EXECUTE_STRUCTURE elements may be present in the interval, but descendents are not highlighted.

The data record description is the same as for PET 2. The C structure member of Ppick_data is pet_u2.

**PET −3** Highlight the whole structure containing the picked primitive, including descendents.

The data record description is the same as for PET 2. The C structure member of Ppick_data is pet_u3.

**PET −4** Highlight all primitives in the same structure as the picked primitive, but do not descend.

The data record description is the same as for PET 2. The C structure member of Ppick_data is pet_u4.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state ( \text{PHOP, WSOP, \ast, \ast} )</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>060</td>
<td>Ignoring function, specified workstation is not of category OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>251</td>
<td>Ignoring function, the function requires the input device to be in REQUEST mode</td>
</tr>
<tr>
<td>253</td>
<td>Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place</td>
</tr>
<tr>
<td>254</td>
<td>Ignoring function, invalid echo area/volume; ( XMIN \geq XMAX, YMIN \geq YMAX, ) or ( ZMIN &gt; ZMAX )</td>
</tr>
<tr>
<td>255</td>
<td>Ignoring function, one of the echo area/volume boundary points is outside the range of the device</td>
</tr>
<tr>
<td>260</td>
<td>Ignoring function, one of the fields within the input device data record is in error</td>
</tr>
<tr>
<td>261</td>
<td>Ignoring function, initial value is invalid</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET PICK FILTER \((3P)\)
- SET PICK IDENTIFIER \((3P)\)
- ESCAPE -19 \((3P)\)
- SET PICK MODE \((3P)\)
- REQUEST PICK \((3P)\)
- SAMPLE PICK \((3P)\)
- GET PICK \((3P)\)
INQUIRE PICK DEVICE STATE (3P)
INITIALIZE PICK 3 (3P)
NAME

INITIALIZE PICK 3 – initialize a PICK input device. IX “Pick Input Devices” “INITIALIZE PICK 3”

SYNOPSIS

C Syntax

```c
void pinit_pick3 ( ws, dev, istat, init, pet, echo_volume, record, order )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pint</td>
<td>ws</td>
<td>workstation identifier</td>
</tr>
<tr>
<td>Pint</td>
<td>dev</td>
<td>pick device number</td>
</tr>
<tr>
<td>Pin_status</td>
<td>istat</td>
<td>initial pick status</td>
</tr>
<tr>
<td>Ppick_path</td>
<td>*init;</td>
<td>initial pick pointer</td>
</tr>
<tr>
<td>Pint</td>
<td>pet;</td>
<td>prompt and echo type</td>
</tr>
<tr>
<td>Plimit3</td>
<td>*echo_volume;</td>
<td>echo volume pointer</td>
</tr>
<tr>
<td>Ppick_data3</td>
<td>*record;</td>
<td>data record pointer</td>
</tr>
<tr>
<td>Ppath_order</td>
<td>order;</td>
<td>pick path order</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

```fortran
SUBROUTINE pinpk3 ( WKID, PKDNR, ISTAT, IPPD, PP, PET, EVOL, LDR, DATREC, PPORDR )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>WKID</td>
<td>workstation identifier</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PKDNR</td>
<td>pick device number</td>
</tr>
<tr>
<td>INTEGER</td>
<td>ISTAT</td>
<td>initial status (POK, PN PICK)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IPPD</td>
<td>depth of initial pick path</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PP(3, IPPD)</td>
<td>pick path</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PET</td>
<td>prompt/echo type</td>
</tr>
<tr>
<td>REAL</td>
<td>EVOL(6)</td>
<td>echo volume (DC)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>LDR</td>
<td>dimension of data record array</td>
</tr>
<tr>
<td>CHARACTER*80</td>
<td>DATREC(LDR)</td>
<td>data record</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PPORDR</td>
<td>pick path order (PPTOP, PPOBOT)</td>
</tr>
</tbody>
</table>

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

**Purpose**

Use INITIALIZE PICK 3 to set the initialization parameters of a PICK device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

**Note:** The specified device must be in REQUEST mode when this function is called.

The **pick filter** of a PICK device controls which output primitives on the device’s workstation are pickable. By default no output primitives are pickable. See SET PICK FILTER for more information on the pick filter.

**C Input Parameters**

- **ws**: The workstation identifier of the workstation associated with the device.
- **dev**: The device number of the PICK device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

206 modified 2 April 1993
The **istat** is the pick status of the initial measure. Valid values from the `Pin_status` enumerated type defined in `phigs.h` are:

```c
PIN_STATUS_OK
PIN_STATUS_NO_IN
```

The **init** is a pointer to a `Ppick_path` structure containing the initial pick path, if any. This value is ignored if `status` is not `PIN_STATUS_OK`. `Ppick_path` is defined in `phigs.h` as:

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

The `depth` indicates the number of elements in the path.

`path_list` is the array of path elements defining the location of the primitive in the CSS. `Ppick_path_elem` is defined in `phigs.h` as:

```c
typedef struct {
    Pint struct_id; /* structure identifier */
    Pint pick_id; /* hierarchical pick identifier */
    Pint elem_pos; /* element sequence number */
} Ppick_path_elem;
```

The `struct_id`, `pick_id`, and `elem_pos` are the structure identifier, pick id, and element number, respectively, of each element in the path.

The **pet** is the prompt/echo type desired. Those supported by each device are listed in the `AVAILABLE DEVICES` section below.

The **echo_volume** is a pointer to a `Plimit3` structure specifying the echo volume, in Device Coordinates. `Plimit3` is defined in `phigs.h` as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
```

The **record** is a pointer to a `Ppick_data3` structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the `AVAILABLE DEVICES` section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the `Ppick_data3`
structure for prompt/echo type 1 is pet_r1.
For some prompt/echo types the data record is not used; however, the record parameter must still be specified.

order
The pick path order of paths returned by REQUEST PICK, GET PICK, and SAMPLE PICK. Ppath_order is an enumerated type defined in phigs.h as follows:

typedef enum {
    PORDER_TOP_FIRST,
    PORDER_BOTTOM_FIRST
} Ppath_order;
If the order is TOP FIRST, the structure specified in any pick path element is a parent of the structure specified in the subsequent pick path element. If the order is BOTTOM FIRST, the structure specified in any pick path element is a child of the structure specified in the subsequent pick path element.

FORTRAN Input

Parameters

WKID  The workstation identifier of the workstation associated with the device.

PKDNR The device number of the PICK device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

ISTAT The pick status of the initial measure. Valid values as defined in phigs77.h are:
    POK   OK
    PNPICK No pick

IPPD  The number of elements in the measure’s path. This value is ignored if ISTAT is PNPICK.

PP    An array containing the measure’s pick path. This value is ignored if ISTAT is PNPICK. This is the 2D array of path elements defining the location of the picked primitive in the CSS. Each row of the array contains the structure identifier, pick id, and element number, respectively, of each element in the path.

PET   The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

EVL   The x, y, and z limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, in Device Coordinates.

LDR   The dimension of the data record array.

DATREC A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

PPORDR The pick path order of paths returned by REQUEST PICK, GET PICK, and SAMPLE PICK. Valid values as defined in phigs77.h are:
    PPOTOP   Top first
**Execution**

INITIALIZE PICK 3 sets the initialization parameters of a PICK device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure will retain this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A PICK device measure consists of a status and a pick path. Status indicates whether an initial pick path is specified. Pick path describes the location of the picked primitive, if any, in the Central Structure Store (CSS).

The pick filter of a PICK device controls which output primitives on the device’s workstation are pickable. By default no output primitives are pickable. See SET PICK FILTER for more information on the pick filter.

The prompt/echo type determines the display characteristics of the device, —that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they don’t display there.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be specified properly or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a PICK input device can be inquired for with the function INQUIRE DEFAULT PICK DEVICE DATA 3. The current state of the device can be inquired for with the function INQUIRE PICK DEVICE STATE 3.

### AVAILABLE DEVICES

**Devices 1 - 6**

- Device 1: Cursor and Left Mouse Button Down Stroke
- Device 2: Cursor and Middle Mouse Button Down Stroke
- Device 3: Cursor and Right Mouse Button Down Stroke
- Device 4: Cursor and Left Mouse Button Up Stroke
- Device 5: Cursor and Middle Mouse Button Up Stroke
- Device 6: Cursor and Right Mouse Button Up Stroke

These devices are all associated with the X pointer device, usually the mouse and cursor. Using devices 1, 2, and 3, the operator picks an output primitive by positioning the pointer on or near the primitive and depressing the appropriate mouse button. Using devices 4, 5, and 6, the operator moves the mouse around, holding the mouse button down; SunPHIGS searches for and echoes any pickable primitives near the pointer position. When the operator releases the button, the highlighted primitive is selected. Echoing is accomplished according to the prompt/echo type.

SunPHIGS searches the workstation's list of posted structures for visible and pickable output primitives that intersect a 3D region around the pointer. This region is called the **pick aperture**.

The workstation's pick testing traverses posted structure networks in priority order; but instead of drawing, the workstation tests each visible, pickable primitive against the pick aperture. If the primitive intersects the pick aperture, then the primitive is considered to be **picked**. (If the workstation's HLHSR mode is z-buffer, then the primitive’s intersection with the pick aperture must also satisfy the z-buffer by being as close or closer to the viewer than other pickable primitives at that pixel. The last such picked primitive is the only primitive that produces the pick path. If the workstation’s HLHSR mode is 0, see SET HLHSR MODE (3P) and SET HLHSR IDENTIFIER (3P) for more information on HLHSR processing.)

Because a structure can appear multiple times on the workstation, this appearance of the picked primitive requires a path through EXECUTE STRUCTURE elements, to the picked primitive element, in order to uniquely specify it. PHIGS calls this a pick path.

If no pickable primitive is found to intersect the aperture, the PICK device measure is set to NO PICK. The device’s pick filter determines which output primitives are pickable.

**Prompt/echo types supported:** 1, 2, 3, −1, −2, −3, −4

In the following PET descriptions, **blinking** and **highlighting** mean alternating the colour of the primitives involved between their intrinsic colour and the **highlight colour**. Like a quick update method (see SET DISPLAY UPDATE STATE), these colour manipulations disregard structure priority. Therefore, after echoing, an implicit regeneration may (depending on the display update state) occur to correct the display.
PET 1  Highlight the picked primitive by blinking it twice. The highlight colour is the background colour. The default aperture is a 10-pixel-square box in the X and Y dimensions, and the full depth of the output device in the Z dimension.

The data record is not used for this PET.

PET 2  Highlight all the primitives in the structure with the same pick identifier as the picked primitive. The primitives are blinked highlight count times and are left displayed in each colour for highlight duration seconds. Aperture size is specified as the half-width of a box in DC units for the X and Y dimensions; the Z dimension is the full depth of the output device.

If highlight_count is negative, the absolute value is used as the number of times to blink, and the primitives are left displayed in the highlight colour. A regeneration of the workstation returns them to their intrinsic colour.

C Data Record:

The pet_r2 member of the Ppick_data3 structure, defined in phigs.h as:

```c
define struct {
    Pint highl_colr;
    Pint highl_count; /* number of times to blink */
    Pfloat highl_duration; /* seconds per half blink */
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_r2;
```

FORTRAN Data Record:

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL  The number of integers = 2.

IA  Contains two integer values:

IA(1)  The pick highlight colour.
IA(2)  The highlight count.

RL  The number of real values = 2.

RA  An array of reals in the following order:

RA(1)  The highlight duration, in seconds.
RA(2)  The aperture size, as the half-width of a box in DC units.

SL  The number of strings = 0.

PET 3  Highlight the whole posted structure network containing the picked primitive.

The data record description is the same as for PET 2. The C structure member of Ppick_data3 is pet_r3.

PET –1  Highlight the picked primitive.
The data record description is the same as for PET 2. The C structure member of Ppick_data3 is pet_u1.

**PET −2**
Highlight the longest contiguous interval of primitives within the same structure with the same pick identifier, and containing the picked primitive. EXECUTE_STRUCTURE elements may be present in the interval, but descendants are not highlighted.

The data record description is the same as for PET 2. The C structure member of Ppick_data3 is pet_u2.

**PET −3**
Highlight the whole structure containing the picked primitive, including descendants.

The data record description is the same as for PET 2. The C structure member of Ppick_data3 is pet_u3.

**PET −4**
Highlight all primitives in the same structure as the picked primitive, but do not descend.

The data record description is the same as for PET 2. The C structure member of Ppick_data3 is pet_u4.

**ERRORS**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>060</td>
<td>Ignoring function, specified workstation is not of category OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>251</td>
<td>Ignoring function, the function requires the input device to be in REQUEST mode</td>
</tr>
<tr>
<td>253</td>
<td>Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place</td>
</tr>
<tr>
<td>254</td>
<td>Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN &gt; ZMAX</td>
</tr>
<tr>
<td>255</td>
<td>Ignoring function, one of the echo area/volume boundary points is outside the range of the device</td>
</tr>
<tr>
<td>260</td>
<td>Ignoring function, one of the fields within the input device data record is in error</td>
</tr>
<tr>
<td>261</td>
<td>Ignoring function, initial value is invalid</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET PICK FILTER (3P)
- SET PICK IDENTIFIER (3P)
- ESCAPE -19 (3P)
- SET PICK MODE (3P)
- REQUEST PICK (3P)
- SAMPLE PICK (3P)
- GET PICK (3P)
INQUIRE PICK DEVICE STATE 3 (3P)
INITIALIZE PICK (3P)
INITIALIZE STRING – initialize a STRING input device using 2D data

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pinst ( WKID, STDNR, LSTR, ISTR, PET, XMIN, XMAX, YMIN,
                    YMAX, LDR, DATREC )
    INTEGER WKID                        /* workstation identifier */
    INTEGER STDNR                       /* string device number */
    INTEGER LSTR                        /* length of the initial string (>= 0) */
                                          /* (the number of characters actually used is the minimum of */
                                          /* LSTR and the length of ISTR) */
    CHARACTER*80 ISTR                   /* initial string */
    INTEGER PET                         /* prompt/echo type */
    REAL XMIN, XMAX, YMIN, YMAX          /* echo area in device coordinates */
    INTEGER LDR                         /* dimension of data record array */
    CHARACTER*80 DATREC(LDR)            /* data record */
```

**FORTRAN Subset Syntax**

```fortran
SUBROUTINE pinst ( WKID, STDNR, LSTR, ISTR, PET, XMIN, XMAX, YMIN,
                    YMAX, LDR, DATREC )
    INTEGER WKID                        /* workstation identifier */
    INTEGER STDNR                       /* string device number */
    INTEGER LSTR                        /* length of the initial string (>= 0) */
                                          /* (the number of characters actually used is the minimum of */
                                          /* LSTR and the length of ISTR) */
    CHARACTER*80 ISTR                   /* initial string */
    INTEGER PET                         /* prompt/echo type */
    REAL XMIN, XMAX, YMIN, YMAX          /* echo area in device coordinates */
    INTEGER LDR                         /* dimension of data record array */
    CHARACTER*80 DATREC(LDR)            /* data record */
```

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

modified 2 April 1993
DESCRIPTION

Purpose

INITIALIZE STRING sets the initialization parameters of a STRING device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

ws The workstation identifier of the workstation associated with the device.

dev The device number of the STRING device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

init A pointer to the initial string. init can be NULL, that is, (char*)0.

pet The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

echo_area A pointer to a Plimit structure defining the x and y components of the echo volume, in Device Coordinates (DC). The z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
} Plimit;
```

record A pointer to a Pstring_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Pstring_data structure for prompt/echo type 1 is pet_r1.

For some prompt/echo types of some STRING devices the data record is not used. The record parameter, however, must still be supplied.

Pstring_data is defined in phigs.h as follows:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
} union {
    struct {
    Pint unused;
    } pet_r1;
};
```
FORTRAN Input
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>STDNR</td>
<td>The device number of the STRING device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.</td>
</tr>
<tr>
<td>LSTR</td>
<td>The length of the initial string.</td>
</tr>
<tr>
<td>ISTR</td>
<td>The initial string.</td>
</tr>
<tr>
<td>PET</td>
<td>The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.</td>
</tr>
<tr>
<td>XMIN, XMAX, YMIN, YMAX</td>
<td>The x and y components of the echo volume, in Device Coordinates (DC). The z component in the workstation state list is left unchanged.</td>
</tr>
<tr>
<td>LDR</td>
<td>The dimension of the data record array.</td>
</tr>
<tr>
<td>DATREC</td>
<td>A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.</td>
</tr>
</tbody>
</table>

Execution

INITIALIZE STRING sets the initialization parameters of a STRING device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A STRING device measure consists of a character string.

The prompt/echo type determines the display characteristics of the device — that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the
region even though they do not display there. This function only specifies the x and y components of the echo volume. The existing z component in the workstation state list is left unchanged.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be specified properly, or this function generates an error. The ERRORS section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a STRING input device can be inquired for with the function INQUIRE DEFAULT STRING DEVICE DATA. The current state of the device can be inquired for with the function INQUIRE STRING DEVICE STATE.

### AVAILABLE DEVICES

#### Device 1 – OLIT Text Field Widget

The string device realization is a collection of widgets from the OLIT widget set. It consists of a Shell widget and a TextField widget. The TextField widget is the item manipulated by the operator to change the string value.

String device events are generated by a carriage return. The device’s measure corresponds to all the characters added to the widget by the operator since the device was last enabled or the last STRING event was generated.

The echo volume is not used by this device. Applications or users can specify the position of the device (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** 1

**PET 1**  
Display the text in the widget.

**C Data Record:**

There are no PET-specific data for PET 1, simply the buffer_size and init_pos members of the Pstring_data structure.

**FORTRAN Data Record:**

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL**  
The number of integers = 2.
- **IA**  
Contains two integer values:
  - **IA(1)**  
The input buffer size.
  - **IA(2)**  
The initial editing position.
- **RL**  
The number of real values = 0.
- **SL**  
The number of strings = 0.
The widgets used and their names, in decreasing hierarchy, are:

\[
\begin{align*}
\text{string1} & \rightarrow \text{popupWindowShellWidgetClass} \\
\text{text} & \rightarrow \text{textFieldWidgetClass}
\end{align*}
\]

Fallback resources for string devices are:

\[
\begin{align*}
\ast \text{string1} \ast & \rightarrow \text{grey} \\
\ast \text{string1} \ast & \rightarrow 400
\end{align*}
\]

The fully qualified name of all widgets is:

\[
<\text{appl_name}>.\text{workstation}<\text{ws_id}>.\text{string}<\text{dev_id}>.<\text{widget_name}>
\]

where \(<\text{appl_name}>\) is the application name specified in the call to OPEN XPHIGS. (This will be "phigs" if OPEN XPHIGS was not called.)

For example, phigs.workstation1.string1.text is the name of the Text Field widget of string device 1 on workstation 1.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 061 Ignoring function, specified workstation’s category is not INPUT or OUTIN
- 250 Ignoring function, the specified device is not available on the specified workstation
- 251 Ignoring function, the function requires the input device to be in REQUEST mode
- 253 Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
- 254 Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
- 255 Ignoring function, one of the echo area/volume boundary points is outside the range of the device
- 260 Ignoring function, one of the fields within the input device data record is in error
- 261 Ignoring function, initial value is invalid
- 263 Ignoring function, length of the initial string is greater than the buffer size

**SEE ALSO**

ESCAPE -19 (3P)
SET STRING MODE (3P)
REQUEST STRING (3P)
SAMPLE STRING (3P)
GET STRING (3P)
INQUIRE STRING DEVICE STATE (3P)
INITIALIZE STRING 3 (3P)
### NAME

INITIALIZE STRING 3 – initialize a STRING input device using 3D data

### SYNOPSIS

#### C Syntax

```c
void pinit_string3 ( ws, dev, init, pet, echo_volume, record )

Pint ws; // workstation identifier
Pint dev; // string device number
char *init; // initial string
Pint pet; // prompt and echo type
Plimit3 *echo_volume; // echo volume pointer
Pstring_data3 *record; // data record pointer
```

#### FORTRAN Syntax

```fortran
SUBROUTINE pinst3 ( WKID, STDNR, LSTR, ISTR, PET, EVOL, LDR, DATREC )

INTEGER WKID // workstation identifier
INTEGER STDNR // string device number
INTEGER LSTR // length of the initial string (>= 0)
CHARACTER(*) ISTR // initial string
INTEGER PET // prompt/echo type
REAL EVOL(6) // echo volume (DC)
INTEGER LDR // dimension of data record array
CHARACTER*80 DATREC(LDR) // data record
```

#### FORTRAN Subset Syntax

```fortran
SUBROUTINE pinst3 ( WKID, STDNR, LSTR, ISTR, PET, EVOL, LDR, DATREC )

INTEGER WKID // workstation identifier
INTEGER STDNR // string device number
INTEGER LSTR // length of the initial string (>= 0)
CHARACTER*80 ISTR // initial string
INTEGER PET // prompt/echo type
REAL EVOL(6) // echo volume (DC)
INTEGER LDR // dimension of data record array
CHARACTER*80 DATREC(LDR) // data record
```

### Required PHIGS Operating States

( PHOP, WSOP, *, * )

### DESCRIPTION

#### Purpose

INITIALIZE STRING 3 sets the initialization parameters of a STRING device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

**Note:** The specified device must be in REQUEST mode when this function is called.

---

modified 2 April 1993
C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ws</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>dev</td>
<td>The device number of the STRING device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.</td>
</tr>
<tr>
<td>init</td>
<td>A pointer to the initial string. init can be NULL, that is, (char*)0.</td>
</tr>
<tr>
<td>pet</td>
<td>The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.</td>
</tr>
<tr>
<td>echo_volume</td>
<td>A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates (DC). Plimit3 is defined in phigs.h as follows:</td>
</tr>
<tr>
<td>record</td>
<td>A pointer to a Pstring_data3 structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.</td>
</tr>
</tbody>
</table>

The members of the data record union correspond to the prompt/echo type being used. As an example, the appropriate member of the Pstring_data3 structure for prompt/echo type 1 is pet_r1.

For some prompt/echo types of some STRING devices, the data record is not used. The record parameter, however, must still be supplied.

Pstring_data3 is defined in phigs.h as follows:

typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    union {
        struct {
            Pint unused;
        } pet_r1;
    } pets;
} Pstring_data3;

FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>STDNR</td>
<td>The device number of the STRING device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.</td>
</tr>
</tbody>
</table>
The length of the initial string.

The initial string.

The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

The $x$, $y$, and $z$ limits of the echo volume, $XMIN$, $XMAX$, $YMIN$, $YMAX$, $ZMIN$, $ZMAX$, in Device Coordinates (DC).

The dimension of the data record array.

A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

Execution

INITIALIZE STRING 3 sets the initialization parameters of a STRING device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A STRING device measure consists of a character string.

The prompt/echo type determines the display characteristics of the device — that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they do not display there.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be properly specified, or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a
STRING input device can be inquired for with the function INQUIRE DEFAULT STRING DEVICE
DATA 3. The current state of the device can be inquired for with the function INQUIRE
STRING DEVICE STATE 3.

**AVAILABLE DEVICES**

**Device 1 - OLIT Text Field Widget**

The string device realization is a collection of widgets from the OLIT widget set. It consists of a Shell widget and a TextField widget. The TextField widget is the item manipulated by the operator to change the *string value*.

String device events are generated by a carriage return. The device’s measure corresponds to all the characters added to the widget by the operator since the device was last enabled or the last STRING event was generated.

The echo volume is not used by this device. Applications or users can specify the position of the device (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** 1

**PET 1** Display the text in the widget.

**C Data Record:**

There are no PET-specific data for PET 1, simply the *buffer_size* and *init_pos* members of the Pstring_data3 structure.

**FORTRAN Data Record:**

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of integers = 2.
- **IA** Contains two integer values:
  - **IA(2)** The input buffer size.
  - **IA(2)** The initial editing position.
- **RL** The number of real values = 0.
- **SL** The number of strings = 0.

The widgets used and their names, in decreasing hierarchy, are:

- `string1` → `popupWindowShellWidgetClass`
- `text` → `textFieldWidgetClass`

Fallback resources for string devices are:

- `*string1*background` → `grey`
- `*string1*textwidth` → `400`
The fully qualified name of all widgets is:

```
<appl_name>.workstation<ws_id>.string<dev_id>.<widget_name>
```

where `<appl_name>` is the application name specified in the call to `OPEN XPHIGS`. (This will be "phigs" if `OPEN XPHIGS` was not called.)

For example, `phigs.workstation1.string1.text` is the name of the Text Field widget of string device 1 on workstation 1.

### ERRORS

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **061** Ignoring function, specified workstation’s category is not INPUT or OUTIN
- **250** Ignoring function, the specified device is not available on the specified workstation
- **251** Ignoring function, the function requires the input device to be in REQUEST mode
- **253** Warning, the specified prompt/echo type is not available on the specified workstation Prompt/echo type one will be used in its place
- **254** Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
- **255** Ignoring function, one of the echo area/volume boundary points is outside the range of the device
- **260** Ignoring function, one of the fields within the input device data record is in error
- **261** Ignoring function, initial value is invalid
- **263** Ignoring function, length of the initial string is greater than the buffer size

### SEE ALSO

- `ESCAPE -19 (3P)`
- `SET STRING MODE (3P)`
- `REQUEST STRING (3P)`
- `SAMPLE STRING (3P)`
- `GET STRING (3P)`
- `INQUIRE STRING DEVICE STATE 3 (3P)`
- `INITIALIZE STRING (3P)`

**modified 2 April 1993**
NAME

INITIALIZE STROKE – initialize a stroke input device using 2D data

SYNOPSIS

C Syntax

void

pinit_stroke ( ws, dev, init_view_ind, init_stroke, pet, echo_area, record )

Pint ws;     /* workstation identifier */
Pint dev;    /* stroke device number */
Pint init_view_ind;     /* initial view indicator */
Ppoint_list *init_stroke;    /* initial stroke pointer */
Pint pet;    /* prompt and echo type */
Plimit *echo_area;    /* echo area pointer */
Pstroke_data *record;    /* data record pointer */

FORTRAN Syntax

SUBROUTINE pinsk ( WKID, SKDNR, IVIEWI, N, IPX, IPY, PET, XMIN, XMAX, YMIN, YMAX, LDR, DATREC )

INTEGER WKID    /* workstation identifier */
INTEGER SKDNR    /* stroke device number */
INTEGER IVIEWI    /* initial view index */
INTEGER N    /* number of points in initial stroke */
REAL IPX(∗), IPY(∗)    /* points in initial stroke (WC) */
INTEGER PET    /* prompt/echo type */
REAL XMIN, XMAX, YMIN, YMAX    /* echo area in device coordinates */
INTEGER LDR    /* dimension of data record array */
CHARACTER∗80 DATREC(LDR)    /* data record */

Required PHIGS Operating States

( PHOP, WSOP, ∗, ∗ )

DESCRIPTION

Purpose

Use INITIALIZE STROKE to set the initialization parameters of a STROKE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

ws     The workstation identifier of the workstation associated with the device.

dev     The device number of the STROKE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

init_view_ind

The index of the view representation in the workstation’s view table to use to map the specified initial positions from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

init_stroke

A pointer to a Ppoint_list structure containing the initial stroke. The Ppoint_list
The structure is defined in phigs.h as follows:

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

`num_points` is the number of points in the initial stroke. This must be less than or equal to the buffer size specified in the input data record, `record`.

`points` is an array of Ppoint structures specifying the x and y WC of the initial stroke points. Ppoint is defined in phigs.h as follows:

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

`pet` The prompt/echo type desired. Those supported by each device are listed in the `AVAILABLE DEVICES` section below.

echo_area A pointer to a Plimit structure defining the x and y components of the echo area, in Device Coordinates (DC). The z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
} Plimit;

`record` A pointer to a Pstroke_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the `AVAILABLE DEVICES` section. Pstroke_data is defined in phigs.h as:

typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    Pfloat x_interval; /* x interval */
    Pfloat y_interval; /* y interval */
    Pfloat time_interval; /* time interval */
    union {
        struct {
            Pint unused;
        } pet_r1;
        struct {
            // more structures...
        } pet_r2;
    };
} Pstroke_data;
FORTRAN Input Parameters

WKID  The workstation identifier of the workstation associated with the device.

SKDNR  The device number of the STROKE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

IVIEWI  The view index of the view representation in the workstation’s view table to use to map the specified initial stroke points from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

N  The number of points in the initial stroke.

IPX, IPY  Arrays of x and y WC values specifying the initial stroke points.

PET  The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

XMIN, XMAX, YMIN, YMAX  The x and y limits of the echo volume, in Device Coordinates (DC). The z component in the workstation state list is left unchanged.

LDR  The dimension of the data record array.

DATREC  A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

Execution  INITIALIZE STROKE sets the initialization parameters of a STROKE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this
function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device is set to whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A STROKE device measure consists of points and a view index. The points are World Coordinate (WC) points corresponding to the positions on the workstation selected by the operator. The view index is the index of the view representation used to transform the stroke positions from Normalized Projection Coordinates (NPC) to WC. This view representation is determined by selecting the highest priority representation that contains all the stroke positions within its NPC limits. See SET VIEW TRANSFORMATION INPUT PRIORITY for more information. The workstation transform is used to transform the operator-selected positions from Device Coordinates (DC) to NPC.

The initial stroke points are transformed to DC by applying the view orientation and view mapping transforms of the specified view representation, then applying the workstation transformation. If the view index is invalid, an error is generated.

The prompt/echo type determines the display characteristics of the device—that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in DC. Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they do not display there. This function specifies only the x and y components of the echo volume. The existing z component in the workstation state list is left unchanged.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

Stroke devices contain a buffer in which to store the points selected by the operator. Points are added to the buffer starting at the edit position. Both the buffer size and the edit position are specified in the input data record.

All the initialization parameters must be specified properly, or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a STROKE input device can be inquired for with the function INQUIRE DEFAULT STROKE DEVICE DATA. The current state of the device can be inquired for with the function INQUIRE STROKE DEVICE STATE.

### AVAILABLE DEVICES

**Device 1 - Cursor and Left Mouse Button**

**Device 2 - Cursor and Middle Mouse Button**

**Device 3 - Cursor and Right Mouse Button**

These devices are all associated with the X pointer device, usually the mouse and cursor. The stroke points in the STROKE device’s measure are the WC points corresponding to the positions specified in an operator-generated series of X pointer events. The operator adds points to the buffer by positioning the cursor to the desired location and depressing the appropriate pointer button. Once all the desired points are selected this way the operator presses both the shift key and the appropriate pointer button at the same time. This triggers the stroke. Points can be removed from the buffer by pressing the CONTROL key and pointer button simultaneously. Only points after the edit position can be removed in this way.

The STROKE echo is removed from the workstation when the cursor leaves the echo area.

**Prompt/echo types supported:** 1, −3, −4

**PET 1**

Display the stroke points by drawing a marker at each stroke position. The marker used is of marker type 2 (plus), size 1.0, and uses colour index 1.

**C Data Record:**

There are no PET-specific data for PET 1; only the buffer_size and init_pos members of the Pstroke_data structure are used.

Neither the space nor the time intervals of Pstroke_data are currently used by any stroke device.

**FORTRAN Data Record:**

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of integers = 2.
- **IA** Contains two integer values:
  - **IA(1)** The input buffer size.
  - **IA(2)** The edit position.
The number of real values = 3.

An array of reals containing:

RA(1) The x value of the space interval.
RA(2) The y value of the space interval.
RA(3) The time interval.

All these values should be 0; neither the space nor the time intervals are currently used by any stroke device.

The number of strings = 0.

Display the stroke points by drawing a marker at each stroke position. The marker attributes to use are specified in the data record.

C Data Record:

A Pstroke_data structure defined in phigs.h, the relevant members of which are:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos;   /* initial editing position */
    Pfloat x_interval; /* x interval */
    Pfloat y_interval; /* y interval */
    Pfloat time_interval; /* time interval */
    union {
        struct {
            Pmarker_bundle marker_bundle;
        } pet_u3;
    } pets;
} Pstroke_data;
```

Neither the space nor the time intervals of Pstroke_data are currently used by any stroke device.

FORTRAN Data Record:

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 4.

IA Contains four integer values:

IA(1) The input buffer size.
IA(2) The edit position.
IA(3) The marker type.
IA(4) The marker colour index.

RL The number of real values = 4.

RA An array of reals containing:

RA(1) The x value of the space interval.
RA(2) The y value of the space interval.
RA(3) The time interval.
RA(4) The marker size.

All the interval values should be 0; neither the space nor the time intervals are currently used by any stroke device.

SL The number of strings = 0.

PET − 4 Display the stroke points by drawing a line between each stroke point. The line attributes to use are specified in the data record.

C Data Record:
A Pstroke_data structure defined in phigs.h, the relevant members of which are:

typedef struct {
    Pint buffer_size; /* input buffer size */
Pint init_pos; /* initial editing position */
Pfloat x_interval; /* x interval */
Pfloat y_interval; /* y interval */
Pfloat time_interval; /* time interval */
union {
    struct {
        Pline_bundle line_bundle;
    } pet_u4;
} pets;
} Pstroke_data;

Neither the space nor the time intervals of Pstroke_data are currently used by any stroke device.

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL The number of integers = 4.
IA Contains four integer values:
    IA(1) The input buffer size.
    IA(2) The edit position.
    IA(3) The line type.
    IA(4) The line colour index.

RL The number of real values = 4.
RA An array of reals containing:
    RA(1) The x value of the space interval.
    RA(2) The y value of the space interval.
    RA(3) The time interval.

modified 2 April 1993
RA(4)  The line width.
All the interval values should be 0; neither the space nor the time intervals are currently used by any stroke device.

SL  The number of strings = 0.

ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation’s category is not INPUT or OUTIN
114  Ignoring function, the view index value is less than zero
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode
253  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
254  Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
255  Ignoring function, one of the echo area/volume boundary points is outside the range of the device
260  Ignoring function, one of the fields within the input device data record is in error
253  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
261  Ignoring function, initial value is invalid
262  Ignoring function, number of points in the initial stroke is greater than the buffer size
114  Ignoring function, the view index value is less than zero

SEE ALSO

ESCAPE -19 (3P)
SET STROKE MODE (3P)
REQUEST STROKE (3P)
SAMPLE STROKE (3P)
GET STROKE (3P)
INQUIRE STROKE DEVICE STATE (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
INITIALIZE STROKE 3 (3P)
INITIALIZE STROKE 3 – initialize a stroke input device using 3D data

C Syntax

```c
void pinit_stroke3 ( ws, dev, init_view_ind, init_stroke, pet, echo_volume, rec )
```

- `Pint ws;` - workstation identifier
- `Pint dev;` - stroke device number
- `Pint init_view_ind;` - initial view indicator
- `Ppoint_list3 *init_stroke;` - initial stroke pointer
- `Pint pet;` - prompt and echo type
- `Plimit3 *echo_volume;` - echo volume pointer
- `Pstroke_data3 *rec;` - data record pointer

FORTRAN Syntax

```fortran
SUBROUTINE pinsk3 ( WKID, SKDNR, IVIEWI, N, IPX, IPY, IPZ, PET, EVOL, LDR, DATREC )
```

- `INTEGER WKID` - workstation identifier
- `INTEGER SKDNR` - stroke device number
- `INTEGER IVIEWI` - initial view index
- `INTEGER N` - number of points in initial stroke
- `REAL IPX(*), IPY(*), IPZ(*)` - points in initial stroke (WC)
- `INTEGER PET` - prompt/echo type
- `REAL EVOL(6)` - echo volume (DC)
- `INTEGER LDR` - dimension of data record array
- `CHARACTER*80 DATREC(LDR)` - data record

**Required PHIGS Operating States**

- `(PHOP, WSOP, *, *)`

**DESCRIPTION**

**Purpose**

Use INITIALIZE STROKE 3 to set the initialization parameters of a STROKE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

**Note:** The specified device must be in REQUEST mode when this function is called.

**C Input Parameters**

- `ws` - The workstation identifier of the workstation associated with the device.
- `dev` - The device number of the STROKE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
- `init_view_ind` - The index of the view representation in the workstation’s view table to use to map the specified initial positions from World Coordinates (WC) to Normalized Projection Coordinates (NPC).
- `init_stroke` - A pointer to a Ppoint_list3 structure containing the initial stroke. The
Ppoint_list3 structure is defined in phigs.h as follows:

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

num_points is the number of points in the initial stroke. This must be less than or equal to the buffer size specified in the input data record, rec.

points is an array of Ppoint3 structures specifying the x, y, and z WC of the initial stroke points. Ppoint3 is defined in phigs.h as follows:

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

pet The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

echo_volume A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates (DC). Plimit3 is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min;    /* minimum x coordinate value */
    Pfloat x_max;    /* maximum x coordinate value */
    Pfloat y_min;    /* minimum y coordinate value */
    Pfloat y_max;    /* maximum y coordinate value */
    Pfloat z_min;    /* minimum z coordinate value */
    Pfloat z_max;    /* maximum z coordinate value */
} Plimit3;

rec A pointer to a Pstroke_data3 structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

Pstroke_data3 is defined in phigs.h as follows:

typedef struct {
    Pint buffer_size;    /* input buffer size */
    Pint init_pos;       /* initial editing position */
    Pfloat x_interval;   /* x interval */
    Pfloat y_interval;   /* y interval */
    Pfloat z_interval;   /* z interval */
    Pfloat time_interval; /* time interval */
    union {
        struct {
        
    struct {

modified 2 April 1993
FORTRAN Input Parameters

WKID  The workstation identifier of the workstation associated with the device.

SKDNR  The device number of the STROKE device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

IVIEWI  The view index of the view representation in the workstation’s view table to use to map the specified initial stroke points from World Coordinates (WC) to Normalized Projection Coordinates (NPC).

N  The number of coordinates in the initial stroke.

IPX, IPY, IPZ  Arrays of x, y, and z WC values specifying the initial stroke points.

PET  The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

EVOL  The x, y, and z limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, in Device Coordinates (DC).

LDR  The dimension of the data record array.

DATREC  A packed data record, built by PACK DATA RECORD, containing the input data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.
**Execution**

INITIALIZE STROKE 3 sets the initialization parameters of a STROKE device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device will be set to whenever it is enabled. The device's measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A STROKE device measure consists of points and a view index. The points are World Coordinate (WC) points corresponding to the positions on the workstation selected by the operator. The view index is the index of the view representation used to transform the stroke positions from Normalized Projection Coordinates (NPC) to WC. This view representation is determined by selecting the highest priority representation that contains all the stroke positions within its NPC limits. See SET VIEW TRANSFORMATION INPUT PRIORITY for more information. The workstation transform is used to transform the operator-selected positions from Device Coordinates (DC) to NPC.

The initial stroke points are transformed to DC by applying the view orientation and view mapping transforms of the specified view representation, then applying the workstation transformation. If the view index is invalid, an error is generated.

The prompt/echo type determines the display characteristics of the device—that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device's description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in Device Coordinates (DC). Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they do not display there.

The input data record contains the prompt/echo type specific information that controls the device's appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

Stroke devices contain a buffer in which to store the points selected by the operator. Points are added to the buffer starting at the edit position. Both the buffer size and the edit position are specified in the input data record.

All the initialization parameters must be specified properly or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a STROKE input device can be inquired for with the function INQUIRE DEFAULT STROKE DEVICE DATA 3. The current state of the device can be inquired for with the function INQUIRE STROKE DEVICE STATE 3.

**AVAILABLE DEVICES**

- **Device 1** - Cursor and Left Mouse Button
- **Device 2** - Cursor and Middle Mouse Button
- **Device 3** - Cursor and Right Mouse Button

These devices are all associated with the X pointer device, usually the mouse and cursor. The stroke points in the STROKE device’s measure are the WC points corresponding to the positions specified in an operator-generated series of X pointer events. The operator adds points to the buffer by positioning the cursor to the desired location and depressing the appropriate pointer button. Once all the desired points are selected this way the operator presses both the shift key and the appropriate pointer button at the same time. This will trigger the stroke. Points can be removed from the buffer by pressing the CONTROL key and pointer button simultaneously. Only points after the edit position can be removed in this way.

The STROKE echo is removed from the workstation when the cursor leaves the echo volume. Only the x and y components of the echo volume are used. The z component is ignored.

**Prompt/echo types supported:** 1, −3, −4

**PET 1** Display the stroke points by drawing a marker at each stroke position. The marker used is of marker type 2 (plus), size 1.0, and uses colour index 1.

**C Data Record:**

There are no PET-specific data for PET 1; only the buffer_size and init_pos members of the Pstroke_data3 structure are used.

Neither the space nor the time intervals of Pstroke_data3 are currently used by any stroke device.

**FORTRAN Data Record:**

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of integers = 2.
- **IA** Contains two integer values:
  - **IA(1)** The input buffer size.
**IA(2)**  The edit position.

**RL**  The number of real values = 4.

**RA**  An array of reals containing:

**RA(1)**  The x value of the space interval.

**RA(2)**  The y value of the space interval.

**RA(3)**  The z value of the space interval.

**RA(4)**  The time interval.

All these values should be 0; neither the space nor the time intervals are currently used by any stroke device.

**SL**  The number of strings = 0.

**PET −3**  Display the stroke points by drawing a marker at each stroke position. The marker attributes to use are specified in the data record.

**C Data Record:**

A `Pstroke_data3` structure defined in `phigs.h`, the relevant members of which are:

```c
typedef struct {
    Pint       buffer_size; /* input buffer size */
    Pint       init_pos;   /* initial editing position */
    Pfloat     x_interval; /* x interval */
    Pfloat     y_interval; /* y interval */
    Pfloat     z_interval; /* z interval */
    Pfloat     time_interval; /* time interval */
    union {
        struct {
            Pmarker_bundle marker_bundle;
        } pet_u3;
    } pets;
} Pstroke_data3;
```

Neither the space nor the time intervals of `Pstroke_data3` are currently used by any stroke device.

**FORTRAN Data Record:**

The arguments passed to `PACK DATA RECORD` for this prompt/echo type’s data record should be:

**IL**  The number of integers = 4.

**IA**  Contains four integer values:

**IA(1)**  The input buffer size.

**IA(2)**  The edit position.

**IA(3)**  The marker type.

**IA(4)**  The marker colour index.
The number of real values = 5.

An array of reals containing:

RA(1) The x value of the space interval.
RA(2) The y value of the space interval.
RA(3) The z value of the space interval.
RA(4) The time interval.
RA(5) The marker size.

All the interval values should be 0; neither the space nor the time intervals are currently used by any stroke device.

The number of strings = 0.

Display the stroke points by drawing a line between each stroke point. The line attributes to use are specified in the data record.

C Data Record:
A Pstroke_data3 structure defined in phigs.h, the relevant members of which are:

typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    Pfloat x_interval; /* x interval */
    Pfloat y_interval; /* y interval */
    Pfloat z_interval; /* z interval */
    Pfloat time_interval; /* time interval */
    union {
        struct {
            Pline_bundle line_bundle;
        } pet_u4;
    } pets;
} Pstroke_data3;

Neither the space nor the time intervals of Pstroke_data3 are currently used by any stroke device.

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

The number of integers = 4.

Contains four integer values:

IA(1) The input buffer size.
IA(2) The edit position.
IA(3) The line type.
IA(4) The line colour index.
**RL**  The number of real values = 5.

**RA**  An array of reals containing:

- **RA(1)**  The x value of the space interval.
- **RA(2)**  The y value of the space interval.
- **RA(3)**  The z value of the space interval.
- **RA(4)**  The time interval.
- **RA(5)**  The line width.

All the interval values should be 0; neither the space nor the time intervals are currently used by any stroke device.

**SL**  The number of strings = 0.

**ERRORS**

- **003**  Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054**  Ignoring function, the specified workstation is not open
- **061**  Ignoring function, specified workstation’s category is not INPUT or OUTIN
- **114**  Ignoring function, the view index value is less than zero
- **250**  Ignoring function, the specified device is not available on the specified workstation
- **251**  Ignoring function, the function requires the input device to be in REQUEST mode
- **253**  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
- **254**  Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX
- **255**  Ignoring function, one of the echo area/volume boundary points is outside the range of the device
- **260**  Ignoring function, one of the fields within the input device data record is in error
- **253**  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place
- **261**  Ignoring function, initial value is invalid
- **262**  Ignoring function, number of points in the initial stroke is greater than the buffer size
- **114**  Ignoring function, the view index value is less than zero

**SEE ALSO**

- **ESCAPE -19 (3P)**
- **SET STROKE MODE (3P)**
- **REQUEST STROKE 3 (3P)**
- **SAMPLE STROKE 3 (3P)**
- **GET STROKE 3 (3P)**
- **INQUIRE STROKE DEVICE STATE 3 (3P)**

modified 2 April 1993
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
INITIALIZE STROKE (3P)
NAME

INITIALIZE VALUATOR – initialize a valuator input device using 2D data

SYNOPSIS

**C Syntax**

```c
void
pinit_val ( ws, dev, init, pet, echo_area, record )
```

- **ws**: workstation identifier
- **dev**: valuator device number
- **init**: initial value
- **pet**: prompt and echo type
- **echo_area**: echo area pointer
- **record**: data record pointer

**FORTRAN Syntax**

```fortran
SUBROUTINE pinvl ( WKID, VLDNR, IVAL, PET, XMIN, XMAX, YMIN, YMAX,
                   LDR, DATREC )
```

- **WKID**: workstation identifier
- **VLDNR**: valuator device number
- **IVAL**: initial value
- **PET**: prompt/echo type
- **XMIN**, **XMAX**, **YMIN**, **YMAX**: echo area in device coordinates
- **LDR**: dimension of data record array
- **DATREC(LDR)**: data record

FORTRAN Syntax

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

Use INITIALIZE VALUATOR to set the initialization parameters of a VALUATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

**Note**: The specified device must be in REQUEST mode when this function is called.

**C Input Parameters**

- **ws**: The workstation identifier of the workstation associated with the device.
- **dev**: The device number of the VALUATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
- **init**: The initial valuator value.
- **pet**: The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.
- **echo_area**: A pointer to a Plimit structure defining the x and y components of the echo volume, in Device Coordinates (DC). The existing z component in the workstation state list is left unchanged. Plimit is defined in phigs.h as follows:

```c
typedef struct {
```

modified 2 April 1993
Pfloat x_min;  /* minimum x coordinate value */
Pfloat x_max;  /* maximum x coordinate value */
Pfloat y_min;  /* minimum y coordinate value */
Pfloat y_max;  /* maximum y coordinate value */

} Plimit;

record A pointer to a Pval_data structure containing the data record information. The contents of the data record for each device and prompt/echo type are described below in the AVAILABLE DEVICES section.

Some members of the data record correspond to the prompt/echo type being used. As an example, the appropriate member of the Pval_data structure for prompt/echo type 1 is pet_r1.

For some prompt/echo types the data record is not used. The record parameter, however, must still be passed.

Pval_data is defined in phigs.h as:

typedef struct {
Pfloat low;  /* low range limit */
Pfloat high; /* high range limit */
union {
    struct {
Pint unused;
    } pet_r1;
    struct {
Pchar *label;  /* the device label widget string */
Pchar *format; /* the current value widget format string */
Pchar *low_label; /* the low value widget format string */
Pchar *high_label; /* the low value widget format string */
    } pet_u1;
    struct Pval_pet_u4 {
        Phigs_dial_limits dial_limit; /* wraparound or stick at limits */
Pint threshold; /* number of dial events to collapse into 1 */
    } pet_u4;
    } pets;
} Pval_data;

FORTRAN Input Parameters

WKID The workstation identifier of the workstation associated with the device.

VLDNR The device number of the VALUATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.
INITIALIZE VALUATOR allows the initialization parameters of a VALUATOR device to be set. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device is set to whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A VALUATOR device measure consists of a floating point number. This number is between the device’s low value and high value, inclusive.

The prompt/echo type determines the display characteristics of the device — that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in DC. Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they do not display there. This function specifies only the x and y components of the echo volume. The existing z component in the workstation state list is left unchanged.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.
All the initialization parameters must be specified properly, or this function generates an error. The ERRORS section below lists the possible error conditions.

The default initialization parameters and the list of prompt/echo types supported by a VALUATOR input device can be inquired for with the function INQUIRE DEFAULT VALUATOR DEVICE DATA. The current state of the device can be inquired for with the function INQUIRE VALUATOR DEVICE STATE.

### AVAILABLE DEVICES

#### Devices 1 through 10

- **OLIT Slider Widgets**

  The valuator device realizations are each a collection of widgets from the OLIT widget set. Each consists of a Shell widget, a Control Area widget, four Caption widgets, and a Slider widget. The Slider is the item manipulated by the operator to change the valuator value. The four Caption widgets display a label for the device and labels for each of the current value, low value, and high value of the device. The device label, current value, and low value are all displayed (by default) to the left of the Slider. The high value is displayed to the right. The Control Area, Caption, and Slider widgets are children of the Shell widget.

  Each of the four Caption widgets has a character string associated with it that indicates what to display in that widget. The strings for the current value, low value, and high value widgets may contain C style printf(3) format descriptors. SunPHIGS expands these descriptors with the corresponding device value so that the value appears in the widget. For example, if the low value widget string is `the low value is %` and the low value is 5, SunPHIGS displays the widget as `the low value is 5.0`.

  The operator selects a valuator value by positioning the cursor on top of the slider box, depressing the SELECT button, and moving the mouse while holding the button down. If the valuator device is in REQUEST mode, the device triggers when the operator lets up on the button. When the device is in EVENT mode, the device triggers with each movement of the mouse within the valuator (thus generates an event for each mouse movement). In sample mode the device’s measure is changed with each mouse movement made while the SELECT button is depressed.

  The echo volume is not used for these devices. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

### Prompt/echo types supported: 1, −1

**PET 1**

Display the valuator item as specified above. The string for the label widget is `value`. The format string for the current value string is `%8.3f`; for the low value and high value strings, `[%8.3g]`. (Use PET −1 if you do not want these values.)

**C Data Record:**

There are no PET-specific data for PET 1, simply the `low` and `high` members of the Pval_data structure.
FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type's data record should be:

IL  The number of integers = 0.
RL  The number of real values = 2.
RA  An array of reals containing:
   RA(1)  The low value limit of valuator range.
   RA(2)  The high value limit of valuator range.
SL  The number of strings = 0.

PET –1 Display the valuator item as specified above. The label, current value, low value, and high value strings are specified in the data record.

C Data Record:
A Pval_data structure defined in phigs.h, the relevant members of which are:
typedef struct {
    Pfloat  low;  /* low range limit */
    Pfloat  high; /* high range limit */
    union {
        struct {
            char  *label;  /* the device label widget string */
            char  *format; /* the current value widget format string */
            char  *low_label; /* the low value widget format string */
            char  *high_label; /* the low value widget format string */
        } pet_u1;
    } pets;
} Pval_data;

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type's data record should be:

IL  The number of integers = 0.
RL  The number of real values = 2.
RA  An array of reals containing:
   RA(1)  The low value limit of valuator range.
   RA(2)  The high value limit of valuator range.
SL  The number of character strings = 4.
**LSTR** An array containing the lengths of the label and format strings.

- **LSTR(1)** The length of the label string.
- **LSTR(2)** The length of the current value format string.
- **LSTR(3)** The length of the low value format string.
- **LSTR(4)** The length of the high value format string.

**STR** An array containing the length and format strings.

- **STR(1)** The label string.
- **STR(2)** The format string for the current value. Use `%w.df` for fixed point notation, `%w.de` for scientific notation, or the shorter of the two, where `w` is the width of the output field and `d` is the number of decimal places to display.
- **STR(3)** The low value format string.
- **STR(4)** The high value format string.

The widgets used and their names, in decreasing hierarchy, are:

- `valuator1` → `popupWindowShellWidgetClass`
  - `label` → `captionWidgetClass`
  - `readout` → `captionWidgetClass`
  - `low_label` → `captionWidgetClass`
  - `slider` → `sliderWidgetClass`
  - `high_label` → `captionWidgetClass`

Fallback resources for the valuator devices are:

- `valuator1*background` → `grey`
- `valuator1*slider.orientation` → `horizontal`
- `valuator1*slider.width` → `200`

The full qualified name of all widgets is:

```
<appl_name>.workstation<ws_id>.valuator<dev_id>.<widget_name>,
```

where `<appl_name>` is the application name specified in the call to `OPEN XPHIGS`. (This is `phigs` if `OPEN XPHIGS` was not called.)

For example, `phigs.workstation1.valuator1.slider` is the name of the slider widget of valuator device 1 on workstation 1.

**Devices 11 through 18 - SunDials**

These devices correspond to SunDials 1 through 8, if a Sun dialbox is attached.

The operator selects a valuator value by turning the appropriate dial. The trigger for these devices is dial movement. These devices only trigger when the pointer is within the display surface of the associated workstation.
The application can specify the behavior of each device at the limits defined by its low and high values; each device either wraps around or sticks at the appropriate limit.

The **threshold** of each dial in **EVENT** mode can also be specified, defining the number of dial **clicks** that should go by before an event is queued; this allows the collapsing of events to avoid quick filling of the input queue.

These devices have no display components, that is, no prompt and no echo.

**Prompt/echo types supported:** 1, −4

PET 1 Each dial wraps around at the limits defined for it, and the **EVENT** mode threshold is 3. Use PET -4 if these values are not as desired.

**C Data Record:**

There are no PET-specific data for PET -1, simply the **low** and **high** members of the **Pval_data** structure.

**FORTRAN Data Record:**

The arguments passed to **PACK DATA RECORD** for this prompt/echo type’s data record should be:

- **IL** The number of real integers = 0.
- **RL** The number of real values = 2.
- **RA** An array of reals containing:
  - **RA(1)** The low value limit of valuator range.
  - **RA(2)** The high value limit of valuator range.
- **SL** The number of strings = 0.

PET −4 The dial behavior at its defined limits and the **EVENT** mode threshold are specified in the data record.

**C Data Record:**

A **Pval_data** structure defined in **phigs.h**, the relevant members of which are:

```c
typedef struct {
    Pfloat low; /* low range limit */
    Pfloat high; /* high range limit */
    union {
        struct {
            Phigs_dial_limits dial_limits;
            Pint threshold;
        } pet_u4;
    } pets;
} Pval_data;
```

**Phigs_dial_limits** is an enumerated type defined in **phigs.h** as follows:

```c
typedef enum {
    PDIAL_WRAPAROUND,
```
PDIAL_STICK
} Phigs_dial_limits;

**FORTRAN Data Record:**
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data
record should be:

**IL**  The number of integers = 2.

**IA**  An array of integers with the following values:

- **IA(1)**  The dial behavior at its defined limits. Valid values as defined in
  phigs77.h are:
  - PDIALWRAP
  - PDIALSTICK

- **IA(2)**  The EVENT mode threshold.

**RL**  The number of real values = 2.

**RA**  An array of reals containing:

- **RA(1)**  The low value limit of valuator range.
- **RA(2)**  The high value limit of valuator range.

**SL**  The number of strings = 0.

**ERRORS**

- **003**  Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054**  Ignoring function, the specified workstation is not open
- **061**  Ignoring function, specified workstation’s category is not INPUT or OUTIN
- **250**  Ignoring function, the specified device is not available on the specified
  workstation
- **251**  Ignoring function, the function requires the input device to be in REQUEST mode
- **253**  Warning, the specified prompt/echo type is not available on the specified
  workstation. Prompt/echo type one will be used in its place
- **254**  Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN
  > ZMAX
- **255**  Ignoring function, one of the echo area/volume boundary points is outside the
  range of the device
- **260**  Ignoring function, one of the fields within the input device data record is in error
- **261**  Ignoring function, initial value is invalid

**SEE ALSO**

- ESCAPE -19 (3P)
- SET VALUATOR MODE (3P)
- REQUEST VALUATOR (3P)
- SAMPLE VALUATOR (3P)
- GET VALUATOR (3P)
INQUIRE VALUATOR DEVICE STATE (3P)
INITIALIZE VALUATOR 3 (3P)
NAME
INITIALIZE VALUATOR 3 – initialize a valuator input device using 3D data

SYNOPSIS
C Syntax

void pinit_val3 ( ws, dev, init, pet, echo_volume, record )

Pint ws; // workstation identifier
Pint dev; // valuator device number
Pfloat init; // initial value
Pint pet; // prompt and echo type
Plimit3 *echo_volume; // echo volume pointer
Pval_data3 *record; // data record pointer

FORTRAN Syntax

SUBROUTINE pinvl3 ( WKID, VLDNR, IVAL, PET, EVOL, LDR, DATREC )

INTEGER WKID // workstation identifier
INTEGER VLDNR // valuator device number
REAL IVAL // initial value
INTEGER PET // prompt/echo type
REAL EVOL(6) // echo volume (DC)
INTEGER LDR // dimension of data record array
CHARACTER*80 DATREC(LDR) // data record

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose
Use INITIALIZE VALUATOR 3 to set the initialization parameters of a VALUATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device.

Note: The specified device must be in REQUEST mode when this function is called.

C Input Parameters

ws The workstation identifier of the workstation associated with the device.

dev The device number of the VALUATOR device to initialize. See the AVAILABLE DEVICES section below for a description of the available devices.

init The initial valuator value.

pet The prompt/echo type desired. Those supported by each device are listed in the AVAILABLE DEVICES section below.

echo_volume
A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates (DC). Plimit3 is defined in phigs.h as follows:

typedef struct {
Pfloat x_min; /* minimum x coordinate value */
Pfloat x_max; /* maximum x coordinate value */
}
record A pointer to a Pval_data3 structure containing the data record information. The
contents of the data record for each device and prompt/echo type are described
below in the AVAILABLE DEVICES section.

Some members of the data record correspond to the prompt/echo type being
used. As an example, the appropriate member of the Pval_data3 structure for
prompt/echo type 1 is pet_r1.

For some prompt/echo types the data record is not used. The record parameter,
however, must still be passed.

Pval_data3 is defined in phigs.h as:

typedef struct {
  Pfloat low;        /* low range limit */
  Pfloat high;       /* high range limit */
union {
  struct {
    Pint unused;
  } pet_r1;
  struct {
    Pchar *label;  /* the device label widget string */
    Pchar *format; /* the current value widget
                    format string */
    Pchar *low_label; /* the low value label widget
                        format string */
    Pchar *high_label; /* the high value label widget
                        format string */
  } pet_u1;
  struct Pval_pet_u4 {
    Phigs_dial_limits dial_limit; /* wraparound or stick at
                                limits */
    Pint threshold; /* number of dial events to
                     collapse into 1 */
  } pet_u4;
} pets;
Pval_data3;

FORTRAN Input
Parameters

WKID  The workstation identifier of the workstation associated with the device.

VLDNR  The device number of the VALUATOR device to initialize. See the AVAILABLE DEVICES
section below for a description of the available devices.
INITIALIZE VALUATOR 3 sets the initialization parameters of a VALUATOR device. This function stores these parameters in the workstation description table of the workstation associated with the specified device. The specified device must be in REQUEST mode when this function is called.

The parameters that are initialized by this function are initial measure, prompt/echo type, echo volume, and input data record.

The initial measure is the logical input value the device is set to whenever it is enabled. The device’s measure retains this value until operator input changes it. A device is enabled when the appropriate REQUEST function is called, or when its input mode is set to SAMPLE or EVENT.

A VALUATOR device measure consists of a floating point number. This number is between the device’s low value and high value, inclusive.

The prompt/echo type determines the display characteristics of the device, that is, how it is presented to the operator and responds to his actions. Each device supports one or more prompt/echo types. Those supported by each device are listed in the device’s description in the AVAILABLE DEVICES section below. All devices support prompt/echo type 1. Positive prompt/echo types are defined by the PHIGS standard. Negative types are implementation-dependent. Most SunPHIGS input devices support both positive and negative prompt/echo types.

The echo volume defines the region of the display surface in which to echo the device. It is specified in DC. Devices that use the echo volume restrict their display to this region. Some of these devices still recognize operator input outside the region even though they do not display there.

The input data record contains the prompt/echo type specific information that controls the device’s appearance and characteristics. Not all the data record contents are used by some devices. The device descriptions in the AVAILABLE DEVICES section below list the data record contents that each device recognizes.

All the initialization parameters must be specified properly, or this function generates an error. The ERRORS section below lists the possible error conditions.
The default initialization parameters and the list of prompt/echo types supported by a VALUATOR input device can be inquired for with the function INQUIRE DEFAULT VALUATOR DEVICE DATA 3. The current state of the device can be inquired for with the function INQUIRE VALUATOR DEVICE STATE 3.

### AVAILABLE DEVICES

**Devices 1 through 10**

- **OLIT Slider**

**Widgets**

The valuator device realizations are each a collection of widgets from the OLIT widget set. Each consists of a Shell widget, a Control Area widget, four Caption widgets, and a Slider widget. The Slider is the item manipulated by the operator to change the valuator value. The four Caption widgets display a label for the device and labels for each of the current value, low value, and high value of the device. The device label, current value, and low value are all displayed (by default) to the left of the Slider. The high value is displayed to the right. The Control Area, Caption, and Slider widgets are children of the Shell widget.

Each of the four Caption widgets has a character string associated with it that indicates what to display in that widget. The strings for the current value, low value, and high value widgets may contain C style printf(3) format descriptors. SunPHIGS expands these descriptors with the corresponding device value so that the value appears in the widget. For example, if the low value widget string is `the low value is %f` and the low value is 5, SunPHIGS displays the widget as `the low value is 5.0`.

The operator selects a valuator value by positioning the cursor on top of the slider box, depressing the SELECT button, and moving the mouse while holding the button down. If the valuator device is in REQUEST mode, the device triggers when the operator lets up on the button. When the device is in EVENT mode, the device triggers with each movement of the mouse within the valuator (thus generates an event for each mouse movement). In sample mode the device’s measure is changed with each mouse movement made while the SELECT button is depressed.

The echo volume is not used for these devices. Applications or users can specify the position of the devices (subject to window manager control) by specifying the appropriate resource values in a resource file.

**Prompt/echo types supported:** `1, -1`

**PET 1**

Display the valuator item as specified above. The string for the label widget is `value`. The format string for the current value string is `%8.3f`; for the low value and high value strings, `[%8.3g]`. (Use PET-1 if you do not want these values.)

**C Data Record:**

There are no PET-specific data for PET 1, simply the `low` and `high` members of the Pval_data3 structure.

---

modified 2 April 1993
FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL. The number of integers = 0.

RL. The number of real values = 2.

RA. An array of reals containing:
   RA(1) The low value limit of valuator range.
   RA(2) The high value limit of valuator range.

SL. The number of strings = 0.

PET –1 Display the valuator item as specified above. The label, current value, low value, and high value strings are specified in the data record.

C Data Record:
A Pval_data3 structure defined in phigs.h, the relevant members of which are:

typedef struct {
    float low;    /* low range limit */
    float high;   /* high range limit */
    union {
        struct {
            char *label;    /* the device label widget string */
            char *format;  /* the current value widget format string */
            char *low_label; /* the low value widget format string */
            char *high_label; /* the low value widget format string */
        } pet_u1;
    } pets;
} Pval_data3;

FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

IL. The number of integers = 0.

RL. The number of real values = 2.

RA. An array of reals containing:
   RA(1) The low value limit of valuator range.
   RA(2) The high value limit of valuator range.

SL. The number of character strings = 4.

modified 2 April 1993
**LSTR** An array containing the lengths of the label and format strings.

$LSTR(1)$

The length of the label string.

$LSTR(2)$

The length of the current value format string.

$LSTR(3)$

The length of the low value format string.

$LSTR(4)$

The length of the high value format string.

**STR** An array containing the length and format strings.

$STR(1)$ The label string.

$STR(2)$ The format string for the current value. Use $\%w.df$ for fixed point notation, $\%w.de$ for scientific notation, or $\%w.de$ for the shorter of the two, where $w$ is the width of the output field and $d$ is the number of decimal places to display.

$STR(3)$ The low value format string.

$STR(4)$ The high value format string.

The widgets used and their names, in decreasing hierarchy, are:

- **valuator1** → `popupWindowShellWidgetClass`
  - label → `captionWidgetClass`
  - readout → `captionWidgetClass`
  - low_label → `captionWidgetClass`
  - slider → `sliderWidgetClass`
  - high_label → `captionWidgetClass`

Fallback resources for the valuator devices are:

*valuator1*background → grey
*valuator1*slider.orientation → horizontal
*valuator1*slider.width → 200

The full qualified name of all widgets is

```
<appl_name>.workstation<ws_id>.valuator<dev_id>.<widget_name>,
```

where <appl_name> is the application name specified in the call to OPEN XPHIGS. (This is *phigs* if OPEN XPHIGS was not called.)

For example, phigs.workstation1.valuator1.slider is the name of the slider widget of valuator device 1 on workstation 1.

**Devices 11 through 18 - SunDials**

These devices correspond to SunDials 1 through 8, if a Sun dialbox is attached.

The operator selects a valuator value by turning the appropriate dial. The trigger for these devices is dial movement. These devices only trigger when the pointer is within the display surface of the associated workstation.

modified 2 April 1993
The application can specify the behavior of each device at the limits defined by its low and high values. Each device either wraps around or sticks at the appropriate limit. The threshold of each dial in EVENT mode can also be specified, defining the number of dial clicks that should go by before an event is queued. This allows the collapsing of events to avoid quick filling of the input queue.

These devices have no display components, that is, no prompt and no echo.

**Prompt/echo types supported**: 1, −4

**PET 1** Each dial wraps around at the limits defined for it, and the EVENT mode threshold is 3. Use PET -4 if these values are not as desired.

**C Data Record:**

There are no PET-specific data for PET 1, simply the low and high members of the Pval_data3 structure.

**FORTRAN Data Record:**

The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

- **IL** The number of integers = 0.
- **RL** The number of real values = 2.
- **RA** An array of reals containing:
  - **RA(1)** The low value limit of valuator range.
  - **RA(2)** The high value limit of valuator range.
- **RA** The number of strings = 0.

**PET −4** The dial behavior at its defined limits and the EVENT mode threshold are specified in the data record.

**C Data Record:**

A Pval_data3 structure defined in phigs.h, the relevant members of which are:

```c
typedef struct {
    Pfloat low; /* low range limit */
    Pfloat high; /* high range limit */
    union {
        struct {
            Phigs_dial_limits dial_limits;
            Pint threshold;
        } pet_u4;
    } pets;
} Pval_data3;
```

Phigs_dial_limits is an enumerated type defined in phigs.h as follows:

```c
typedef enum {
    PDIAL_WRAPAROUND,
    /* other values */
} Phigs_dial_limits;
```

modified 2 April 1993
FORTRAN Data Record:
The arguments passed to PACK DATA RECORD for this prompt/echo type’s data record should be:

**IL**  The number of integers = 2.

**IA**  An array of integers with the following values:

IA(1)  The dial behavior at its defined limits. Valid values as defined in phigs77.h are:
    PDIALWRAP
    PDIALSTICK

IA(2)  The EVENT mode threshold.

**RL**  The number of real values = 2.

**RA**  An array of reals containing:

RA(1)  The low value limit of valuator range.

RA(2)  The high value limit of valuator range.

**SL**  The number of strings = 0.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)

054  Ignoring function, the specified workstation is not open

061  Ignoring function, specified workstation’s category is not INPUT or OUTIN

250  Ignoring function, the specified device is not available on the specified workstation

251  Ignoring function, the function requires the input device to be in REQUEST mode

253  Warning, the specified prompt/echo type is not available on the specified workstation. Prompt/echo type one will be used in its place

254  Ignoring function, invalid echo area/volume; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX

255  Ignoring function, one of the echo area/volume boundary points is outside the range of the device

260  Ignoring function, one of the fields within the input device data record is in error

261  Ignoring function, initial value is invalid

**SEE ALSO**

ESCAPE -19 (3P)

SET VALUATOR MODE (3P)

REQUEST VALUATOR (3P)

SAMPLE VALUATOR (3P)

GET VALUATOR (3P)
INQUIRE VALUATOR DEVICE STATE 3 (3P)
INITIALIZE VALUATOR (3P)
NAME
INTERPRET ITEM – interpret supplied metafile item data record

SYNOPSIS
C Syntax
void interpret_item ( type, item_data_length, item_data )
  Pint type;
  Pint item_data_length;
  Pitem_data *item_data;

FORTRAN Syntax
SUBROUTINE piitm ( TYPE, NCHS, LDR, DATREC )
  INTEGER TYPE
  INTEGER NCHS
  INTEGER LDR
  CHARACTER*80 DATREC(LDR)

Required PHIGS Operating States (PHOP, *, *, *)

DESCRIPTION
Note: This function has C and FORTRAN bindings, but its functionality is not implemented.

ERRORS
002 Ignoring function, function requires state (PHOP, *, *, *)
301 Ignoring function, item length is invalid
304 Ignoring function, item type is unknown
303 Ignoring function, metafile item is invalid
305 Ignoring function, content of item data record is invalid for the specified item type
307 Ignoring function, user item cannot be interpreted

SEE ALSO
GET ITEM TYPE FROM METAFILE (3P)
READ ITEM FROM METAFILE (3P)
NAME
LABEL – create a structure element containing a label identifier

SYNOPSIS

C Syntax
void
  plabel ( label_id )
  Pint  label_id;  label identifier

FORTRAN Syntax
SUBROUTINE plb ( LABEL )
  INTEGER   LABEL  label identifier

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION

Purpose
The LABEL subroutine puts a structure element containing a label identifier into the currently open structure. Labels are used to mark locations within a structure.

If the current edit mode is INSERT, the LABEL element is inserted into the currently open structure after the element currently pointed to by the element pointer. If the edit mode is REPLACE, the LABEL element replaces the element pointed to. In either case, the element pointer is updated to point to the new LABEL element.

C Input Parameter
label_id The label identifier is used by other subroutines to reference the label element’s location in the structure.

FORTRAN Input Parameter
LABEL The label identifier is used by other subroutines to reference the label element’s location in the structure.

Execution
The LABEL element marks an element position within the structure. When the structure is traversed, LABEL elements are ignored.

LABEL elements are used by the SET ELEMENT POINTER AT LABEL and DELETE ELEMENTS BETWEEN LABELS subroutines to identify locations within a structure. The labels allow you to locate an element or group of elements even if structure editing changes the numbering of elements within the structure. For example, you may wish to bracket a sequence of related elements in a structure with labels.

Label identifiers need not be unique in a structure. The search for a label within a structure is always from the element after the current position of the element pointer toward the end of the structure. Subroutines that refer to labels reference the next occurrence of the specified label within the structure. If the specified label is not found before the end of the structure, an error is generated.

260 modified 2 April 1993
ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET ELEMENT POINTER AT LABEL (3P)
DELETE ELEMENTS BETWEEN LABELS (3P)
INQUIRE ELEMENT POINTER (3P)
ELEMENT SEARCH (3P)
NAME | MESSAGE – display message on workstation

SYNOPSIS | void

C Syntax | pmessage ( ws, msg )
Pint ws; workstation identifier
char *msg; message string

FORTRAN Syntax | SUBROUTINE pmsg ( WKID, MESS )
INTEGER WKID workstation identifier
CHARACTER*(*) MESS message

FORTRAN Subset Syntax | SUBROUTINE pmsgs ( WKID, LSTR, MESS )
INTEGER WKID workstation identifier
INTEGER LSTR length of string (in characters)
CHARACTER*80 MESS message

Required PHIGS Operating States | (PHOP, WSOP, *, *)

DESCRIPTION | MESSAGE displays a message (character string) on the specified workstation. See Execution below for a description of the placement of the message on the workstation.

Purpose | MESSAGE displays a message (character string) on the specified workstation. See Execution below for a description of the placement of the message on the workstation.

C Input Parameters | ws The workstation to send the message to.
msg The null terminated array of char (character string) to display on the workstation.

FORTRAN Input Parameters | WKID The workstation to send the message to.
MESS The array of characters to display on the workstation.

Execution | MESSAGE displays the specified character string on the specified workstation. The location of the message’s display is dependent on the workstation type as follows:

X Tool workstation types
The message is written to a pop-up window that is dismissable by the program operator. The message window is implemented with OLIT widgets. It recognizes the resource database attribute *message.pushpin with a default value of in.

X Drawable workstation types
The message is written to the lower left corner of the workstation’s graphics window, and will be removed upon the next clearing of that window. See UPDATE WORKSTATION and SET DISPLAY UPDATE STATE for a description of when the window is cleared.

CGM output workstation types
A CGM message element is written to the CGM file. The interpretation of this
element is dependent on the subsequent reader of the file.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open

**SEE ALSO**

ANNOTATION TEXT RELATIVE (3P)
GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
TEXT (3P)
WORKSTATION TYPE SET (3P)
NAME
OFFSET ELEMENT POINTER – move element pointer in open structure relative to its
current position

SYNOPSIS
C Syntax
void
poffset_elem_ptr ( ep_offset )

Pint ep_offset; element pointer offset

FORTRAN Syntax
SUBROUTINE posep ( EPO )
INTEGER EPO element pointer offset

Required PHIGS
Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
OFFSET ELEMENT POINTER moves the element pointer in the currently open structure a
specified number of elements from its current position.

C Input Parameter
ep_offset
Specifies the number of structure elements to move the element pointer from its
current position.

FORTRAN Input
Parameter
EPO
Specifies the number of structure elements to move the element pointer from its
current position.

Execution
OFFSET ELEMENT POINTER adds element pointer offset to the value of the element pointer in
the open structure. The offset may be positive or negative. The pointer is set to point to
the resulting element number.

If the resulting element number is less than 0, the element pointer is set to 0.

If the resulting element number is greater than the number of elements in the open
structure, the element pointer is set to the last element in the structure.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
INQUIRE ELEMENT POINTER (3P)
INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
ELEMENT SEARCH (3P)
SET ELEMENT POINTER (3P)
SET ELEMENT POINTER AT LABEL (3P)

modified 2 April 1993
NAME
OPEN ARCHIVE FILE – open specified archive file

SYNOPSIS
C Syntax

void
popen_ar_file ( archive_id, archive_file )

Pint archive_id; // archive identifier
char *archive_file; // archive file name

FORTRAN Syntax

SUBROUTINE poparf ( AFID, ARCFIL )

INTEGER AFID // archive file identifier
INTEGER ARCFIL // archive file name

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION

Purpose
Use OPEN ARCHIVE FILE to open an archive file and associate a specified archive file identifier with the open archive file.

C Input Parameters

archive_id
An integer specifying the identifier to be associated with the open archive file.

archive_file
A character string giving the file name of the archive file. The application should not open this file; PHIGS will do this when it needs to access the file.

Neither the application nor the operator should attempt to interact with this file after passing it to OPEN ARCHIVE FILE.

FORTRAN Input Parameters

AFID
An integer specifying the identifier to be associated with the open archive file.

ARCFIL
The logical unit number (LUN) of the physical archive file to use. This LUN must either be associated with an opened and named file or not be associated with any file. (It cannot be the LUN associated with standard output, for instance.) If the LUN is associated with an open and named file, PHIGS first closes the file and then reopens it when reading from or writing to it. If the logical unit number is not associated with an open file, OPEN ARCHIVE FILE creates a file named fort.lun, where lun is the specified logical unit number.

Neither the application nor the operator should attempt to interact with this logical unit number after passing it to OPEN ARCHIVE FILE.

Execution
If the specified archive file does not exist, then a new file is created and remains empty until CLOSE ARCHIVE FILE is called. At that time, any archived structures will be written to the file.

modified 2 April 1993
If the specified file exists and is a valid SunPHIGS archive file, then the contents are read and available for retrieval. If the named file exists but is empty, then it is treated as if it did not exist (that is, it becomes a valid SunPHIGS archive file and remains empty until it is closed, at which time any archived structures are written to it).

The PHIGS archive state is set to archive open (AROP) and the specified archive file identifier is added to the set of open archive files maintained in the PHIGS state list.

**Note:** It is recommended that the application leave an archive file open until it is no longer needed, in order to avoid repeating the overhead associated with the OPEN ARCHIVE FILE operation.

Two archive file formats are supported: clear text format (defined by the PHIGS standard) and a binary PEX format. The binary format is supported for users who want compact archives over standard conformance. Read-only capability of SunPHIGS 1.x binary archive files is provided for backward compatibility with previous SunPHIGS releases. SunPHIGS writes clear text as the default archive file format. The ESCAPE function can be used to control the type of archive written; see the ESCAPE -15 reference manual page for further information. SunPHIGS reads either clear text or binary archives, as appropriate for how they were written.

**ERRORS**

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **402** Ignoring function, archive file identifier already in use
- **400** Ignoring function, the archive file cannot be opened
- **401** Ignoring function, opening this archive file would exceed the maximum number of simultaneously open archive files
- **403** Ignoring function, the archive file is not a PHIGS archive file
- **409** Warning, archive file is read-only. Contains unsupported PHIGS PLUS elements
- **410** Warning, archive file is read-only. Contains obsolete SunPHIGS elements
- **411** Warning, archive file is read-only. File is a foreign PEX archive file
- **412** Ignoring function, the archive file is read-only

**SEE ALSO**

- ARCHIVE STRUCTURES (3P)
- CLOSE ARCHIVE FILE (3P)
- ESCAPE -15 (3P)
- INQUIRE ARCHIVE FILES (3P)
- INQUIRE ARCHIVE STATE VALUE (3P)
- RETRIEVE STRUCTURE IDENTIFIERS (3P)
- RETRIEVE STRUCTURES (3P)
- DELETE ALL STRUCTURES FROM ARCHIVE (3P)
**NAME**
OPEN PHIGS – open and initialize PHIGS environment

**SYNOPSIS**

**C Syntax**

```c
void popen_phigs ( error_file, memory )
char *error_file; name of error file
size_t memory; NOT USED
```

**FORTRAN Syntax**

```fortran
SUBROUTINE popph ( ERRFIL, BUFA )
INTEGER ERRFIL error message file
INTEGER BUFA amount of memory units
```

**Required PHIGS Operating States**
(PHCL, WSCL, STCL, ARCL)

**DESCRIPTION**

**Purpose**
OPEN PHIGS initializes the PHIGS environment and enables access to the PHIGS functions.
OPEN PHIGS must be called prior to calling any other PHIGS functions.

**C Input Parameters**

- `error_file`
  A pointer to the *error file* to which to log PHIGS error messages. The *error file* can be either a pointer to a valid UNIX file name or a null pointer, for example (char*)0. A null pointer implies that standard error is to be used as the error file. If a file name is specified, PHIGS attempts to access the file for writing. If this attempt fails, OPEN PHIGS fails and the appropriate error is reported to standard error.

  The error file argument passed to OPEN PHIGS is passed to ERROR HANDLING. ERROR HANDLING also passes this argument to ERROR LOGGING. If for some reason ERROR LOGGING cannot access the specified error file, then the error message is written to standard error. ERROR LOGGING appends messages to the error file; it does not truncate the file when OPEN PHIGS is called. If the specified file does not exist, it will be created only if ERROR LOGGING is called.

  ERROR LOGGING writes the abstract PHIGS function name, the error number, and an error description to the error file. If for some reason the text for the function name and/or error description can’t be determined, then ERROR LOGGING simply writes the function number and the error number.

- `memory`
  This parameter should be set to the constant PDEF_MEM_SIZE, which is defined in phigs.h.

modified 2 April 1993
**FORTRAN Input Parameters**

**ERRFIL**

The FORTRAN Logical Unit Number of the error file to which to log PHIGS error messages. Valid logical unit numbers are those that are:

- Associated with the standard preconnected files: standard error and standard output.
- Associated with a file that the application has opened prior to calling OPEN PHIGS.
- Valid with respect to the FORTRAN I/O system (but not those which are not yet associated with an open file).

PHIGS interaction with these various types of logical unit numbers is described in the *Execution* section below.

**BUFA**

This parameter is ignored. −1 should be passed.

**Execution**

OPEN PHIGS performs the following tasks:

- Connects to a default server. (The default server is the one specified by the environment variable DISPLAY.)
- Sets the PHIGS system state variable to PHOP.
- Initializes the default workstation description tables and makes them available for other PHIGS functions to use.
- Creates the PHIGS state list and initializes it with default values taken from the PHIGS description table.
- Stores *error file* in the PHIGS error state list.

For the PHIGS state list, PHIGS description table, and workstation description tables default values, see PHIGS TRAVERSAL STATE LIST (7P), PHIGS DESCRIPTION TABLE (7P), and PHIGS WORKSTATION DESCRIPTION TABLE (7P).

When an error in PHIGS is detected, ERROR HANDLING is called and passed the error file, the function number of the PHIGS function that detected the error, and the error number. ERROR HANDLING calls ERROR LOGGING, which writes an error message to the error file. PHIGS users can replace ERROR HANDLING with a function of their own. This function may in turn call ERROR LOGGING. See ERROR HANDLING and ERROR LOGGING for more information.

PHIGS writes to the error file only if ERROR LOGGING is called by ERROR HANDLER or by the application. If the *error file* does not exist when OPEN PHIGS is called, it is created only if, and when, ERROR LOGGING is called.

**FORTRAN Error Files**

The error file parameter to OPEN PHIGS is a FORTRAN logical unit number. Valid logical unit numbers are those enumerated in the *FORTRAN Parameters* section above. PHIGS interaction with these various types of logical unit numbers is as follows:
- Logical unit numbers associated with the preconnected units standard error and standard output. ERROR LOGGING logs error messages to these files.
- Logical unit numbers associated with files that the application opened prior to calling OPEN PHIGS. ERROR LOGGING logs error messages to these files; PHIGS does not otherwise access them.

**Note:** If it’s desired to append the output of ERROR LOGGING to the file, the file should be opened with the option `fileopt=’eof’` in the FORTRAN open statement.
- Valid logical unit numbers that are not associated with an open file. If ERROR LOGGING is called, a file with the name `fort.lun` is created, where `lun` is replaced with the value of the logical unit number. ERROR LOGGING then logs error messages to the new file. If ERROR LOGGING is not called, then no file is created. If a file with the derived name already exists, then its content is overwritten.

Specifying an invalid logical unit number causes OPEN PHIGS to fail with error number 450: *Ignoring function, the specified error file is invalid*. ERROR LOGGING uses logical unit number 0 (stderr by default) to log this error.

The logical unit number preconnected to standard input (logical unit number 5, by default) cannot be used as the PHIGS error file if it is still associated with standard input when ERROR LOGGING is called. If you wish to use this logical unit number, it must be disassociated from standard input (see the *Sun FORTRAN Programmer’s Guide* for a description of how to do this).

A logical unit number whose value is valid, but which cannot be written to, will cause an additional error message to be logged when ERROR LOGGING is called. This additional error, along with the specified error, is logged to logical unit number 0 (stderr by default) and indicates the FORTRAN error number returned from the FORTRAN write call (that returned to the iostat specifier) and the logical unit number used. If logical unit number 0 cannot be written to in this case, neither error is logged.

The default PHIGS error file is standard error. When the PHIGS system state is PPHCL (PHIGS Closed), invocations of ERROR LOGGING logs errors to this file. An invocation of ERROR LOGGING within a call to OPEN PHIGS is an exception. In this case, the error file specified in the OPEN PHIGS call is used.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Ignoring function, function requires state (PHCL, WSCL, STCL, ARCL)</td>
</tr>
<tr>
<td>450</td>
<td>Ignoring function, the specified error file is invalid</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INQUIRE SYSTEM STATE VALUE (3P)
- CLOSE PHIGS (3P)
- OPEN ARCHIVE FILE (3P)
- OPEN STRUCTURE (3P)
- OPEN WORKSTATION (3P)
- OPEN XPHIGS (3PP)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>OPEN STRUCTURE – create new structure or begin editing existing structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNOPSIS</strong></td>
<td></td>
</tr>
<tr>
<td><strong>C Syntax</strong></td>
<td><code>void popen_struct ( struct_id )</code></td>
</tr>
<tr>
<td></td>
<td><code>Pint struct_id;   structure identifier</code></td>
</tr>
<tr>
<td><strong>FORTRAN Syntax</strong></td>
<td><code>SUBROUTINE popst ( STRID )</code></td>
</tr>
<tr>
<td></td>
<td><code>INTEGER STRID   structure identifier</code></td>
</tr>
<tr>
<td><strong>Required PHIGS Operating States</strong></td>
<td>(PHOP, *, STCL, *)</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Use OPEN STRUCTURE to create a new structure or to begin editing an existing structure. Some structure inquiry functions can only be performed on an open structure.</td>
</tr>
<tr>
<td><strong>C Input Parameter</strong></td>
<td><code>struct_id</code></td>
</tr>
<tr>
<td></td>
<td>The identifier of the structure to open or create.</td>
</tr>
<tr>
<td><strong>FORTRAN Input Parameter</strong></td>
<td><code>STRID</code></td>
</tr>
<tr>
<td></td>
<td>An integer specifying the identifier to be assigned to a new structure or the identifier of an existing structure in the Central Structure Store (CSS) to be opened.</td>
</tr>
<tr>
<td><strong>Execution</strong></td>
<td>If <code>structure identifier</code> does not exist, a new empty structure is created and opened, and the element pointer is set to 0.</td>
</tr>
<tr>
<td></td>
<td>If the structure specified in OPEN STRUCTURE already exists, the structure is opened for editing and the element pointer is set to the last element in the structure.</td>
</tr>
<tr>
<td></td>
<td>A structure consists of a sequence of numbered structure elements. A structure element may contain</td>
</tr>
<tr>
<td></td>
<td>• The specification for an output primitive</td>
</tr>
<tr>
<td></td>
<td>• Attribute selection</td>
</tr>
<tr>
<td></td>
<td>• View selection</td>
</tr>
<tr>
<td></td>
<td>• A modelling transformation</td>
</tr>
<tr>
<td></td>
<td>• A reference to another structure</td>
</tr>
<tr>
<td></td>
<td>• A name set or label</td>
</tr>
<tr>
<td></td>
<td>• A generalized structure element</td>
</tr>
<tr>
<td></td>
<td>• Application data</td>
</tr>
</tbody>
</table>
An element pointer in the open structure is used to reference specific structure elements. When a PHIGS function creates a new element, the current edit mode determines whether the new element is inserted after, or replaces the element pointed to by the current element pointer. You can perform the following operations on the open structure:

- Add or delete structure elements
- Copy all elements from another structure
- Inquire about the current element type and size or the current element content
- Change the position of the element pointer
- Inquire the current position of the element pointer

The PHIGS state list stores the identifier for the currently-open structure, the position of the current element pointer, the current edit mode, and a list of all structure identifiers in use.

**ERRORS**

006 Ignoring function, function requires state (PHOP, *, STCL, *)

**SEE ALSO**

OPEN PHIGS (3P)
INQUIRE OPEN STRUCTURE (3P)
CLOSE STRUCTURE (3P)
INQUIRE STRUCTURE STATE VALUE (3P)
INQUIRE STRUCTURE IDENTIFIERS (3P)
**NAME**
OPEN WORKSTATION – open workstation of specified workstation type

**SYNOPSIS**

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE popwk ( WKID, CONID, WTYPE )
INTEGER WKID    /* workstation identifier */
INTEGER CONID   /* connection identifier */
INTEGER WTYPE   /* workstation type */
```

Required PHIGS Operating States

(PHOP, *, *, * )

**DESCRIPTION**

**Purpose**
OPEN WORKSTATION opens a workstation of the specified workstation type. The workstation state list is initialized to conform as nearly as possible to the workstation type’s workstation description table.

SunPHIGS supports four predefined workstation types, X Tool, X Drawable, X Drawable Region, and CGM Output. Their characteristics are described below. X Tool and X Drawable types cannot be used simultaneously; the decision of which one to use must be made when the application is linked.

If the workstation is opened successfully, a new, specific workstation type is created and associated with the open workstation. This new type contains the actual workstation description table of the workstation, with all fields indicating the actual capabilities of the workstation. This specific type can be retrieved with INQUIRE WORKSTATION CONNECTION AND TYPE.

If the workstation is successfully opened, then the PHIGS workstation state variable is set to WSOP.

**C Input Parameters**

- `ws_id` The workstation identifier to be associated with this workstation. This value is used to identify the workstation in subsequent PHIGS function calls.
- `conn_id` A pointer to the connection identifier of the workstation. The type of value to use depends on the workstation type:
  - X Tool If the `conn_id` is NULL, a window is created on the default server. If the `conn_id` is not NULL, then it is interpreted as a display name and a window for the workstation is created on the named server. An example of using this is:
    ```c
    popen_ws (ws_id, (char*)"unix:0", phigs_ws_type_x_tool);
    ```

272 modified 2 April 1993
The connection identifier must be a pointer to a Pconnid_x_drawable structure, cast to a char*. Pconnid_x_drawable is defined in phigs.h as follows:

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

The connection identifier is a filename. The file should not be open, and the application should not attempt to access the file while the workstation is open. The CGM is incomplete until the workstation is closed, at which time the file is also closed.

The type of workstation to open. Recognized types are described fully in the Available Workstation Types section below. They are declared in phigs.h. A short summary is listed here:

- `phigs_ws_type_x_tool`: PHIGS creates an X window for the workstation on a specified or default server.
- `phigs_ws_type_x_drawable`: PHIGS uses a specified X window for the workstation.
- `phigs_ws_type_x_drawable_region`: PHIGS creates CGM output.

The workstation identifier to be associated with this workstation. This value is used to identify the workstation in subsequent PHIGS function calls.

The connection identifier of the workstation. The type of value to use depends on the workstation type:

- `x_tool`: The connection id contains the address of a null-terminated character string specifying the desired display name (for example, "unix:0"). If the connection id is 0, then a window is created on the default server. A window for the workstation is created on the named server.
- `x_drawable`: The connection identifier should be the address of an array of two integers, the first of which is an X display pointer and the second of which is an X drawable identifier.
**CGM Output**

The logical unit number (LUN) of the file to create. This LUN must be either associated with an opened and named file or not associated with any file. (It cannot be the LUN associated with standard output, for instance.) If the LUN is associated with an open and named file, PHIGS will first close the file and then reopen it if the call to OPEN WORKSTATION is successful. If the logical unit number is not associated with an open file, OPEN WORKSTATION will create a file named *fort.lun*, where *lun* is the specified logical unit number.

Neither the application nor the operator should attempt to interact with this logical unit number while the workstation is open. The file is closed when the workstation is closed. The CGM will be incomplete until the workstation is closed.

**WTYPE** The type of workstation to open. Recognized types are described fully in the *Available Workstation Types* section below. They are defined in phigs77.h, which must be included by a SunPHIGS FORTRAN program. A short summary is listed here:

- **phigswsttool**: PHIGS creates an X window for the workstation on a specified or default server.
- **phigswstdraw**: PHIGS uses a specified X window for the workstation.
- **phigswstdrawregion**: PHIGS uses a specified X window for the workstation.
- **phigswstcgmout**: PHIGS creates CGM output.

**Execution**

OPEN WORKSTATION opens a workstation of the specified type and associates it with the specified workstation identifier.

When a SunPHIGS application opens a workstation, SunPHIGS creates a copy of the workstation type used and binds it to the opened workstation. This copy is called the specific workstation type. The description table values of this specific type are checked for validity with respect to the device’s capabilities. If the values cannot be realized on the device SunPHIGS is using, SunPHIGS will change the description table values of this specific type to the realized values. For example, if 32 colors are requested for a monochrome frame buffer, SunPHIGS will adjust the description table accordingly. The workstation type used in the call to OPEN WORKSTATION is not modified, only the specific workstation type is (potentially) modified. This specific type is the one returned from INQUIRE WORKSTATION CONNECTION AND TYPE.

Many of a workstation type’s workstation description table values can be changed prior to opening a workstation of that type. See WORKSTATION TYPE CREATE and WORKSTATION TYPE SET for more information.
When a workstation is opened, it is running in one of the three modes, depending on the connection identifier:

- **local** The connection identifier specifies a local server.
- **remote PEX** The connection identifier specifies a remote server that supports the PEX extension PHIGS Workstation Only subset. The workstation communicates with the server via PEX 5.0P protocol.
- **remote Xlib** The connection identifier specifies a remote server that does not support the PEX extension PHIGS Workstation Only subset. The workstation communicates with the server via X protocol.

**Available Workstation Types**

**X Tool** PHIGS creates an X window on a specified or default server and uses it for the PHIGS workstation’s display surface. The default category of this workstation type is OUTIN but can be changed to OUTPUT using WORKSTATION TYPE SET.

If the category is OUTIN, then this workstation type provides full PHIGS input support. The behavior and appearance of the X Tool workstation window and input devices are OPEN LOOK compliant. Workstation DC limits correspond to the window size used by the PHIGS workstation when the workstation is opened. The units are drawable coordinates. When PHIGS responds to a window resize event, more or less of the window will be exposed; the PHIGS output will not be scaled. Decreases in size cause portions of the PHIGS output to be clipped away if the new size is less than the PHIGS viewport limits. Size increase beyond the viewport limits will not reveal any additional PHIGS output. For more information, see ESCAPE -3 (3P); and, WORKSTATION TYPE SET (3P), subsections DC Errors and PHIGS DC Model under Workstation Type Attributes.

PHIGS automatically redraws the PHIGS workstation when portions of it are exposed, such as when it is brought to the top of other windows or moved from an iconic state to an open state. This redrawing may potentially change portions of the workstation state list by making the state of visual representation correct and by making all requested entries current.

Many of the characteristics of an X Tool workstation, such as location and color table size, can be set prior to opening it. See WORKSTATION TYPE CREATE and WORKSTATION TYPE SET for a list of the modifiable attributes and their default values.

**X Drawable** PHIGS uses an application-specific X window for the PHIGS workstation’s display surface. The connection identifier parameter specifies the window to use. The window must be open and associated with an X server that supports the PEX extension PHIGS Workstation.
Only subset. The category of this workstation type is OUTPUT; no PHIGS input support is provided.

Workstation DC limits correspond to the size of the application-specific X window when the workstation is opened. The units are drawable coordinates.

The application must handle X window system events, such as resizing and exposure; this workstation type does no window system interaction. For more information, see ESCAPE -8, Raster Resize; and WORKSTATION TYPE SET (3P), subsection PHIGS DC Model under Workstation Type Attributes.

Many of the characteristics of an X Drawable workstation can be set prior to opening it. See WORKSTATION TYPE CREATE and WORKSTATION TYPE SET for a list of the modifiable attributes and their default values.

X Drawable Region

PHIGS uses an application-specific X window for the PHIGS workstation’s display surface. The connection identifier parameter specifies the window to use. The category of this workstation type is OUTPUT; no PHIGS input support is provided.

Workstation DC limits correspond to the size of the application-specific X window when the workstation is opened. The units are drawable coordinates.

The application must handle X window system events, such as resizing and exposure; this workstation type does no window system interaction. For more information, see ESCAPE -8, Raster Resize; and WORKSTATION TYPE SET (3P), subsection PHIGS DC Model under Workstation Type Attributes.

The X Drawable Region workstation type is similar to the X Drawable workstation type, except that it has additional attribute functions (and limits). It allows you to choose independent regions, and several workstations can share a drawable. For more information, see INTRO DRAWABLE REGION (7P).

Many of the characteristics of an X Drawable Region workstation can be set prior to opening it. See WORKSTATION TYPE CREATE and WORKSTATION TYPE SET for a list of the modifiable attributes and their default values.

Note: This workstation type may not be supported when a workstation is running in remote PEX mode.

CGM Output

A Computer Graphics Metafile conforming to the CGM Standard ANSI X3.122-1986. The category of this workstation type is MO. The connection id is the file identification of the CGM file to create.

SunPHIGS creates the CGM file and writes the metafile header and descriptor when the workstation is opened. Graphical CGM elements (primitives and attributes) are generated when PHIGS performs a traversal of the structures posted to the CGM workstation. The default deferral state for CGM Output workstation types is WAIT, NIVE; therefore, traversal is deferred until the application explicitly initiates a traversal (such as by REDRAW ALL STRUCTURES). A new CGM picture element is generated for each PHIGS traversal. BEGIN PICTURE is written at the start of each traversal and END PICTURE is written when each traversal ends.

modified 2 April 1993
The default device space for SunPHIGS CGM workstations is [0,1] in all dimensions. The addressable units of the CGM device are [0, 32767] in the x and y dimensions and 0 in the z dimension. Consequently all SunPHIGS primitives are clipped in 3D and projected to two dimensions prior to writing them to the CGM file. (CGM is a 2D Standard.) Only integer CGM coordinate values are generated and no hidden line or hidden surface removal is applied.

SunPHIGS TEXT elements are written to the CGM as CGM polyline or polygon elements. These elements are the result of rotating the text to the specified text plane, applying the attributes and rendering the text, with clipping; that is, the text is written to the CGM as it would appear on an X Drawable or X Tool workstation.

C binding users specify a file name as the connection identifier; FORTRAN users specify a logical unit number (LUN). When a SunPHIGS CGM workstation is opened, the specified file is truncated if it exists or created if it does not.

The default encoding for the CGM file is binary. This, as well as other characteristics of the workstation, can be changed with WORKSTATION TYPE CREATE and WORKSTATION TYPE SET. See the manual pages for these function for a list of the modifiable attributes and their default values.

ERRORS

002 Ignoring function, function requires state (PPOP, *, *, *)
050 Ignoring function, connection identifier not recognized by the implementation
053 Ignoring function, workstation identifier already is in use
052 Ignoring function, workstation type not recognized by the implementation
055 Ignoring function, workstation cannot be opened for an implementation dependent reason
063 Ignoring function, opening this workstation would exceed the maximum number of simultaneously open workstations
-200 Ignoring function, cannot connect to the designated or default server
-201 Ignoring function, the specified or default X server does not support a compatible PEX extension

SEE ALSO

INQUIRE SET OF OPEN WORKSTATIONS (3P)
POST STRUCTURE (3P)
CLOSE WORKSTATION (3P)
INQUIRE WORKSTATION CONNECTION AND TYPE (3P)
INQUIRE WORKSTATION STATE VALUE (3P)
WORKSTATION TYPE CREATE (3P)
WORKSTATION TYPE SET (3P)
**NAME**

PACK DATA RECORD – pack values from FORTRAN arrays into data record

**SYNOPSIS**

**FORTRAN Syntax**

SUBROUTINE pprec ( IL, IA, RL, RA, SL, LSTR, STR, MLDR, ERRIND, LDR, DATREC )

INTEGER IL 
number of integer entries (>= 0)

INTEGER IA(*)
array containing integer entries

INTEGER RL 
number of real entries (>= 0)

REAL RA(*)
array containing real entries

INTEGER SL 
number of character string entries (>= 0)

INTEGER LSTR(*)
lengths of each character string entry (>= 0)

CHARACTER(*) STR(*)
character string entries

INTEGER MLDR 
dimension of data record array

INTEGER ERRIND 
OUT error indicator (zero if no error)

INTEGER LDR 
OUT number of array elements used in DATREC

CHARACTER*80 DATREC(MLDR) 
OUT data record

**FORTRAN Subset Syntax**

SUBROUTINE pprec ( IL, IA, RL, RA, SL, LSTR, STR, MLDR, ERRIND, LDR, DATREC )

INTEGER IL 
array containing integer entries (>= 0)

INTEGER IA(IL)
array containing integer entries

INTEGER RL 
array containing real entries

REAL RA(RL)
array containing real entries

INTEGER SL 
number of character string entries (>= 0)

INTEGER LSTR(SL)
lengths of each character string entry (>= 0)

CHARACTER*80 STR(*)
character string entries

INTEGER MLDR 
dimension of data record array

INTEGER ERRIND 
OUT error indicator (zero if no error)

INTEGER LDR 
OUT number of array elements used in DATREC

CHARACTER*80 DATREC(MLDR) 
OUT data record

**Required PHIGS Operating States**

(*, *, *, *)

**DESCRIPTION**

**Purpose**

PACK DATA RECORD is a PHIGS FORTRAN utility function. It packs variable or implementation-dependent data values into a data record, to be an input parameter for the fixed argument lists of the FORTRAN binding functions.
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IL</strong></td>
<td>The number of INTEGER entries to pack.</td>
</tr>
<tr>
<td><strong>IA</strong></td>
<td>An array containing the IL INTEGER values to pack. The FORTRAN subset syntax restricts IA to be of dimension IL.</td>
</tr>
<tr>
<td><strong>RL</strong></td>
<td>The number of REAL entries to pack.</td>
</tr>
<tr>
<td><strong>RA</strong></td>
<td>An array containing the RL REAL values to pack. The FORTRAN subset syntax restricts RA to be of dimension RL.</td>
</tr>
<tr>
<td><strong>SL</strong></td>
<td>The number of character string entries to pack.</td>
</tr>
<tr>
<td><strong>LSTR</strong></td>
<td>An array containing the lengths of the SL character strings.</td>
</tr>
<tr>
<td><strong>STR</strong></td>
<td>An array containing the SL character strings to pack. The FORTRAN subset syntax restricts the STR array to be CHARACTER*80. The STR argument is required, even if the number of strings, SL, is 0.</td>
</tr>
<tr>
<td><strong>MLDR</strong></td>
<td>The number of 80-character strings available in DATREC.</td>
</tr>
</tbody>
</table>

### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERRIND</strong></td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td><strong>LDR</strong></td>
<td>The number of 80-character strings used in the DATREC array. If MLDR is less than LDR, LDR returns the number of elements that would be required to pack the data.</td>
</tr>
<tr>
<td><strong>DATREC</strong></td>
<td>An array of 80-character strings into which the data record is packed.</td>
</tr>
</tbody>
</table>

### Execution

PACK DATA RECORD packs multiple INTEGER, REAL, and CHARACTER string values into a single data record, in contiguous 80-character elements of the DATREC array.

The data record may be used as an input parameter to a PHIGS FORTRAN function, along with the INTEGER LDR, used to dimension the array holding the data record. This scheme allows the FORTRAN function to accept variable or implementation-dependent data, in simple fixed-format parameters. The FORTRAN ESCAPE function, `pesc` is an example of a PHIGS FORTRAN function that accepts a data record.

Although the content of the data record is implementation-dependent, UNPACK DATA RECORD extracts the multiple values from the data record back into arrays.

### ERRORS

- **2001** **FORTRAN**: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.
- **2004** **FORTRAN**: Ignoring function, input parameter size out of range — the INTEGER passed as an input parameter defining the size of FORTRAN array parameters is negative or greater than an absolute maximum. (For example, a FORTRAN subset function taking a string as an input parameter can cause this error.)

### SEE ALSO

UNPACK DATA RECORD (3P)

modified 2 April 1993
### NAME
POLYLINE – create structure element specifying 2D polyline

### SYNOPSIS

#### C Syntax

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

#### FORTRAN Syntax

```fortran
SUBROUTINE ppl ( N, PXA, PYA )
INTEGER N number of points
REAL PXA(N), PYA(N) coordinates of points (MC)
```

### Required PHIGS Operating States

(PHOP, *, STOP, *)

### DESCRIPTION

#### Purpose
The POLYLINE function places a structure element containing the full specification of a 2D polyline into the currently-open structure, according to the current edit mode. A 2D polyline primitive is a set of connected straight lines in two dimensions, \(x\) and \(y\). The \(z\) coordinate is assumed to be 0. The polyline is defined by a series of points in Modelling Coordinates (MC). A polyline may be closed or intersect itself.

#### C Input Parameters

- `point_list`  
  A pointer to a Ppoint_list structure containing a list of Ppoint structures, which contain the \(x\) and \(y\) coordinates for each point. At least two points must be specified; a polyline structure element that has fewer than two points will be treated in a workstation-dependent fashion.

The Ppoint_list structure is defined in phigs.h as follows:

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

The `num_points` component specifies the number of elements in the list. The `points` component is a pointer to a list of Ppoints `num_points` long. The Ppoint structure is defined in phigs.h as follows:

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

#### FORTRAN Input Parameters

- `N`  
  The number of points used to define the polyline. You must specify at least two points; a polyline structure element that has fewer than two points will be ignored when the structure is traversed.
**Execution**

If the current edit mode is INSERT, the POLYLINE element is inserted into the open structure after the element pointed to by the element pointer. If the edit mode is REPLACE, the POLYLINE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new POLYLINE element.

When the structure is traversed, the polyline element draws straight lines between the points specified, beginning with the first point and ending with the last. The current values of the attributes listed below will be applied to the polyline when it is drawn.

The coordinates used to specify the polyline primitive are MC. These may be any coordinate units that are convenient to the application. At traversal, these coordinate values will be transformed by the current local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

**Attributes Applied**

The attributes listed below are used to display the POLYLINE primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- polyline colour
- linewidth scale factor
- linetype
- polyline shading method
- polyline index
- depth cue index
- name set

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

INQUIRE POLYLINE FACILITIES (3P)
POLYLINE 3 (3P)
POLYLINE SET 3 WITH DATA (3PP)

---

*PX A  An array of N real values containing the x coordinates of the polyline.
PY A  An array of N real values containing the y coordinates of the polyline.*
NAME
POLYLINE 3 - create structure element specifying 3D polyline

SYNOPSIS
C Syntax
void ppolyline3 ( point_list )
  Ppoint_list3 *point_list;  list of points

FORTRAN Syntax
SUBROUTINE ppl3 ( N, PXA, PYA, PZA )
  INTEGER N  number of points
  REAL PXA(N), PYA(N), PZA(N)  coordinates of points (MC)

Required PHIGS Operating States
( PHOP, *, STOP, * )

DESCRIPTION
Purpose
The POLYLINE 3 function places a structure element containing the full specification of a 3D polyline
into the currently open structure, according to the current edit mode. A 3D polyline primitive is a set of
connected straight lines in 3D space defined by a series of Modelling Coordinate (MC) points. A polyline
may be closed or intersect itself.
If the current edit mode is INSERT, the POLYLINE 3 element is inserted into the open
structure after the element pointed to by the element pointer. If the current edit mode is
REPLACE, the POLYLINE 3 element replaces the element pointed to by the element pointer.
In either case, the element pointer is updated to point to the new POLYLINE 3 element.

C Input Parameters
point_list
A pointer to a Ppoint_list3 structure containing a list of Ppoint3 structures, which
contain the x, y, and z coordinates for each point. At least two points must be
specified; a polyline structure element that has fewer than two points is treated in
a workstation-dependent fashion.
The Ppoint_list3 structure is defined in phigs.h as follows:

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

The num_points component specifies the number of elements in the list. The points component is a pointer to a list of Ppoint3s num_points long. The Ppoint3 structure is defined in phigs.h as follows:

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

modified 2 April 1993
**FORTRAN Input**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>The number of points used to define the polyline. You must specify at least two points; a polyline structure element that has fewer than two points will be ignored when the structure is traversed.</td>
</tr>
<tr>
<td>$PXA$</td>
<td>An array of $N$ real values containing the $x$ coordinates of the polyline.</td>
</tr>
<tr>
<td>$PYA$</td>
<td>An array of $N$ real values containing the $y$ coordinates of the polyline.</td>
</tr>
<tr>
<td>$PZA$</td>
<td>An array of $N$ real values containing the $z$ coordinates of the polyline.</td>
</tr>
</tbody>
</table>

**Execution**

When the structure is traversed, the polyline element draws straight lines between the points specified, beginning with the first point and ending with the last. The current values of the attributes listed below are applied to the polyline when it is drawn.

The coordinates used to specify the polyline primitive in this subroutine are MCs. These coordinates may be any coordinate units that are convenient to the application. At traversal, these coordinate values are transformed by the current local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

**Attributes Applied**

The attributes listed below are used to display the POLYLINE 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>polyline colour</td>
<td>polyline</td>
</tr>
<tr>
<td>linewidth scale factor</td>
<td>linewidth scale factor ASF</td>
</tr>
<tr>
<td>linetype</td>
<td>linetype ASF</td>
</tr>
<tr>
<td>polyline shading method</td>
<td>polyline shading method ASF</td>
</tr>
<tr>
<td>polyline index</td>
<td></td>
</tr>
<tr>
<td>depth cue index</td>
<td></td>
</tr>
<tr>
<td>name set</td>
<td></td>
</tr>
</tbody>
</table>

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

INQUIRE POLYLINE FACILITIES (3P)
POLYLINE (3P)
POLYLINE SET 3 WITH DATA (3PP)
NAME
POLYMARKER – create structure element specifying 2D polymarker primitive

SYNOPSIS
C Syntax
void ppolymarker ( point_list )
    Ppoint_list *point_list;  /* list of points */

FORTRAN Syntax
SUBROUTINE ppm ( N, PXA, PYA )
    INTEGER N  /* number of points */
    REAL PXA(N), PYA(N)  /* coordinates of points (MC) */

DESCRIPTION
Purpose
The POLYMARKER function puts the full specification of a POLYMARKER primitive into the
currently open structure. A POLYMARKER primitive is a series of markers, such as a dot,
plus sign, asterisk, circle, or x, drawn in the display at specified Modelling Coordinate
(MC) points on the z = 0 plane.

If the current edit mode is INSERT, the POLYMARKER element is inserted in the open
structure after the element pointed at by element pointer. If the current edit mode is
REPLACE, the POLYMARKER element replaces the element pointed to by the element
pointer. In either case, the element pointer is updated to point to the new POLYMARKER
element.

C Input Parameters
    point_list
A pointer to a Ppoint_list structure containing a list of Ppoint structures, which
contain the x and y coordinates for each point at which markers will be placed.

The Ppoint_list structure is defined in phigs.h as follows:

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

The num_points component specifies the number of elements in the list. The points
component is a pointer to a list of Ppoints num_points long. The Ppoint structure
is defined in phigs.h as follows:

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

modified 2 April 1993
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>The number of points in the display at which to place markers.</td>
</tr>
<tr>
<td>( PXA )</td>
<td>An array of ( N ) real values containing the ( x ) coordinates of the points at which to place markers.</td>
</tr>
<tr>
<td>( PYA )</td>
<td>An array of ( N ) real values containing the ( y ) coordinates of the points at which to place markers.</td>
</tr>
</tbody>
</table>

### Execution

When the structure is traversed, the `POLYMARKER` element places a marker at each of the coordinate locations specified. The current values of the attributes listed below are applied to the marker.

The coordinates used to specify the polymarker primitive are `MCS`. These may be any coordinate units that are convenient to the application. At traversal, these coordinate values are transformed by the current local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted. More information.

### Attributes Applied

The attributes listed below are used to display the `POLYMARKER` primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- marker type
- marker size scale factor
- polymarker colour
- polymarker index
- depth cue index
- name set

### ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

### SEE ALSO

- `INQUIRE POLYMARKER FACILITIES (3P)`
- `POLYMARKER 3 (3P)`

modified 2 April 1993
**NAME**

POLYMARKER 3 – create structure element specifying 3D polymarker primitive

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE ppm3 ( N, PXA, PYA, PZA )
    INTEGER N  /* number of points */
    REAL PXA(N), PYA(N), PZA(N)  /* coordinates of points (MC) */
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

The POLYMARKER 3 function puts the full specification of a POLYMARKER 3 primitive into the currently open structure. A POLYMARKER 3 primitive is a series of markers, such as dots, plus signs, asterisks, circles, or x’s drawn in the display at the specified Modelling Coordinate (MC) points.

If the current edit mode is INSERT, the POLYMARKER 3 element is inserted in the open structure after the element pointed at by the element pointer. If the current edit mode is REPLACE, the POLYMARKER element replaces the element pointed at by the element pointer. In either case, the element pointer is updated to point to the new POLYMARKER 3 element.

**C Input Parameters**

*point_list*

A pointer to a Ppoint_list3 structure containing a list Ppoint3 structures, which contain the x and y coordinates for each point at which markers will be placed. The Ppoint_list3 structure is defined in phigs.h as follows:

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

The `num_points` component specifies the number of elements in the list. The `points` component is a pointer to a list of Ppoint3s `num_points` long. The Ppoint3 structure is defined in phigs.h as follows:

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

Parameters

\( N \) \quad \text{The number of points in the display at which to place markers.}

\( \textit{PXA} \) \quad \text{An array of} \ N \ \text{real values containing the} \ x \ \text{coordinates of the points at which to place markers.}

\( \textit{PYA} \) \quad \text{An array of} \ N \ \text{real values containing the} \ y \ \text{coordinates of the points at which to place markers.}

\( \textit{PZA} \) \quad \text{An array of} \ N \ \text{real values containing the} \ z \ \text{coordinates of the points at which to place markers.}

Execution

When the structure is traversed, the POLYLINE 3 element places a marker at each of the coordinate locations specified. The current values of the attributes listed below is applied to the marker.

The marker positions are specified in MCS. These coordinates may be any coordinate units that are convenient to the application. At traversal, these coordinate values are transformed by the current local and global modelling transformations, the view representation selected by the current view index, and the workstation transformation current on the workstation to which the structure is posted.

Attributes Applied

The attributes listed below are used to display the POLYMARKER 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- marker type
- marker scale factor
- polymarker colour
- polymarker index
- depth cue index
- name set
- marker type ASF
- marker size factor ASF
- polymarker colour index ASF

ERRORS

005 \quad \text{Ignoring function, function requires state (PHOP, *, STOP, *)}

SEE ALSO

INQUIRE POLYMARKER FACILITIES (3P)
POLYMARKER (3P)
NAME

POST STRUCTURE – assign structure network to workstation for display

SYNOPSIS

C Syntax
void ppost_struct ( ws_id, struct_id, priority )
int ws_id; /* workstation identifier */
int struct_id; /* structure identifier */
float priority; /* priority */

FORTRAN Syntax
SUBROUTINE ppost ( WKID, STRID, PRIORT )
INTEGER WKID /* workstation identifier */
INTEGER STRID /* structure identifier */
REAL PRIORT /* display priority */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION

Purpose
POST STRUCTURE assigns a structure, and the structure network of which it is the root, to a workstation for display. The workstation’s display update state determines when the structure network is traversed, updating the display.

C Input Parameters
ws_id The identifier of the workstation to which the structure should be posted.
struct_id The identifier of the structure to be posted.
priority The display priority of this structure in relation to other structures posted to the same workstation, in the range of 0.0 to 1.0, inclusive.

FORTRAN Input Parameters
WKID The identifier of the workstation to which the structure should be posted.
STRID The identifier of the structure to be posted.
PRIORT The display priority of this structure in relation to other structures posted to the same workstation, in the range of 0.0 to 1.0, inclusive.

Execution
POST STRUCTURE adds structure identifier to the table of posted structures on the workstation with workstation identifier. If the structure does not exist, it is created as an empty structure and posted to ws_id.

If more than one structure network is posted to the workstation, display pri determines the display order of the structure networks. Structures with higher priority values take precedence over (that is, are drawn on top of) structures with lower priority values. If the structure identifier is already posted on the workstation, the structure is removed from the table of posted structures, and reposted with the priority value specified. If two structures are posted with the same priority, the structure posted (or reposted) last.
assumes a higher priority.

If the workstation’s display update state permits, POST STRUCTURE initiates traversal of all the structure networks posted to the workstation, ordered by their priority. (See SET DISPLAY UPDATE STATE for more information on the workstation display update state.)

The structure elements are traversed in sequence, beginning with the first element of the posted structure. Traversal of subordinate structures in the network is prompted by EXECUTE STRUCTURE elements.

The structure network and the workstation state together determine the image displayed on the workstation’s display surface. While a structure remains posted to a workstation, any changes in the structure network or the workstation’s tables can result in immediate changes to the display, depending on the structure’s priority and the workstation’s display update state.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
208 Ignoring function, the display priority is out of range

**SEE ALSO**

UNPOST STRUCTURE (3P)
UNPOST ALL STRUCTURES (3P)
INQUIRE POSTED STRUCTURES (3P)
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED (3P)
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED (3P)
NAME
READ ITEM FROM METAFILE – read current item from metafile

SYNOPSIS

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

FORTRAN Syntax
SUBROUTINE prditm ( WKID, MIDRL, MLDR, DATREC )
INTEGER WKID          // workstation identifier
INTEGER MIDRL        // maximum item data record length
                    // (number of characters in the data record array)
INTEGER MLDR         // dimension of item data record
CHARACTER*80 DATREC(MLDR)   // OUT data record

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Note: This function has C and FORTRAN bindings, but its functionality is not implemented.

ERRORS
003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
058 Ignoring function, specified workstation is not of category MI
306 Ignoring function, maximum item data record length is invalid
302 Ignoring function, no item is left in metafile input
303 Ignoring function, metafile item is invalid

SEE ALSO
OPEN WORKSTATION (3P)
GET ITEM TYPE FROM METAFILE (3P)
INTERPRET ITEM (3P)
NAME
REDRAW ALL STRUCTURES – redisplay all structures posted to workstation

SYNOPSIS
C Syntax
void
predraw_all_structs ( ws, control_flag )

Pint ws; /* workstation identifier */
Pctrl_flag control_flag; /* controls the redraw of the structures */

FORTRAN Syntax
SUBROUTINE prst ( WKID, COFL )

INTEGER WKID /* workstation identifier */
INTEGER COFL /* control flag (PCONDI, PALWAY) */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
REDRAW ALL STRUCTURES updates the specified workstation’s display by:

- Completing any actions in progress
- Clearing the display surface if the surface is NOT EMPTY, or if the control_flag is ALWAYS
- Updating any pending viewing transformation and pending HLHSR mode
- Redisplaying all structures posted to the workstation
- Setting the workstation’s state of visual representation to CORRECT

C Input Parameters
ws The identifier of the workstation whose structures are to be redrawn.
control_flag
Controls whether the display surface is automatically cleared. Pctrl_flag is defined in phigs.h. Valid values are:

typedef enum {
    PFLAG_COND,
    PFLAG_ALWAYS
}Pctrl_flag;

PFLAG_COND clears the display surface before redrawing only if the surface is not empty.
PFLAG_ALWAYS automatically clears the surface before redrawing.

FORTRAN Input Parameters
WKID The identifier of the workstation whose structures are to be redrawn.
COFL Controls whether the display surface is automatically cleared or not. Valid values are defined in phigs77.h:

PCONDI Conditionally
PALWAY  

Always

PCONDI clears the display surface before redrawing only if the surface is not empty.
PALWAY automatically clears the surface before redrawing.

**Execution**

REDRAW ALL STRUCTURES performs the following actions in sequence:

1. Any buffered communications for the specified workstation are completed without first clearing the display surface.

2. If the control_flag is set to CONDITIONALLY, then the workstation’s display surface is only cleared if it is NOT EMPTY. If the control flag is set to ALWAYS, the display surface is cleared regardless of whether the display is EMPTY. In any case, the entry is then set to EMPTY.

   **Note:** Using a control flag of CONDITIONALLY avoids unnecessarily clearing an empty display surface. A value of ALWAYS is useful when generating multiple copies of the current image on hardcopy devices.

3. For every view representation in the workstation’s state list, if the view transformation update state is PENDING, the current view representation is loaded from the requested view representation, and the update state is set to NOTPENDING.

4. If the workstation transformation update state is PENDING, the current workstation window, and current workstation viewport are loaded with the requested values for each, and the update state is set to NOTPENDING.

5. If the workstation’s HLHSR update state is PENDING, the current HLHSR mode is updated to the value of the requested HLHSR mode; and the update state is set to NOTPENDING.

6. All structure networks posted for this workstation are redisplayed in their priority order.

7. The workstation’s state of visual representation is set to CORRECT.

The workstation’s state of visual representation indicates if the display is CORRECT, SIMULATED, or DEFERRED. INQUIRE DISPLAY UPDATE STATE returns the workstation’s display surface empty and state of visual representation state list entries. Normally, the traversal in Step 6 causes the display surface empty workstation state to become NOTEMPTY. (If all the posted structures are empty, or contain elements but no output primitive elements, the display surface empty may be EMPTY or NOTEMPTY.)
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- UPDATE WORKSTATION (3P)
- INQUIRE DISPLAY UPDATE STATE (3P)
- SET DISPLAY UPDATE STATE (3P)
### NAME

REMOVE NAMES FROM SET – create structure element to remove names from current name set

### SYNOPSIS

**C Syntax**

```c
void premove_names_set ( set )
Pint_list *set;  set of names to be removed
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pres ( N, NAMSET )
INTEGER N  number of names in the set
INTEGER NAMSET(N)  name set
```

### Required PHIGS Operating States

(HPH, *, STOP, *)

### DESCRIPTION

#### Purpose

REMOVE NAMES FROM SET creates a structure element containing names to be removed from the traversal-time current name set, and puts the element into the currently-open structure according to the current edit mode. The current name set is compared during traversal to the workstation’s name set filters to determine if primitives which follow in the structure network are invisible, highlighted, and/or selectable by PICK input devices. Each name in the name set is a small non-negative integer.

If the current edit mode is INSERT, the REMOVE NAMES FROM SET element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the REMOVE NAMES FROM SET element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

#### C Input Parameter

*set*  A pointer to a Pint_list structure containing the set of names to be removed. The Pint_list data structure is defined in phigs.h as:

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

#### FORTRAN Input Parameters

*N*  The number of names to be removed.

*NAMSET*  An array containing the set of *N* names to be removed.

#### Execution

When traversal of a posted structure network starts, the current name set is empty. During traversal, the member names specified by ADD NAMES TO SET elements are added to the current name set. The member names specified by REMOVE NAMES FROM SET elements are removed from the current name set.

294 modified 2 April 1993
The updated current name set applies to primitives that follow in the structure network by set-intersection with the workstation’s name set filters, which are set by SET INVISIBILITY FILTER, SET HIGHLIGHTING FILTER, and SET PICK FILTER. Each workstation has a single invisibility filter, a single highlighting filter, and a pick filter for each PICK input device. The actual appearance of highlighting is workstation-dependent.

Each filter contains an inclusion set and an exclusion set of names, both empty by default. A primitive is eligible if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. If the current name set is empty, subsequent primitives are not eligible. If the workstation’s inclusion set is empty (the default), no primitives are eligible. That is, no primitives are invisible, highlighted, or selectable by PICK input devices.

Each name is a small non-negative integer. PHIGS conformance requires support for at least 64 names; SunPHIGS supports the range 0 to 1023. The same names may be added and removed any number of times during traversal. Since the presence or absence of each name in the current name set and the workstation’s filter affect the eligibility of subsequent primitives, 1024 names provide up to 1024 different simultaneous groupings of primitives.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ADD NAMES TO SET (3P)
SET INVISIBILITY FILTER (3P)
SET HIGHLIGHTING FILTER (3P)
SET PICK FILTER (3P)
INCREMENTAL SPATIAL SEARCH (3P)
INCREMENTAL SPATIAL SEARCH 3 (3P)
NAME
REQUEST CHOICE – request operator interaction with specified choice device

SYNOPSIS
C Syntax

```c
void preq_choice ( ws, dev, in_status, choice )
Pint ws;
Pint dev;
Pin_status *in_status;
Pint *choice;
```

FORTRAN Syntax

```
SUBROUTINE prqch ( WKID, CHDNR, STAT, CHNR )
INTEGER WKID
INTEGER CHDNR
INTEGER STAT
INTEGER CHNR
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use REQUEST CHOICE to request an operator interaction with a specified choice device. The device must be in request mode. See SET CHOICE MODE and INITIALIZE CHOICE for more information. See the AVAILABLE DEVICES section of INITIALIZE CHOICE for a description of the available choice devices.

The request suspends PHIGS until the specified device is triggered, or the operator performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys simultaneously ('D').

C Input Parameters

- **ws** Workstation identifier. An integer specifying the workstation with which the requested choice device is associated.
- **dev** The device number of the requested choice device.

C Output Parameters

- **in_status** A pointer to the location to store the status of the request. Pin_status is defined in phigs.h as follows:

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

- **choice** A pointer to the location to store the selected choice value, if any.

296 modified 2 April 1993
### FORTRAN Input Parameters

**WKID**  
The workstation identifier of the workstation associated with the device.

**CHDNR**  
The device number of the CHOICE device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE for a description of the available devices.

### FORTRAN Output Parameters

**STAT**  
The measure’s request choice status. Valid values as defined in phigs77.h are:
- PNONE Break
- POK OK
- PNCHOI No choice

**CHNR**  
The measure’s choice number. This value is defined only if the status returned is OK.

### Execution

REQUEST CHOICE requests a logical input value from the specified choice device. The specified device must be in request mode. See SET CHOICE MODE for more information.

REQUEST CHOICE creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the CONTROL and D keys simultaneously (‘D’).

If the choice device is triggered and a choice has not been made, the status is returned as NO CHOICE, and no choice value is returned.

If the choice device is triggered and a choice has been made, the status is returned as OK, and the choice value contains the choice number selected by the operator.

If a BREAK occurs, the status is returned as NONE, and no choice value is returned.

In either case, when the device is triggered or a BREAK is issued, the current measure process is terminated and PHIGS processing resumes.

### ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation is not of category INPUT or OUTIN
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode

### SEE ALSO

INITIALIZE CHOICE 3 (3P)  
SET CHOICE MODE (3P)
NAME
REQUEST LOCATOR – request operator interaction with specified locator device

SYNOPSIS
void
preq_loc ( ws, dev, in_status, view_ind, loc_pos )

Pint ws;          workstation identifier
Pint dev;         locator device number
Pin_status *in_status; OUT input status
Pint *view_ind;   OUT view index
Ppoint *loc_pos;  OUT locator position

FORTRAN Syntax
SUBROUTINE prqlc ( WKID, LCDNR, STAT, VIEWI, PX, PY )

INTEGER WKID       workstation identifier
INTEGER LCDNR      locator device number
INTEGER STAT       OUT status (PNONE, POK)
INTEGER VIEWI      OUT view index
REAL PX, PY        OUT locator position

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use REQUEST LOCATOR to request an operator interaction with a specified locator device.
The device must be in request mode. See SET LOCATOR MODE and INITIALIZE LOCATOR for
more information. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a
description of the available locator devices.
The request suspends PHIGS until the specified device is triggered, or the operator
performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys
simultaneously (’D).

C Input Parameters
ws  Workstation identifier. An integer specifying the workstation with which the
     requested locator device is associated.
dev  The device number of the requested locator device. See the AVAILABLE DEVICES
     section of INITIALIZE LOCATOR for a description of the available locator devices.

C Output Parameters
in_status
     A pointer to the location to store the status of the request. Pin_status
typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;

modified 2 April 1993
view_ind
A pointer to the location to store the view representation used to map from Device Coordinates (DC) to World Coordinates (WC).

loc_pos A pointer to a Ppoint structure in which the system returns the locator position coordinates. Ppoint is defined in phigs.h as follows:

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

FORTRAN Input Parameters

WKID The workstation identifier of the workstation associated with the device.

LCDNR The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

FORTRAN Output Parameters

STAT The request status. Valid values as defined in phigs77.h are:

    PNONE   Break
    POK     OK

VIEWI The view index used to map the locator to World Coordinates (WC). This value is defined only if the status returned is OK.

PX, PY The WC locator position. This value is defined only if the status returned is OK.

Execution

REQUEST LOCATOR requests a logical input value from the specified locator device. The specified device must be in request mode. See SET LOCATOR MODE for more information.

REQUEST LOCATOR creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the Control and D keys simultaneously ('D').

If a BREAK occurs, a status of NONE is returned and the locator data is undefined. If a status of OK is returned, the locator data is available. In either case, the measure process is then terminated and PHIGS processing resumes.

A locator measure consists of a view index and WC position. The view index specifies the view representation from the workstation’s view table that was used to map the locator position from Normalized Projections Coordinates (NPC) to WC. The position is the position of the locator in WC.

Note: The 2D and 3D locator measure processes are the same except that the 2D process discards the z coordinate. The 3D version of this function, REQUEST LOCATOR 3, can be used if the value of the z coordinate is needed.

See INITIALIZE LOCATOR for a description of the available locator devices.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation is not of category INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
251 Ignoring function, the function requires the input device to be in REQUEST mode

SEE ALSO

INITIALIZE LOCATOR (3P)
SET LOCATOR MODE (3P)
REQUEST LOCATOR 3 (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
### NAME
REQUEST LOCATOR 3 – request operator interaction with specified locator device

### SYNOPSIS

#### C Syntax
```c
void preq_loc3 ( ws, dev, in_status, view_ind, loc_pos )
```
- `Pint ws;` \textit{workstation identifier}
- `Pint dev;` \textit{locator device number}
- `Pin\_status \*in\_status;` \textit{OUT input status}
- `Pint \*view\_ind;` \textit{OUT view index}
- `Ppoint3 \*loc\_pos;` \textit{OUT locator position}

#### FORTRAN Syntax
```fortran
SUBROUTINE prqlc3 ( WKID, LCDNR, STAT, VIEWI, PX, PY, PZ )
```
- `INTEGER WKID` \textit{workstation identifier}
- `INTEGER LCDNR` \textit{locator device number}
- `INTEGER STAT` \textit{OUT status (PNONE, POK)}
- `INTEGER VIEWI` \textit{OUT view index}
- `REAL PX, PY, PZ` \textit{OUT locator position}

### Required PHIGS Operating States
(PHOP, WSOP, *, *)

### DESCRIPTION
#### Purpose
Use REQUEST LOCATOR 3 to request an operator interaction with a specified locator device. The device must be in request mode. See SET LOCATOR MODE and INITIALIZE LOCATOR 3 for more information. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available locator devices.

The request suspends PHIGS until the specified device is triggered, or the operator performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys simultaneously ('D').

### C Input Parameters
- `ws` \textit{Workstation identifier. An integer specifying the workstation with which the requested locator device is associated.}
- `dev` \textit{The device number of the requested locator device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available locator devices.}

### C Output Parameters
- `in_status` \textit{A pointer to the location to store the status of the request. Pin\_status is defined in phigs.h as follows:}
  ```c
  typedef enum {
    PIN\_STATUS\_NONE,
    PIN\_STATUS\_OK,
    PIN\_STATUS\_NO\_IN
  } Pin\_status;
  ```
**REQUEST LOCATOR 3 (3P)**

A pointer to the location to store the view representation used to map from Device Coordinates (DC) to World Coordinates (WC).

**loc_pos**  
A pointer to a Ppoint3 structure in which the system stores the locator position coordinates. Ppoint3 is defined in phigs.h as follows:

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

**FORTRAN Input Parameters**

- **WKID**  
The workstation identifier of the workstation associated with the device.

- **LCDNR**  
The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

**FORTRAN Output Parameters**

- **STAT**  
The request status. Valid values as defined in phigs77.h are:
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNONE</td>
<td>Break</td>
</tr>
<tr>
<td>POK</td>
<td>OK</td>
</tr>
</tbody>
</table>

- **VIEWI**  
The view index used to map the locator to World Coordinates (WC). This value is defined only if the status returned is OK.

- **PX, PY, PZ**  
The WC locator position. This value is defined only if the status returned is OK.

**Execution**

REQUEST LOCATOR 3 requests a logical input value from the specified locator device. The specified device must be in request mode. See SET LOCATOR MODE for more information. REQUEST LOCATOR 3 creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the Control and D keys simultaneously (\'CD\').

If a BREAK occurs, a status of NONE is returned and the locator data is undefined. If a status of OK is returned, the locator data is available. In either case, the measure process is then terminated and PHIGS processing resumes.

A locator measure consists of a view index and WC position. The view index specifies the view representation from the workstation's view table that was used to map the locator position from Normalized Projections Coordinates (NPC) to WC. The position is the position of the locator in WC.

**Note:** The 2D and 3D locator measure processes are the same except that the 2D process discards the z coordinate. The 2D version of this function, REQUEST LOCATOR, can be used if the value of the z coordinate is not needed.

See INITIALIZE LOCATOR 3 for a description of the available locator devices.
ERRORS
003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation is not of category INPUT or OUTIN
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode

SEE ALSO
INITIALIZE LOCATOR 3 (3P)
SET LOCATOR MODE (3P)
REQUEST LOCATOR (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
NAME
REQUEST PICK – request operator interaction with specified pick device

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

FORTRAN Syntax
SUBROUTINE prqpk ( WKID, PKDNR, IPPD, STAT, PPD, PP )
INTEGER WKID /* workstation identifier */
INTEGER PKDNR /* pick device number */
INTEGER IPPD /* depth of pick path to return */
INTEGER STAT /* OUT status (PNONE, POK, PNPICK) */
INTEGER PPD /* OUT depth of actual pick path */
INTEGER PP(3, IPPD) /* OUT pick path */

DESCRIPTION
Purpose
Use REQUEST PICK to request an operator interaction with a specified pick device. The
device must be in request mode. See SET PICK MODE and INITIALIZE PICK for more
information. See the AVAILABLE DEVICES section of INITIALIZE PICK for a description of the
available pick devices.
The request suspends PHIGS until the specified device is triggered, or the operator
performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys
simultaneously (‘D').

C Input Parameters
ws          Workstation identifier. An integer specifying the workstation with which the
            requested pick device is associated.
dev         The device number of the requested pick device.
depth       The maximum depth of the pick path to return.

C Output Parameters
in_status   A pointer to the location to store the status of the request. Pin_status is defined
            in phigs.h as follows:
            typedef enum {
                PIN_STATUS_NONE,
                PIN_STATUS_OK,
                PIN_STATUS_NO_IN
            }
} Pin_status;

rpick  A pointer to a Ppick_path structure in which
the system stores the pick path.
Ppick_path is defined in phigs.h as:

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

depth  indicates the number of elements in the measure’s path. This is the depth
value contained in the device’s current measure. It is not affected by the
maximum depth to return parameter. Thus, the number of elements
returned in path_list may be less than depth.

path_list  is the array of path elements defining the location of the picked primitive
in the Central Structure Store (CSS).

Note: This array must be allocated by the calling program and the array
pointer assigned to this field before calling this function.
The array must be at least of length depth. Ppick_path_elem is defined in
phigs.h as:

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

struct_id, pick_id, and elem_pos
are the structure identifier, pick identifier, and element number,
respectively, of each element in the path.

FORTRAN Input Parameters

WKID  The workstation identifier of the workstation associated with the device.

PKDNR  The device number of the PICK device. See the AVAILABLE DEVICES section of
INITIALIZE PICK for a description of the available devices.

IPPD  The maximum number of path elements to return.

FORTRAN Output Parameters

STAT  The measure’s request pick status. Valid values as defined in phigs77.h are:

    PNONE    Break
    POK      OK
    PN PICK   No Pick

PPD  The actual pick path depth. This value is defined only if the status returned is OK.
This is the depth value contained in the measure and is not affected by the

modified 2 April 1993
REQUEST PICK (3P)  SunPHIGS Release 3.0

maximum depth to return parameter, IPPD. Thus the number of elements returned in PP may be less than PPD.

PP  An array in which to store the measure’s pick path. The contents of this array are undefined if the status returned is NPICK. This is the two-dimensional array of path elements defining the location of the picked primitive in the Central Structure Store (CSS). Each row of the array contains the structure identifier, pick identifier, and element number, respectively, of each element in the path. The array must be at least of dimension (3, IPPD).

Execution  REQUEST PICK requests a logical input value from the specified pick device. The specified device must be in request mode. See SET PICK MODE for more information.

REQUEST PICK creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the CONTROL and D keys simultaneously ("D").

If the pick device is triggered and a pick has not been made, the status is returned as NO PICK, and no pick data is returned. If the pick device is triggered and a pick has been made, the status is returned as OK, and the pick data is available in the output parameters. If a BREAK occurs, the status is returned as NONE, and no pick value is returned.

In all cases, when the device is triggered or a BREAK is issued, the current measure process is terminated and PHIGS processing resumes.

See INITIALIZE PICK 3 for a description of the available PICK devices and how their measure values are determined.

ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
060  Ignoring function, specified workstation is not of category OUTIN
250  Ignoring function, the specified device is not available on the specified workstation
251  Ignoring function, the function requires the input device to be in REQUEST mode

SEE ALSO

INITIALIZE PICK 3 (3P)
SET PICK FILTER (3P)
SET PICK IDENTIFIER (3P)
SET PICK MODE (3P)
NAME
REQUEST STRING – request operator interaction with specified string device

SYNOPSIS

C Syntax

```c
void
preq_string ( ws, dev, in_status, string )
```

Pint ws;  // workstation identifier
Pint dev;  // string device number
Pin_status *in_status;  // OUT input string
char *string;  // OUT string

FORTRAN Syntax

```fortran
SUBROUTINE prqst ( WKID, STDNR, STAT, LOSTR, STR )
```

INTEGER WKID  // workstation identifier
INTEGER STDNR  // string device number
INTEGER STAT  // OUT status (PNONE, POK)
INTEGER LOSTR  // OUT number of characters returned
CHARACTER*80 STR  // OUT character string

FORTRAN Subset Syntax

```fortran
SUBROUTINE prqst ( WKID, STDNR, STAT, LOSTR, STR )
```

INTEGER WKID  // workstation identifier
INTEGER STDNR  // string device number
INTEGER STAT  // OUT status (PNONE, POK)
INTEGER LOSTR  // OUT number of characters returned
CHARACTER*80 STR  // OUT character string

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use REQUEST STRING to request an operator interaction with a specified string device. The device must be in request mode. See SET STRING MODE and INITIALIZE STRING for more information. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available string devices.

The request suspends PHIGS until the specified device is triggered, or the operator performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys simultaneously (‘D’).

C Input Parameters

ws  // Workstation identifier. An integer specifying the workstation with which the requested string device is associated.

`dev`  // The device number of the requested string device. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available string devices.

C Output Parameter

`in_status`  // A pointer to the location to store the status of the request. Pin_status is defined in phigs.h as follows:

modified 2 April 1993
typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;

string A pointer to an array in which to store the string. The array must be allocated by
the caller and should be at least as large as the buffer size of the device.

FORTRAN Input Parameters

WKID The workstation identifier of the workstation associated with the device.

STDNR The device number of the STRING device. See the AVAILABLE DEVICES section of
INITIALIZE STRING 3 for a description of the available devices.

FORTRAN Output Parameters

STAT The request status. Valid values as defined in phigs77.h are:

    PNONE   Break
    POK      OK

LOSTR The length of the string.

STR The character array into which to copy the string. The array must be allocated by
the caller and should be at least as large as the buffer size of the device.

Execution REQUEST STRING requests a logical input value from the specified string device. The
specified device must be in request mode. See SET STRING MODE for more information.
REQUEST STRING creates a measure process for the specified device and suspends PHIGS
until the device is triggered or the operator issues a BREAK. A BREAK is generated by the
operator by pressing the CONTROL and D keys simultaneously ("D").

If a BREAK occurs, a status of NONE is returned and the string data is undefined. If a status
of OK is returned the string data is available. In either case the measure process is then
terminated, and PHIGS processing resumes.

See INITIALIZE STRING for a description of the available string devices.

ERRORS 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation is not of category INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified
workstation
251 Ignoring function, the function requires the input device to be in REQUEST mode

SEE ALSO INITIALIZE STRING (3P)
SET STRING MODE (3P)
NAME
REQUEST STROKE – request operator interaction with specified stroke device

SYNOPSIS
void
preq_stroke ( ws, dev, in_status, view_ind, stroke )

Pint   ws;   workstation identifier
Pint   dev;   stroke device number
Pin_status *in_status;   OUT input status
Pint   *view_ind;   OUT view index
Ppoint_list *stroke;   OUT stroke

FORTRAN Syntax
SUBROUTINE prqsk ( WKID, SKDNR, N, STAT, VIEWI, NP, PXA, PYA )
INTEGER WKID   workstation identifier
INTEGER SKDNR   stroke device number
INTEGER N   maximum number of points
INTEGER STAT   OUT status (PNONE, POK)
INTEGER VIEWI   OUT view index
INTEGER NP   OUT number of points
REAL PXA(N), PYA(N)   OUT points in stroke (WC)

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use REQUEST STROKE to request an operator interaction with a specified stroke device. The
device must be in request mode. See SET STROKE MODE and INITIALIZE STROKE for more
information. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the
available stroke devices.

The request suspends PHIGS until the specified device is triggered, or the operator
performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys
simultaneously (‘D).

C Input Parameters
ws   Workstation identifier. An integer specifying the workstation with which the
     requested stroke device is associated.

dev   The device number of the requested stroke device. See the AVAILABLE DEVICES
     section of INITIALIZE STROKE for a description of the available stroke devices.

C Output Parameter
in_status
A pointer to the location to store the status of the request. Pin_status is defined
in phigs.h as follows:

typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
}
pin_status;  

view_ind
A pointer to the location to store the view index used to transform the Device Coordinate (DC) positions to World Coordinate (WC) points.

stroke
A pointer to the location to store the stroke points. Ppoint_list is defined in phigs.h as follows:

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

num_points is the number of points in points. points is the array of Ppoint structures specifying the points in WC.

**Note:** This array must be allocated by the calling program and the array pointer assigned to this field before calling this function. The array must be at least as large as the buffer of the STROKE device. This buffer size is set when the device is initialized. Ppoint is defined in phigs.h as follows:

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

**FORTRAN Input Parameters**

- **WKID** The workstation identifier of the workstation associated with the device.
- **SKDNR** The device number of the STROKE device. See the *AVAILABLE DEVICES* section of INITIALIZE STROKE for a description of the available devices.
- **N** The maximum number of stroke points to return.

**FORTRAN Output Parameters**

- **STAT** The request status. Valid values as defined in phigs77.h are:
  - PNONE Break
  - POK   OK

- **VIEWI** The view index used to map the stroke points to WC. This value is defined only if the status returned is OK.
- **NP** The actual number of stroke points.
- **PXA, PYA** The arrays of WC stroke positions. These are only used if the status returned is OK.
REQUEST STROKE requests a logical input value from the specified stroke device. The specified device must be in request mode. See SET STROKE MODE for more information. REQUEST STROKE creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the CONTROL and D keys simultaneously ("D").

If a BREAK occurs, a status of NONE is returned and the stroke data is undefined. If a status of OK is returned the stroke data is available. In either case the measure process is then terminated, and PHIGS processing resumes.

A stroke measure consists of a view index and WC positions. The view index specifies the view representation from the workstation’s view table that was used to map the stroke positions from Normalized Projections Coordinates (NPC) to WC. The positions are the positions of the stroke in WC.

Note: The 2D and 3D stroke measure processes are the same except that the 2D process discards the z coordinates. The 3D version of this function, REQUEST STROKE 3, can be used if the z coordinates are needed.

See INITIALIZE STROKE for a description of the available stroke devices.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
060 Ignoring function, the specified workstation is not of category OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
251 Ignoring function, the function requires the input device to be in REQUEST mode

SEE ALSO

INITIALIZE STROKE (3P)
SET STROKE MODE (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
REQUEST STROKE 3 (3P)
NAME REQUEST STROKE 3 – request operator interaction with specified stroke device

SYNOPSIS
C Syntax

```c
void preq_stroke3 ( ws, dev, in_status, view_ind, stroke )
```

- `Pint ws;` workstation identifier
- `Pint dev;` stroke device number
- `Pin_status *in_status OUT input status`
- `Pint *view_ind; OUT view index`
- `Ppoint_list3 *stroke; OUT stroke`

FORTRAN Syntax

```fortran
SUBROUTINE prqsk3 ( WKID, SKDNR, N, STAT, VIEWI, NP, PXA, PYA, PZA )
```

- `INTEGER WKID` workstation identifier
- `INTEGER SKDNR` stroke device number
- `INTEGER N` maximum number of points
- `INTEGER STAT` OUT status (PNONE, POK)
- `INTEGER VIEWI` OUT view index
- `INTEGER NP` OUT number of points
- `REAL PXA(N), PYA(N), PZA(N)` OUT points in stroke (WC)

Required PHIGS Operating States (PHOP, WSOP, *, *)

DESCRIPTION
Purpose

Use REQUEST STROKE 3 to request an operator interaction with a specified stroke device. The device must be in request mode. See SET STROKE MODE and INITIALIZE STROKE 3 for more information. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available stroke devices.

The request suspends PHIGS until the specified device is triggered, or the operator performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys simultaneously ('D').

C Input Parameters

- `ws` Workstation identifier. An integer specifying the workstation with which the requested stroke device is associated.
- `dev` The device number of the requested stroke device. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available stroke devices.

C Output Parameter

- `in_status` A pointer to the location to store the request status. Pin_status is defined in phigs.h as follows:

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

view_ind
A pointer to the location to store the view index used to transform the Device
Coordinate (DC) positions to World Coordinate (WC) points.

stroke
A pointer to the location to store the stroke points. Ppoint_list3 is defined in
phig.h as follows:

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

num_points
is the number of points in points.
points
is the array of Ppoint3 structures specifying the points in WC.

Note: This array must be allocated by the calling program and the array
pointer assigned to this field before calling this function. The array must
be at least as large as the buffer of the STROKE device. This buffer size is
set when the device is initialized. Ppoint3 is defined in phigs.h as
follows:

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

FORTRAN Input
Parameters

WKID The workstation identifier of the workstation associated with the device.

SKDNR The device number of the STROKE device. See the AVAILABLE DEVICES
section of INITIALIZE STROKE 3 for a description of the available devices.

N The maximum number of stroke points to return.

FORTRAN Output
Parameters

STAT The request status. Valid values as defined in phigs77.h are:

    PNONE    Break
    POK      OK

VIEWI The view index used to map the stroke points to WC. This value is defined only if
the status returned is OK.

NP The actual number of stroke points.

PX, PY, PZ
The arrays of WC stroke positions. These are only used if the status returned is
OK.
REQUEST STROKE 3 requests a logical input value from the specified stroke device. The specified device must be in request mode. See SET STROKE MODE for more information.

REQUEST STROKE 3 creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the CONTROL and D keys simultaneously ("^D").

If a BREAK occurs, a status of NONE is returned and the stroke data is undefined. If a status of OK is returned the stroke data is available. In either case the measure process is then terminated, and PHIGS processing resumes.

A stroke measure consists of a view index and WC positions. The view index specifies the view representation from the workstation's view table that was used to map the stroke positions from Normalized Projections Coordinates (NPC) to WC. The positions are the positions of the stroke in WC.

**Note:** The 2D and 3D stroke measure processes are the same except that the 2D process discards the z coordinates. The 2D version of this function, REQUEST STROKE, can be used if the z coordinates are not needed.

See INITIALIZE STROKE 3 for a description of the available stroke devices.

### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>251</td>
<td>Ignoring function, the function requires the input device to be in REQUEST mode</td>
</tr>
</tbody>
</table>

### SEE ALSO

- INITIALIZE STROKE 3 (3P)
- SET STROKE MODE (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
- REQUEST STROKE (3P)
NAME

REQUEST VALUATOR – request operator interaction with specified valuator device

SYNOPSIS

C Syntax

```c
void preq_val ( ws, dev, in_status, value )

Pint ws;  // workstation identifier
Pint dev;  // valuator device number
Pin_status *in_status;  // OUT input status
Pfloat *value;  // OUT value
```

FORTRAN Syntax

```fortran
SUBROUTINE prqvl ( WKID, VLDNR, STAT, VAL )

INTEGER WKID  // workstation identifier
INTEGER VLDNR  // valuator device number
INTEGER STAT  // OUT status (PNONE, POK)
REAL VAL  // OUT value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use REQUEST VALUATOR to request an operator interaction with a specified valuator device. The device must be in request mode. See SET VALUATOR MODE and INITIALIZE VALUATOR for more information. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available valuator devices.

The request suspends PHIGS until the specified device is triggered, or the operator performs a BREAK. A BREAK is performed by pressing the CONTROL and D keys simultaneously (‘D’).

C Input Parameters

- `ws` Workstation identifier. An integer specifying the workstation with which the requested valuator device is associated.
- `dev` The device number of the requested valuator device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available valuator devices.

C Output Parameter

- `in_status` A pointer to the location to store the request status. Pin_status is defined in phigs.h as follows:
  ```c
  typedef enum {
      PIN_STATUS_NONE,
      PIN_STATUS_OK,
      PIN_STATUS_NO_IN
  } Pin_status;
  ```
- `value` A pointer to the location to store the valuator value.

modified 2 April 1993
REQUEST VALUATOR (3P)  

FORTRAN Input Parameters  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WKID</strong></td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td><strong>VLDNR</strong></td>
<td>The device number of the VALUATOR device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR 3 for a description of the available devices.</td>
</tr>
</tbody>
</table>

FORTRAN Output Parameters  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STAT</strong></td>
<td>The request status. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PNONE Break</td>
</tr>
<tr>
<td></td>
<td>POK OK</td>
</tr>
<tr>
<td><strong>VAL</strong></td>
<td>The valuator’s value.</td>
</tr>
</tbody>
</table>

Execution  
REQUEST VALUATOR requests a logical input value from the specified valuator device. The specified device must be in request mode. See SET VALUATOR MODE for more information.  
REQUEST VALUATOR creates a measure process for the specified device and suspends PHIGS until the device is triggered or the operator issues a BREAK. A BREAK is generated by the operator by pressing the CONTROL and D keys simultaneously ("D").  
If a BREAK occurs, a status of NONE is returned and the valuator data is undefined. If a status of OK is returned, then the valuator data is available. In either case, the measure process is then terminated and PHIGS processing resumes.  
See INITIALIZE VALUATOR for a description of the available valuator devices.  

ERRORS  
003 Ignoring function, function requires state (PHOP, WSOP, *, *)  
054 Ignoring function, the specified workstation is not open  
061 Ignoring function, specified workstation is not of category INPUT or OUTIN  
250 Ignoring function, the specified device is not available on the specified workstation  
251 Ignoring function, the function requires the input device to be in REQUEST mode  

SEE ALSO  
INITIALIZE VALUATOR (3P)  
SET VALUATOR MODE (3P)
**NAME**

RESTORE MODELLING CLIPPING VOLUME – create structure element to restore modelling clipping volume

**SYNOPSIS**

<table>
<thead>
<tr>
<th>C Syntax</th>
<th>FORTRAN Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>void prestore_model_clip_vol ( )</td>
<td>SUBROUTINE prmcv</td>
</tr>
</tbody>
</table>

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

RESTORE MODELLING CLIPPING VOLUME creates a structure element to restore the modelling clipping volume, undoing the effect of SET MODELLING CLIPPING VOLUME and SET MODELLING CLIPPING VOLUME 3 elements in this structure.

**Execution**

If the current edit mode is INSERT, a RESTORE MODELLING CLIPPING VOLUME element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new RESTORE MODELLING CLIPPING VOLUME element.

During traversal, a RESTORE MODELLING CLIPPING VOLUME element resets the current modelling clipping volume to the value of that attribute inherited by the structure being traversed. The attribute may have been modified by traversal of SET MODELLING CLIPPING VOLUME and/or SET MODELLING CLIPPING VOLUME 3 elements.

Modelling clipping to the volume controlled by these three elements is enabled and disabled by SET MODELLING CLIPPING INDICATOR.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

INQUIRE MODELLING CLIPPING FACILITIES (3P)
SET MODELLING CLIPPING INDICATOR (3P)
SET MODELLING CLIPPING VOLUME (3P)
SET MODELLING CLIPPING VOLUME 3 (3P)

Modified 2 April 1993
<table>
<thead>
<tr>
<th>NAME</th>
<th>RETRIEVE ALL STRUCTURES – retrieve all structures from specified archive file</th>
</tr>
</thead>
</table>
| SYNOPSIS | **C Syntax**  
void pret_all_structs ( archive_id )  
Pint archive_id; archive identifier |
|        | **FORTRAN Syntax**  
SUBROUTINE prast ( AFID )  
INTEGER AFID archive file identifier |
| Required PHIGS Operating States | (PHOP, *, *, AROP) |
| DESCRIPTION | **Purpose**  
Use RETRIEVE ALL STRUCTURES to retrieve all structures from the specified open archive file into the Central Structure Store (CSS). |
| C Input Parameter | archive_id  
The archive identifier specifying the open archive file from which to read. |
| FORTRAN Input Parameter | AFID The archive identifier specifying the open archive file from which to retrieve structures. |
| Execution | All structures in the specified open archive file are copied into the CSS. If any of the structures to be retrieved currently exist in the CSS, then the conflict is resolved as follows:  
- When the retrieval conflict resolution flag is MAINTAIN, the conflicting structure will not be copied into the CSS (its contents are maintained).  
- When the retrieval conflict resolution flag is UPDATE, the conflicting structure in the CSS will be overwritten (its contents are updated).  
- When the retrieval conflict resolution flag is ABANDON, no structures will be copied to the CSS at all.  
The retrieval conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine.  
If the currently open structure must be overwritten because of a conflict while the retrieval conflict resolution flag is UPDATE, then it is closed, its contents are overwritten, and it is re-opened. The element pointer will be set to point to the last element. |
| ERRORS | 007 Ignoring function, function requires state (PHOP, *, *, AROP)  
404 Ignoring function, the specified archive file is not open  
405 Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON |
SEE ALSO

- OPEN ARCHIVE FILE (3P)
- ARCHIVE ALL STRUCTURES (3P)
- SET CONFLICT RESOLUTION (3P)
- INQUIRE ALL CONFLICTING STRUCTURES (3P)
- RETRIEVE STRUCTURE IDENTIFIERS (3P)
- RETRIEVE STRUCTURE NETWORKS (3P)
- RETRIEVE STRUCTURES (3P)
- DELETE ALL STRUCTURES FROM ARCHIVE (3P)

modified 2 April 1993
### NAME
RETRIEVE PATHS TO ANCESTORS – retrieve ancestors of specified structure from archive file

### SYNOPSIS
**C Syntax**
```c
void pret_paths_ances ( ar_id, struct_id, po, pd, store, paths, status )
```
- `Pint ar_id;` archive identifier
- `Pint struct_id;` structure identifier
- `Ppath_order po;` path order
- `Pint pd;` path depth
- `Pstore store;` handle to store object
- `Pelem_ref_list_list **paths;` OUT structure path list
- `Pint *status;` OUT status of retrieval

**FORTRAN Syntax**
```fortran
SUBROUTINE prepan ( AFID, STRID, PTHORD, PTHDEP, IPTHSZ, N, OL, APTHSZ, PATHS )
```
- `INTEGER AFID` archive file identifier
- `INTEGER STRID` structure identifier
- `INTEGER PTHORD` path order (PPOTOP, PPOBOT)
- `INTEGER PTHDEP` path depth
- `INTEGER IPTHSZ` size of path buffer
- `INTEGER N` element of list of paths
- `INTEGER OL` OUT number of paths available
- `INTEGER APTHSZ` OUT actual size of the Nth structure path
- `INTEGER PATHS(2, IPTHSZ)` OUT Nth structure path

### Required PHIGS Operating States
(PHOP, *, *, AROP)

### DESCRIPTION
**Purpose**
Use RETRIEVE PATHS TO ANCESTORS to determine the path or paths in the specified archive file which reference the specified structure.

**C Input Parameters**
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by `paths`.

- `ar_id` Identifier of the archive file from which to retrieve structure path data.
- `struct_id` Identifier of the structure whose ancestors are to be retrieved.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `po`      | Order in which the path lists are to be returned. Valid values defined in phigs.h are:  
  ```c
  PORDER_TOP_FIRST  
PORDER_BOTTOM_FIRST  
  ```  
| `pd`      | Depth (maximum number of references) of path lists to return. |
| `store`   | The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)). |
| `paths`   | A pointer to a structure `Pelem_ref_list_list`, containing a list of the specified structure’s structure path lists. `Pelem_ref_list_list` is defined in phigs.h as follows:  
  ```c
  typedef struct{
      Pint num_elem_ref_lists;  
      /∗ number of execute reference lists ∗/
      Pelem_ref_list *elem_ref_lists;  
      /∗ list of execute reference lists ∗/
  } Pelem_ref_list_list;
  ```  
  The `num_elem_ref_lists` component specifies the number of structure path lists, or execute reference lists.  
  The `elem_ref_lists` component is a pointer to a list, `num_elem_ref_lists` long, of `Pelem_ref_list` structures, containing the specified structure’s execute reference lists. `Pelem_ref_list` is defined in phigs.h as follows:  
  ```c
  typedef struct{
      Pint num_elemRefs;  
      /∗ number of execute references ∗/
      Pelem_ref *elemRefs;  
      /∗ list of execute references ∗/
  } Pelem_ref_list;
  ```  
| `elem_refs` | is a pointer to a list `num_elem_refs` long of `Pelem_ref` structures containing the structure identifier and element number of each execute reference structure element in the execute reference list. `Pelem_ref` is defined in phigs.h as follows:  
  ```c
  typedef struct {  
      Pint struct_id;  
      /∗ structure identifier ∗/
      Pint elem_pos;  
      /∗ element number ∗/
  } Pelem_ref;
  ```  
| `status`   | An integer used for returning the success or failure of the routine. |
FORTRAN Input

**Parameters**

- **AFID**: Identifier of the archive file from which to retrieve structure path data.
- **STRID**: Identifier of the structure whose ancestors are to be retrieved.
- **PTHORD**: Order in which the structure path is to be returned. Valid values defined in phigs77.h are:
  - PPOTOP: Top First
  - PPBOBOT: Bottom First
- **PTHDEP**: Depth (maximum number of references) of the structure path to return.
- **IPTHSZ**: Size of the `PATHS` array in which the returned structure path data will be stored. If this value is smaller than the actual size of the structure path (`APTHSZ`), no data is returned in the `PATHS` array, but `APTHSZ` is set to indicate the array size required. If you call this function with an array size of zero, `APTHSZ` is returned with the required array size. Error 2001 is generated if `IPTHSZ` is too small, but not if it is zero.
- **N**: List element of structure paths list to return; only one structure path may be retrieved per subroutine call. If a value of 0 is used here, no structure path data is returned, but the total number of structure paths is returned in `OL`.

FORTRAN Output

**Parameters**

- **OL**: The total number of structure paths for this structure identifier.
- **APTHSZ**: The number of structure path elements returned in `PATHS`.
- **PATHS**: A $2 \times IPTHSZ$ integer array containing the $N$th structure path for the specified structure, where the (1,*) components contain the structure identifiers, and the (2,*) components contain the element sequence numbers.

Execution

When `RETRIEVE PATHS TO ANCESTORS` is called, `structure path list` is filled with list(s) identifying the `EXECUTE STRUCTURE` structure elements which refer to `structure identifier` in the order of traversal. These `EXECUTE STRUCTURE` structure elements are represented as (structure identifier, element position) pairs, giving the parent structure identifier and the position of the `EXECUTE STRUCTURE` structure element. Whenever `structure identifier` itself is included at the bottom of a returned path of ancestors, it is represented by a (structure identifier, element position) pair with an element position of 0. `path order` and `path depth` are used to determine the portion of each path returned. The number of references returned in each path is specified by `path depth`; a `path depth` of 0 returns all the references in the path. In case of truncation, `path order` determines whether the head (top first) or the tail (bottom first) portion of a path is returned. If a path truncation results in two or more partial paths with the same set of element references, only one of the identical path portions is returned.

For example, specifying `top first` and a depth of 0 returns all paths to `structure identifier`. Specifying `top first` and a depth of 1 returns the root of each structure network that references `structure identifier`. Specifying `bottom first` and a depth of 2 returns all the parents of `structure identifier`. 

modified 2 April 1993
### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>Ignoring function, function requires state (PHOP, *, *, AROP)</td>
</tr>
<tr>
<td>201</td>
<td>Ignoring function, the specified structure does not exist</td>
</tr>
<tr>
<td>207</td>
<td>Ignoring function, the specified path depth is less than zero</td>
</tr>
</tbody>
</table>

### SEE ALSO

- OPEN ARCHIVE FILE (3P)
- ARCHIVE STRUCTURE NETWORKS (3P)
- SET CONFLICT RESOLUTION (3P)
- INQUIRE ALL CONFLICTING STRUCTURES (3P)
- INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)
- RETRIEVE PATHS TO DESCENDANTS (3P)
- RETRIEVE STRUCTURE NETWORKS (3P)
- DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
NAME
RETRIEVE PATHS TO DESCENDANTS – retrieve descendants of specified structure from archive file

SYNOPSIS
C Syntax
void
pret_paths_descs ( ar_id, struct_id, po, pd, store, paths, status )

Pint ar_id;         // archive identifier
Pint struct_id;    // structure identifier
Ppath_order po;    // path order
Pint pd;           // path depth
Pstore store;      // handle to Store object
Pelem_ref_list_list **paths; // OUT path list
Pint *status;      // OUT status of retrieval

FORTRAN Syntax
SUBROUTINE prepde ( AFID, STRID, PTHORD, PTHDEP, IPTHSZ, N, OL, APTHSZ, PATHS )
INTEGER AFID          // archive file identifier
INTEGER STRID         // structure identifier
INTEGER PTHORD        // path order (PPOTOP, PPOBOT)
INTEGER PTHDEP        // path depth
INTEGER IPTHSZ        // size of path buffer
INTEGER N             // element of list of paths
INTEGER OL            // OUT number of paths available
INTEGER APTHSZ        // OUT actual size of the Nth structure path
INTEGER PATHS(2, IPTHSZ) // OUT Nth structure path

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use RETRIEVE PATHS TO DESCENDANTS to determine the path or paths in the specified archive file that are referenced by the specified structure.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by paths.

ar_id    Identifier of the archive file from which to retrieve structure path data.
struct_id Identifier of the structure whose descendants are to be retrieved.

modified 2 April 1993
### po
Order in which the path lists are to be returned. Valid values defined in phigs.h are:

- `PORDER_TOP_FIRST`
- `PORDER_BOTTOM_FIRST`

### pd
Depth (maximum number of references) of path lists to return.

### store
The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paths</td>
<td>A pointer to a structure, Pelem_ref_list_list, containing a list of the specified structure’s structure path lists. Pelem_ref_list_list is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef struct</td>
</tr>
<tr>
<td></td>
<td>Pint num_elem_ref_lists; /* number of execute reference lists */</td>
</tr>
<tr>
<td></td>
<td>Pelem_ref_list <em>elem_ref_lists; /</em> list of execute reference lists */</td>
</tr>
<tr>
<td></td>
<td>} Pelem_ref_list_list;</td>
</tr>
<tr>
<td></td>
<td>The <code>num_elem_ref_lists</code> component specifies the number of structure path lists, or execute reference lists.</td>
</tr>
<tr>
<td></td>
<td>The <code>elem_ref_lists</code> component is a pointer to a list, <code>num_elem_ref_lists</code> long, of Pelem_ref_list structures, containing the specified structure’s execute reference lists. Pelem_ref_list is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef struct</td>
</tr>
<tr>
<td></td>
<td>Pint num_elem_refs; /* number of execute references */</td>
</tr>
<tr>
<td></td>
<td>Pelem_ref <em>elem_refs; /</em> list of execute references */</td>
</tr>
<tr>
<td></td>
<td>} Pelem_ref_list;</td>
</tr>
<tr>
<td>elem_ref_lists</td>
<td>is a pointer to a list <code>num_elem_refs</code> long of Pelem_ref structures containing the structure identifier and element number of each execute reference structure element in the execute reference list. Pelem_ref is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef struct {</td>
</tr>
<tr>
<td></td>
<td>Pint struct_id; /* structure identifier */</td>
</tr>
<tr>
<td></td>
<td>Pint elem_pos; /* element number */</td>
</tr>
<tr>
<td></td>
<td>} Pelem_ref;</td>
</tr>
<tr>
<td>status</td>
<td>An integer used for returning the success or failure of the routine.</td>
</tr>
</tbody>
</table>
FORTRAN Input

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFID</td>
<td>Identifier of the archive file from which to retrieve structure path data.</td>
</tr>
<tr>
<td>STRID</td>
<td>Identifier of the structure whose descendants are to be retrieved.</td>
</tr>
<tr>
<td>PTHORD</td>
<td>Order in which the structure path is to be returned. Valid values defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PPOTOP</td>
</tr>
<tr>
<td></td>
<td>PPOBOT</td>
</tr>
<tr>
<td>PTHDEP</td>
<td>Depth (maximum number of references) of the structure path to return.</td>
</tr>
<tr>
<td>IPTHSZ</td>
<td>Size of the PATHS array in which the returned structure path data is stored. If this value is smaller than the actual size of the structure path (APTHSZ), no data is returned in the PATHS array, but APTHSZ is set to indicate the array size required. If you call this function with an array size of zero, APTHSZ is returned with the required array size. Error 2001 is generated if IPTHSZ is too small, but not if it is zero.</td>
</tr>
<tr>
<td>N</td>
<td>List element of structure paths list to return; only one structure path may be retrieved per subroutine call. If a value of 0 is used here, no structure path data will be returned, but the total number of structure paths is returned in OL.</td>
</tr>
</tbody>
</table>

FORTRAN Output

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OL</td>
<td>The total number of structure paths for this structure identifier.</td>
</tr>
<tr>
<td>APTHSZ</td>
<td>The number of structure path elements returned in PATHS.</td>
</tr>
<tr>
<td>PATHS</td>
<td>A $2 \times$ IPTHSZ integer array containing the Nth structure path for the specified structure, where the (1,<em>) components contain the structure identifiers, and the (2,</em>) components contain the element sequence numbers.</td>
</tr>
</tbody>
</table>

Execution

When RETRIEVE PATHS TO DESCENDANTS is called, structure path list is filled with list(s) identifying the EXECUTE STRUCTURE structure elements which are referenced by structure identifier, in the order of traversal. These EXECUTE STRUCTURE structure elements are represented as (structure identifier, element position) pairs, giving the parent structure identifier and the position of the EXECUTE STRUCTURE structure element. The bottom-most element of a structure network, if included in a returned path, is indicated by a (structure identifier, element position) pair containing the identifier of the bottom-most structure and an element position of 0. path order and path depth are used to determine the portion of each path returned. The number of references returned in each path is specified by path depth; a path depth of 0 returns all the references in the path. In case of truncation, path order determines whether the head (top first) or the tail (bottom first) portion of a path is returned. If a path truncation results in two or more partial paths with the same set of element references, only one of the identical path portions is returned.

For example, specifying top first and a depth of 0 returns all paths from structure identifier. Specifying top first and a depth of 1 returns each EXECUTE STRUCTURE structure element in structure identifier as a separate path list. Specifying bottom first and a depth of 1 returns all the bottom-most structures of the network.
ERRORS

007 Ignoring function, function requires state (PHOP, *, *, AROP)
201 Ignoring function, the specified structure does not exist
207 Ignoring function, the specified path depth is less than zero

SEE ALSO

OPEN ARCHIVE FILE (3P)
ARCHIVE STRUCTURE NETWORKS (3P)
SET CONFLICT RESOLUTION (3P)
INQUIRE ALL CONFLICTING STRUCTURES (3P)
INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)
RETRIEVE PATHS TO ANCESTORS (3P)
RETRIEVE STRUCTURE NETWORKS (3P)
DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
NAME
RETRIEVE STRUCTURE IDENTIFIERS – retrieve list of structure identifiers from specified archive file

SYNOPSIS
C Syntax

```c
void pret_struct_ids ( archive_id, max_ids, start, ids, actual_ids )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>archive_id</td>
<td>archive identifier</td>
</tr>
<tr>
<td>max_ids</td>
<td>size of appl. archive id list</td>
</tr>
<tr>
<td>start</td>
<td>start position of ids</td>
</tr>
<tr>
<td>*ids</td>
<td>OUT list of structure ids</td>
</tr>
<tr>
<td>*actual_ids</td>
<td>OUT actual number of ids in PHIGS</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

```fortran
SUBROUTINE prsid ( AFID, ILSIZE, N, LSTRID )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFID</td>
<td>archive file identifier</td>
</tr>
<tr>
<td>ILSIZE</td>
<td>size of the list (LSTRID)</td>
</tr>
<tr>
<td>N</td>
<td>OUT number of structure identifiers in the list</td>
</tr>
<tr>
<td>LSTRID(*)</td>
<td>OUT list of structure identifiers</td>
</tr>
</tbody>
</table>

Required PHIGS Operating States

(PHOP, *, *, AROP)

DESCRIPTION

Purpose
Use RETRIEVE STRUCTURE IDENTIFIERS to retrieve a list of the identifiers of all structures archived in the specified open archive file.

C Input Parameters

- `archive_id`
  The archive identifier specifying the open archive file to read from.

- `max_ids`
  The number of `ints` items in the `ids` output parameter for which the application has allocated memory. `max_ids` is the number of list elements (archive structure identifiers) that the system can return in `ints`. When a value of 0 is used here, no data is returned in the `ints` list, but the total number of structure identifiers in the archive file is returned in `actual_ids`.

- `start`
  Starting position in the list of archive structure identifiers at which to begin the inquiry. The elements of the list of structure identifiers, beginning with the item number specified by `start`, are sequentially copied into `ints` until `i` is full or all the structure identifiers have been copied.

C Output Parameters

- `ids`
  A pointer to a `Pint_list` structure in which the list of structure identifiers in the specified open archive file is returned. The `Pint_list` structure is defined in `phigs.h` as follows:

```c
typedef struct {
```

modified 2 April 1993
Pint num_ints; /* number of Pints in list */
Pint *ints; /* list of integers */
} Pint_list;

The num_ints component specifies the number of structure identifiers in the list. The ints component is a pointer to a list, num_ints long, of the structure identifiers. The application must allocate memory for max_ids elements in the ints list.

actual_ids
A pointer to an integer in which the total number of structure identifiers in the specified archive file is returned. This is the value required for max_ids if all structure identifiers are to be returned.

FORTRAN Input Parameters

AFID The archive identifier specifying the open archive file from which to read.

ILSIZE The size of the LSTRID array. ILSIZE is the number of list elements (archive structure identifiers) that the system can return in LSTRID. When this value is smaller than the actual number of archive structure identifiers (N), no data is returned in the LSTRID array, but N is set to indicate the array size required. If you call this function with an array size of zero, N is returned with the required array size. Error 2001 is returned if ILSIZE is too small, but not if it is zero.

FORTRAN Output Parameters

N The number of archive structure identifiers returned in the LSTRID array.

LSTRID An array of integers in which the list of structure identifiers in the specified archive file is returned.

ERRORS 007 Ignoring function, function requires state (PHOP, *, *, AROP)
404 Ignoring function, the specified archive file is not open

SEE ALSO

OPEN ARCHIVE FILE (3P)
ARCHIVE STRUCTURES (3P)
INQUIRE ALL CONFLICTING STRUCTURES (3P)
SET CONFLICT RESOLUTION (3P)
RETRIEVE STRUCTURES (3P)
DELETE STRUCTURES FROM ARCHIVE (3P)
**NAME**  
RETRIEVE STRUCTURE NETWORKS – retrieve specified structure networks from specified archive file

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE presn ( AFID, N, LSTRID )
INTEGER AFID; /* archive file identifier */
INTEGER N; /* number of structure identifiers in the list */
INTEGER LSTRID(N); /* list of structure identifiers */
```

**Required PHIGS Operating States**

(HPOP, * , * , AROP)

**DESCRIPTION**

**Purpose**

Use RETRIEVE STRUCTURE NETWORKS to retrieve a list of structure networks from the specified open archive file into the Central Structure Store (CSS).

**C Input Parameters**

- `archive_id`  
  The archive identifier specifying the open archive file from which to read.

- `struct_ids`  
  A pointer to a Pint_list structure containing the list of the root structure identifiers of the networks to be retrieved. The Pint_list structure is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of structure identifiers in the list. The `ints` component is a pointer to a list of structure identifiers `num_ints` long.

**FORTRAN Input Parameters**

- `AFID`  
  The archive identifier specifying the open archive file from which to read.

- `N`  
  The number of structure identifiers to be retrieved.

- `LSTRID`  
  An array of (N) integers containing the root structure identifiers of the networks to be retrieved.

**Execution**

The structures belonging to the specified networks are copied from the indicated open archive file into the Central Structure Store (CSS). If any of the structures to be retrieved currently exist in the CSS, then the conflict is resolved as follows:

330 modified 2 April 1993
When the retrieval conflict resolution flag is MAINTAIN, the conflicting structure is not copied into the CSS (its contents are maintained).

When the retrieval conflict resolution flag is UPDATE, the conflicting structure in the CSS is overwritten (its contents are updated).

When the retrieval conflict resolution flag is ABANDON, no structures are copied to the CSS.

The retrieval conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine.

If the currently open structure must be overwritten because of a conflict while the retrieval conflict resolution flag is UPDATE, then it is closed, its contents are overwritten, and it is re-opened. The element pointer is set to point to the last element.

If any of the structures to be retrieved do not exist in the archive file, then an empty structure is created in the CSS, a warning is generated, and the retrieval operation continues for the remaining structures.

**ERRORS**

007 Ignoring function, function requires state (PHOP, *, *, AROP)

404 Ignoring function, the specified archive file is not open

405 Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON

408 Warning, some of the specified structures do not exist on the archive file. PHIGS will create empty structures in their places

**SEE ALSO**

OPEN ARCHIVE FILE (3P)

ARCHIVE STRUCTURE NETWORKS (3P)

SET CONFLICT RESOLUTION (3P)

INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)

RETRIEVE ALL STRUCTURES (3P)

RETRIEVE PATHS TO ANCESTORS (3P)

RETRIEVE PATHS TO DESCENDANTS (3P)

DELETE STRUCTURE NETWORKS FROM ARCHIVE (3P)
NAME
RETRIEVE STRUCTURES – retrieve specified structures from specified archive file

SYNOPSIS
C Syntax
void
pret_structs ( archive_id, struct_ids )
Pint archive_id; // archive identifier
Pint_list *struct_ids; // list of structure identifiers

FORTRAN Syntax
SUBROUTINE prest ( AFID, N, LSTRID )
INTEGER AFID // archive file identifier
INTEGER N // number of structure identifiers in the list
INTEGER LSTRID(N) // list of structure identifiers

Required PHIGS Operating States
(PHOP, *, *, AROP)

DESCRIPTION
Purpose
Use RETRIEVE STRUCTURES to retrieve a list of structures from the specified open archive file into the CSS.

C Input Parameters
archive_id
The archive identifier specifying the open archive file to read from.

struct_ids
A pointer to a Pint_list structure containing the list of structure identifiers to be retrieved. The Pint_list structure is defined in phigs.h as follows:

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

The num_ints component specifies the number of structure identifiers in the list.
The ints component is a pointer to a list, num_ints long, of the structure identifiers.

FORTRAN Input Parameters
AFID The archive identifier specifying the open archive file to read from.
N The number of structure identifiers to be retrieved.
LSTRID An array of N integers containing the structure identifiers to be retrieved.

Execution
The specified structures are copied from the indicated open archive file into the Central Structure Store (CSS). If any of the structures to be retrieved currently exist in the CSS, then the conflict is resolved as follows:

- When the retrieval conflict resolution flag is MAINTAIN, then the conflicting structure will not be copied into the CSS (its contents are maintained).
When the retrieval conflict resolution flag is UPDATE, then the conflicting structure in the CSS will be overwritten (its contents are updated).

When the retrieval conflict resolution flag is ABANDON, then no structures will be copied to the CSS at all.

The retrieval conflict resolution flag is set by the SET CONFLICT RESOLUTION subroutine. If the currently open structure must be overwritten because of a conflict while the retrieval conflict resolution flag is UPDATE, then it is closed, its contents are overwritten, and it is re-opened. The element pointer is set to point to the last element.

If any of the structures to be retrieved do not exist in the archive file, then an empty structure is created in the CSS, a warning is generated, and the retrieval operation continues for the remaining structures.

**ERRORS**

- 007 Ignoring function, function requires state (PHOP, *, *, AROP)
- 404 Ignoring function, the specified archive file is not open
- 405 Ignoring function, name conflict occurred while conflict resolution flag has value ABANDON
- 408 Warning, some of the specified structures do not exist on the archive file. PHIGS will create empty structures in their places

**SEE ALSO**

- OPEN ARCHIVE FILE (3P)
- ARCHIVE STRUCTURES (3P)
- SET CONFLICT RESOLUTION (3P)
- INQUIRE ALL CONFLICTING STRUCTURES (3P)
- RETRIEVE ALL STRUCTURES (3P)
- RETRIEVE STRUCTURE IDENTIFIERS (3P)
- RETRIEVE STRUCTURE NETWORKS (3P)
- DELETE STRUCTURES FROM ARCHIVE (3P)
NAME

ROTATE – calculate 2D transformation matrix to perform specified 2D rotation

SYNOPSIS

C Syntax

void protate ( angle, error_ind, m )

Pfloat angle; // rotation angle
Pint *error_ind; // OUT error indicator
Pmatrix m; // OUT transformation matrix

FORTRAN Syntax

SUBROUTINE pro ( ROTANG, ERRIND, XFRMT )

REAL ROTANG // rotation angle in radians (positive if anticlockwise)
INTEGER ERRIND // OUT error indicator
REAL XFRMT(3, 3) // OUT transformation matrix

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use ROTATE to generate a 2D homogeneous (3 × 3) transformation matrix that performs a 2D rotation.
The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION or SET GLOBAL TRANSFORMATION to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameter

angle The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, and negative angles indicate clockwise rotation.

C Output Parameters

error_ind A pointer to the location to store the error number of any error detected by this function.
m The 3 × 3 transformation matrix that performs the specified rotation. Pmatrix is defined in phigs.h as follows:

typedef Pfloat Pmatrix[3][3];

FORTRAN Input Parameter

ROTANG The angle of the rotation in radians. Positive angles indicate counter-clockwise rotation, negative angles indicate clockwise rotation.

FORTRAN Output Parameters

ERRIND The error number of any error detected by this function.
XFRMT The 3 × 3 transformation matrix that performs the specified rotation.
Execution

ROTATE returns a 2D homogeneous \((3 \times 3)\) transformation matrix that performs the rotation specified by \textit{angle}. The rotation is specified in radians. The rotation is relative to the origin of the current modelling coordinate system.

Errors

002 Ignoring function, function requires state \((\text{PHOP}, *, *, *)\)

See Also

- \textsc{rotate x (3p)}
- \textsc{rotate y (3p)}
- \textsc{rotate z (3p)}
- \textsc{set local transformation 3 (3p)}
- \textsc{build transformation matrix 3 (3p)}
- \textsc{compose matrix (3p)}
- \textsc{transform point (3p)}
NAME

ROTATE X – calculate 3D transformation matrix to perform specified rotation about x axis

SYNOPSIS

C Syntax

```c
void rotate_x ( angle, error_ind, m )
```

```c
Pfloat angle; /* rotation angle
Pint *error_ind; /* OUT error indicator
Pmatrix3 m; /* OUT transformation matrix
```

FORTRAN Syntax

```fortran
SUBROUTINE prox ( ROTANG, ERRIND, XFRMT )
```

```fortran
REAL ROTANG /* rotation angle in radians (positive if anticlockwise)
INTEGER ERRIND /* OUT error indicator
REAL XFRMT(4, 4) /* OUT transformation matrix
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use ROTATE X to generate a 3D homogeneous (4 x 4) transformation matrix that performs a 3D rotation about the x axis.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameter

angle

The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, and negative angles indicate clockwise rotation.

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

m

The 4 x 4 homogeneous transformation matrix that performs the specified rotation. Pmatrix3 is defined in phigs.h as follows:

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

FORTRAN Input Parameter

ROTANG

The angle of the rotation in radians. Positive angles indicate counter-clockwise rotation, negative angles indicate clockwise rotation.

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

XFRMT

The 4 x 4 homogeneous transformation matrix that performs the specified rotation.

336 modified 2 April 1993
**Execution**  
ROATE X returns the 3D homogeneous \((4 \times 4)\) transformation matrix that performs the rotation specified by \(angle\) about the \(x\) axis. The rotation is specified in radians and is relative to the origin of the current modelling coordinate system.

**ERRORS**  
002 Ignoring function, function requires state \((PHOP, *, *, *)\)

**SEE ALSO**  
- ROATE (3P)
- SET LOCAL TRANSFORMATION 3 (3P)
- BUILD TRANSFORMATION MATRIX 3 (3P)
- COMPOSE MATRIX 3 (3P)
- TRANSFORM POINT 3 (3P)
NAME

ROTATE Y – calculate 3D transformation matrix to perform specified rotation about y axis

SYNOPSIS

C Syntax

void protate_y ( angle, error_ind, m )

Pfloat angle; rotation angle
Pint *error_ind; OUT error indicator
Pmatrix3 m; OUT transformation matrix

FORTRAN Syntax

SUBROUTINE proy ( ROTANG, ERRIND, XFRMT )

REAL ROTANG rotation angle in radians (positive if anticlockwise)
INTEGER ERRIND OUT error indicator
REAL XFRMT(4, 4) OUT transformation matrix

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use ROTATE Y to generate a 3D homogeneous (4 x 4) transformation matrix that performs a 3D rotation about the y axis.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameter

angle The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, and negative angles indicate clockwise rotation.

C Output Parameters

error_ind A pointer to the location to store the error number of any error detected by this function.

m The 4 x 4 homogeneous transformation matrix that performs the specified rotation. Pmatrix3 is defined in phigs.h as follows:

typedef Pfloat Pmatrix3[4][4];

FORTRAN Input Parameter

ROTANG

The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, negative angles indicate clockwise rotation.

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

XFRMT The 4 x 4 homogeneous transformation matrix that performs the specified rotation.

modified 2 April 1993
Execution

ROTATE Y returns the 3D homogeneous (4 × 4) transformation matrix that performs the rotation specified by angle about the y axis. The rotation is specified in radians and is relative to the origin of the current modelling coordinate system.

ERRORS

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

SEE ALSO

ROTA TE (3P)  
SET LOCAL TRANSFORMATION 3 (3P)  
BUILD TRANSFORMATION MATRIX 3 (3P)  
COMPOSE MATRIX 3 (3P)  
TRANSFORM POINT 3 (3P)
NAME

ROTATE Z – calculate 3D transformation matrix to perform specified rotation about z axis

SYNOPSIS

C Syntax

void protate_z ( angle, error_ind, m )

Pfloat angle;  // rotation angle
Pint *error_ind;  // OUT error indicator
Pmatrix3 m;  // OUT transformation matrix

FORTRAN Syntax

SUBROUTINE proz ( ROTANG, ERRIND, XFRMT )

REAL ROTANG  // rotation angle in radians (positive if anticlockwise)
INTEGER ERRIND  // OUT error indicator
REAL XFRMT(4, 4)  // OUT transformation matrix

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use ROTATE Z to generate a 3D homogeneous (4 × 4) transformation matrix that performs a 3D rotation about the z axis.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation that is applied to output primitives during traversal.

C Input Parameter

angle  // The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, and negative angles indicate clockwise rotation.

C Output Parameters

error_ind  // A pointer to the location to store the error number of any error detected by this function.

m  // The 4 × 4 homogeneous transformation matrix that performs the specified rotation. Pmatrix3 is defined in phigs.h as follows:

typedef Pfloat Pmatrix3[4][4];

FORTRAN Input Parameter

ROTANG  // The angle of the rotation in radians. Positive angles indicate counterclockwise rotation, negative angles indicate clockwise rotation.

FORTRAN Output Parameters

ERRIND  // The error number of any error detected by this function.

XFRMT  // The 4 × 4 homogeneous transformation matrix that performs the specified rotation.

340 modified 2 April 1993
**Execution**

ROTATE $Z$ returns the 3D homogeneous $(4 \times 4)$ transformation matrix that performs the rotation specified by *angle* about the $z$ axis. The rotation is specified in radians and is relative to the origin of the current modelling coordinate system.

**ERRORS**

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

**SEE ALSO**

- ROTATE (3P)
- SET LOCAL TRANSFORMATION 3 (3P)
- BUILD TRANSFORMATION MATRIX 3 (3P)
- COMPOSE MATRIX 3 (3P)
- TRANSFORM POINT 3 (3P)
NAME
SAMPLE CHOICE – sample current measure of specified choice device

SYNOPSIS
C Syntax
void
psample_choice ( ws, dev, choice_in_status, choice )

Pint ws; /* workstation identifier */
Pint dev; /* choice device number */
Pin_status *choice_in_status; /* OUT choice input status */
Pint *choice; /* OUT choice */

FORTRAN Syntax
SUBROUTINE psmch ( WKID, CHDNR, STAT, CHNR )
INTEGER WKID /* workstation identifier */
INTEGER CHDNR /* choice device number */
INTEGER STAT /* OUT status (POK, PNCHOI) */
INTEGER CHNR /* OUT choice number */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use SAMPLE CHOICE to sample the current measure of the specified choice device without
waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET CHOICE
MODE and INITIALIZE CHOICE for more information.

C Input Parameters
ws Workstation identifier. An integer specifying the workstation with which the
specified choice device is associated.
dev The device number of the choice device to be sampled. See the AVAILABLE DEVICES
section of INITIALIZE CHOICE for a description of the available devices.

C Output Parameter
choice_in_status A pointer to a location to store the status of the sample. Pin_status is defined in
phigs.h as:
typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;
choice A pointer to a location to store the results of the choice.

FORTRAN Input Parameters
WKID The workstation identifier of the workstation associated with the device.
CHDNR The device number of the CHOICE device. See the AVAILABLE DEVICES section of
INITIALIZE CHOICE for a description of the available devices.

modified 2 April 1993
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>The measure’s choice status. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>POK</td>
</tr>
<tr>
<td></td>
<td>PNCHOI</td>
</tr>
<tr>
<td>CHNR</td>
<td>The measure’s choice number. This value is defined only if the status returned is OK.</td>
</tr>
</tbody>
</table>

### Execution

SAMPLE CHOICE samples the specified choice device. The device must be in SAMPLE mode. See SET CHOICE MODE and INITIALIZE CHOICE for more information. When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE CHOICE is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>252</td>
<td>Ignoring function, the function requires the input device to be in SAMPLE Mode</td>
</tr>
</tbody>
</table>

### See Also

- SET CHOICE MODE (3P)
- INITIALIZE CHOICE 3 (3P)
- REQUEST CHOICE (3P)
- GET CHOICE (3P)
- INQUIRE CHOICE DEVICE STATE (3P)
NAME  
SAMPLE LOCATOR – sample current measure of specified locator device

SYNOPSIS

<table>
<thead>
<tr>
<th>C Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td>psample_loc ( ws, dev, view_ind, loc_pos )</td>
</tr>
</tbody>
</table>
| Pint ws; | workstation identifier  
| Pint dev; | locator device number  
| PInt *view_ind; | OUT view index  
| Ppoint *loc_pos; | OUT locator data  

FORTRAN Syntax  
SUBROUTINE psmlc ( WKID, LCDNR, VIEWI, LPX, LPY )

| INTEGER WKID | workstation identifier  
| INTEGER LCDNR | locator device number  
| INTEGER VIEWI | OUT view index  
| REAL LPX, LPY | OUT locator position in WC  

Required PHIGS Operating States  
(PHOP, WSOP, *, *)

DESCRIPTION

Purpose
Use SAMPLE LOCATOR to sample the current measure of the specified locator device without waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET LOCATOR MODE and INITIALIZE LOCATOR for more information.

C Input Parameters

| ws | Workstation identifier. An integer specifying the workstation with which the specified locator device is associated.  
| dev | The device number of the locator device to be sampled. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.  

C Output Parameters

| view_ind | The index of the view representation used to transform the Normalized Projection Coordinate (NPC) position to a World Coordinates (WC) position.  
| loc_pos | Ppoint is defined in phigs.h as follows:  

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

FORTRAN Input Parameters

| WKID | The workstation identifier of the workstation associated with the device.  
| LCDNR | The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.  
| VIEWI | The index of the view representation used to transform the Normalized Projection Coordinate (NPC) position to a World Coordinates (WC) position.  

modified 2 April 1993
SAMPLE LOCATOR

**LPX, LPY**  
The locator position in WC.

**Execution**  
SAMPLE LOCATOR samples the specified locator device. The device must be in SAMPLE mode. See SET LOCATOR MODE and INITIALIZE LOCATOR for more information.

When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE LOCATOR is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

A LOCATOR device measure consists of a *position* and a *view index*. Position is the WC point corresponding to the position on the workstation selected by the operator. The view index is the index of the view used to transform *position* from Device Coordinates (DC) to WC. See INITIALIZE LOCATOR and SET VIEW TRANSFORMATION INPUT PRIORITY for a description of how this view representation is determined. The workstation transform is used to transform the DC position to a NPC position.

**Note:** The 2D and 3D locator measure processes are the same except that the 2D process discards the \( z \) coordinate. The 3D version of this function, SAMPLE LOCATOR 3, can be used if the value of the \( z \) coordinate is needed.

**ERRORS**

- **003** Ignoring function, function requires state (PHOP, WSOF, *, *)
- **054** Ignoring function, the specified workstation is not open
- **061** Ignoring function, specified workstation is not category INPUT or category OUTIN
- **250** Ignoring function, the specified device is not available on the specified workstation
- **252** Ignoring function, the function requires the input device to be in SAMPLE Mode

**SEE ALSO**

- INITIALIZE LOCATOR (3P)
- SET LOCATOR MODE (3P)
- REQUEST LOCATOR 3 (3P)
- GET LOCATOR (3P)
- INQUIRE LOCATOR DEVICE STATE (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
- SAMPLE LOCATOR 3 (3P)
NAME  
SAMPLE LOCATOR 3 – sample current measure of specified locator device

SYNOPSIS

C Syntax

void psample_loc3 ( ws, dev, view_ind, loc_pos )

Pint ws;  /* workstation identifier */
Pint dev;  /* locator device number */
Pint *view_ind;  /* OUT view index */
Ppoint3 *loc_pos;  /* OUT locator position */

FORTRAN Syntax

SUBROUTINE psmlc3 ( WKID, LCDNR, VIEWI, LPX, LPY, LPZ )

INTEGER WKID  /* workstation identifier */
INTEGER LCDNR  /* locator device number */
INTEGER VIEWI  /* OUT view index */
REAL LPX, LPY, LPZ  /* OUT locator position in WC */

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use SAMPLE LOCATOR 3 to sample the current measure of the specified locator device without waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET LOCATOR MODE and INITIALIZE LOCATOR 3 for more information.

C Input Parameters

ws  Workstation identifier. An integer specifying the workstation with which the specified locator device is associated.

dev  The device number of the locator device to be sampled. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

C Output Parameters

view_ind  The index of the view representation used to transform the Normalized Projection Coordinate (NPC) position to a World Coordinates (WC) position.

loc_pos  Ppoint3 is defined in phigs.h as follows:

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

FORTRAN Input Parameters

WKID  The workstation identifier of the workstation associated with the device.

LCDNR  The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

VIEWI  The index of the view representation used to transform the Normalized
Projection Coordinate (NPC) position to a World Coordinates (WC) position.

\[ LPX, LPY, LPZ \]

The locator position in WC.

**Execution**

SAMPLE LOCATOR 3 samples the specified locator device. The device must be in SAMPLE mode. For more information, see SET LOCATOR MODE and INITIALIZE LOCATOR 3.

When a device is set to SAMPLE mode, a measure process is created for the device, and the device is activated. When SAMPLE LOCATOR 3 is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

A LOCATOR device measure consists of a position and a view index. position is the WC point corresponding to the position on the workstation selected by the operator. The view index is the index of the view used to transform position from Device Coordinates (DC) to WC. For a description of how this view representation is determined, see INITIALIZE LOCATOR 3 and SET VIEW TRANSFORMATION INPUT PRIORITY. The workstation transform is used to transform the DC position to a NPC position.

**Note:** The 2D and 3D locator measure processes are the same except that the 2D process discards the z coordinate. The 2D version of this function, SAMPLE LOCATOR, can be used if the value of the z coordinate is not needed.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, category of the specified workstation is not INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
252 Ignoring function, the function requires the input device to be in SAMPLE mode

**SEE ALSO**

INITIALIZE LOCATOR 3 (3P)
SET LOCATOR MODE (3P)
REQUEST LOCATOR 3 (3P)
GET LOCATOR 3 (3P)
INQUIRE LOCATOR DEVICE STATE 3 (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
SAMPLE LOCATOR (3P)
**NAME**
SAMPLE PICK – sample current measure of specified pick device

**SYNOPSIS**

**C Syntax**

```c
void
psample_pick ( ws, dev, depth, pick_in_status, rpick )
```

- `Pint ws;`  
  workstation identifier
- `Pint dev;`  
  pick device number
- `Pint depth;`  
  max. depth of returned path
- `Pin_status *pick_in_status;`  
  OUT pick input status
- `Ppick_path *rpick;`  
  OUT pick path

**FORTRAN Syntax**

```fortran
SUBROUTINE psmpk ( WKID, PKDNR, IPPD, STAT, PPD, PP )
```

- `INTEGER WKID`  
  workstation identifier
- `INTEGER PKDNR`  
  pick device number
- `INTEGER IPPD`  
  depth of pick path to return
- `INTEGER STAT`  
  OUT status (POK, PNPICK)
- `INTEGER PPD`  
  OUT depth of actual pick path
- `INTEGER PP(3, IPPD)`  
  OUT pick path

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**
Use SAMPLE PICK to sample the current measure of the specified pick device without waiting for the device trigger to fire. The device must be in SAMPLE mode. For more information, see SET PICK MODE and INITIALIZE PICK.

**C Input Parameters**

- `ws`  
  Workstation identifier. An integer specifying the workstation with which the specified pick device is associated.
- `dev`  
  The device number of the pick device to be sampled. For a description of the available devices, see the AVAILABLE DEVICES section of INITIALIZE PICK.
- `depth`  
  The maximum number of path elements to return.

**C Output Parameter**

- `pick_in_status`  
  `Pin_status` is defined in phigs.h as:
  ```c
typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
  } Pin_status;
```

- `rpick`  
  `Ppick_path` is defined in phigs.h as:
  ```c
typedef struct {
```
Pint depth; /* pick path depth */
Ppick_path_elem *path_list; /* pick path list */
} Ppick_path;

depth indicates the number of elements in the measure’s path. This is the depth
value contained in the device’s current measure. It is not affected by the
maximum depth to return parameter. Thus the number of elements
returned in path_list may be less than depth.

path_list

is the array of path elements defining the location of the picked primitive
in the Central Structure Store (CSS).

Note: Before calling this function, this array must be allocated by the
calling program and the array pointer assigned to this field.

The array must be at least of length maximum depth to return.

Ppick_path_elem is defined in phigs.h as:

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

struct_id, pick_id, and elem_pos

are the structure identifier, pick identifier, and element number,
respectively, of each element in the path.

FORTRAN Input
Parameters

WKID The workstation identifier of the workstation associated with the device.

PKDNR The device number of the PICK device. For a description of the available devices,
see the AVAILABLE DEVICES section of INITIALIZE PICK.

IPPD The maximum number of path elements to return.

FORTRAN Output
Parameters

STAT The measure’s pick status. Valid values as defined in phigs77.h are:

    POK     OK
    PNPICK  No pick

PPD The number of elements in the measure’s path. This value is undefined if the
status returned is PNPICK. This is the depth value contained in the measure and is
not affected by the maximum depth to return parameter, PPD. Thus the number of
elements returned in PP may be less than IPPD.

PP An array in which to store the measure’s pick path. The contents of this array are
undefined if the status returned is PNPICK. This is the two-dimensional array of
path elements defining the location of the picked primitive in the CSS. Each row
of the array contains the structure identifier, pick identifier, and element number,
respectively, of each element in the path. The array must be at least of dimension
(3, IPPD).

modified 2 April 1993
SAMPLE PICK samples the specified pick device. The device must be in SAMPLE mode. For more information, see SET PICK MODE and INITIALIZE PICK.

When a device is set to SAMPLE mode, a measure process is created for the device, and the device is activated. When SAMPLE PICK is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

A PICK device measure consists of a status and a pick path. status indicates whether a pick by the operator was successfully resolved. pick_path describes the location of the picked primitive, if any, in the Central Structure Store (CSS). The pick filter of a PICK device controls which output primitives on the device’s workstation are pickable. By default, no output primitives are pickable. For more information about the pick filter, see SET PICK FILTER.

ERRORS

054 Ignoring function, the specified workstation is not open
060 Ignoring function, category of the specified workstation is not OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
252 Ignoring function, the function requires the input device to be in SAMPLE mode

SEE ALSO

SET PICK FILTER (3P)
SET PICK IDENTIFIER (3P)
SET PICK MODE (3P)
INITIALIZE PICK 3 (3P)
REQUEST PICK (3P)
GET PICK (3P)
INQUIRE PICK DEVICE STATE (3P)
NAME
  SAMPLE STRING – sample current measure of specified string device

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

  FORTRAN Syntax
  SUBROUTINE psmst ( WKID, STDNR, LOSTR, STR )
  INTEGER  WKID       workstation identifier
  INTEGER  STDNR      string device number
  INTEGER  LOSTR      OUT number of characters returned
  CHARACTER(*)  STR   OUT string

  FORTRAN Subset Syntax
  SUBROUTINE psmst ( WKID, STDNR, LOSTR, STR )
  INTEGER  WKID       workstation identifier
  INTEGER  STDNR      string device number
  INTEGER  LOSTR      OUT number of characters returned
  CHARACTER(*80)  STR   OUT string

Required PHIGS Operating States
  (PHOP, WSOP, *, *)

DESCRIPTION
  Purpose
  Use SAMPLE STRING to sample the current measure of the specified string device without
  waiting for the device trigger to fire. The device must be in SAMPLE mode. For more
  information, see SET STRING MODE and INITIALIZE STRING.

  C Input Parameters
  ws         Workstation identifier. An integer specifying the workstation with which the
              specified string device is associated.

  dev        The device number of the string device to be sampled. See the AVAILABLE DEVICES
              section of INITIALIZE STRING for a description of the available devices.

  C Output Parameter
  string     A pointer to a character array in which to store the STRING measure. The string is
              null terminated. The array should be at least as large as the buffer of the STRING
              device. This buffer size is set when the device is initialized.

  FORTRAN Input Parameters
  WKID        The workstation identifier of the workstation associated with the device.

  STDNR       The device number of the STRING device. See the AVAILABLE DEVICES section of
              INITIALIZE STRING for a description of the available devices.

modified 2 April 1993
**FORTRAN Output Parameters**

- **LOSTR** The number of characters returned in *STR*.
- **STR** An array in which to store the STRING measure. The array must be at least as large as the buffer of the STRING device. This buffer size is set when the device is initialized.

   The FORTRAN subset version of this function returns no more than 80 of the characters in the measure.

**Execution**

SAMPLE STRING samples the specified string device. The device must be in SAMPLE mode. See SET STRING MODE and INITIALIZE STRING for more information.

When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE STRING is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 061 Ignoring function, category of the specified workstation is not INPUT or OUTIN
- 250 Ignoring function, the specified device is not available on the specified workstation
- 252 Ignoring function, the function requires the input device to be in SAMPLE mode

**SEE ALSO**

- SET STRING MODE (3P)
- INITIALIZE STRING 3 (3P)
- REQUEST STRING (3P)
- GET STRING (3P)
- INQUIRE STRING DEVICE STATE (3P)
NAME | SAMPLE STROKE – sample current measure of specified stroke device

SYNOPSIS

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE psmsk ( WKID, SKDNR, N, VIEWI, NP, PXA, PYA )
INTEGER WKID           /* workstation identifier */
INTEGER SKDNR          /* stroke device number */
INTEGER N              /* maximum number of points */
INTEGER VIEWI          /* OUT view index */
INTEGER NP             /* OUT number of points */
REAL PXA(N), PYA(N)    /* OUT points in stroke in World Coordinates */
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use SAMPLE STROKE to sample the current measure of the specified stroke device without waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET STROKE MODE and INITIALIZE STROKE for more information.

C Input Parameters

- **ws** Workstation identifier. An integer specifying the workstation with which the specified stroke device is associated.
- **dev** The device number of the stroke device to be sampled. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.

C Output Parameters

- **view_ind** The view index used to transform the Device Coordinate (DC) positions to World Coordinate (WC) points.
- **stroke** Ppoint_list is defined in phigs.h as follows:

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

  *num_points* is the number of points in *points*.

  *points* is the array of Ppoint structures specifying the points in WC.

  **Note:** This array must be allocated by the calling program and the array pointer.

modified 2 April 1993
assigned to this field before calling this function. The array must be at least as large as the buffer of the STROKE device. This buffer size is set when the device is initialized. Ppoint is defined in phigs.h as follows:

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

FORTRAN Input Parameters

- **WKID**  
The workstation identifier of the workstation associated with the device.

- **SKDNR**  
The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.

- **N**  
The maximum number of points to store in **PXA** and **PYA**.

FORTRAN Output Parameters

- **VIEWI**  
The view index used to transform the Device Coordinate (DC) positions to World Coordinate (WC) points.

- **NP**  
The number of points in the measure.

- **PXA, PYA**  
The arrays in which to store the points in WC. The arrays must be at least as large as NP.

Execution

SAMPLE STROKE samples the specified stroke device. The device must be in SAMPLE mode. See SET STROKE MODE and INITIALIZE STROKE for more information.

When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE STROKE is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

A STROKE device measure consists of a list of WC points and a view index. The points correspond to positions on the workstation selected by the operator. The view index is the index of the view used to transform these positions from DC to WC. See INITIALIZE STROKE and SET VIEW TRANSFORMATION INPUT PRIORITY for a description of how this view representation is determined. The workstation transformation is used to map the DC position to a Normalized Projection Coordinate (NPC) position.

Note: The 2D and 3D stroke measure processes are the same except that the 2D process discards the z coordinate. The 3D version of this function, SAMPLE STROKE 3, can be used if the value of the z coordinate is needed.

ERRORS

- **003**  
  Ignoring function, function requires state (PHOP, WSOP, *, *)

- **054**  
  Ignoring function, the specified workstation is not open

- **061**  
  Ignoring function, category of the specified workstation is not INPUT or OUTIN

modified 2 April 1993
Ignoring function, the specified device is not available on the specified workstation

Ignoring function, the function requires the input device to be in SAMPLE mode

SEE ALSO

INITIALIZE STROKE (3P)
SET STROKE MODE (3P)
REQUEST STROKE 3 (3P)
GET STROKE (3P)
INQUIRE STROKE DEVICE STATE (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
SAMPLE STROKE 3 (3P)
### NAME
SAMPLE STROKE 3 – sample current measure of specified stroke device

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE psmsk3 ( WKID, SKDNR, N, VIEWI, NP, PXA, PYA, PZA )
INTEGER WKID /* workstation identifier */
INTEGER SKDNR /* stroke device number */
INTEGER N /* maximum number of points */
INTEGER VIEWI /* OUT view index */
INTEGER NP /* OUT number of points */
REAL PXA(N), PYA(N), PZA(N) /* OUT points in stroke in World Coordinates */
```

### FORTRAN Syntax

**SUBROUTINE psmsk3 ( WKID, SKDNR, N, VIEWI, NP, PXA, PYA, PZA )**

- **WKID** — workstation identifier
- **SKDNR** — stroke device number
- **N** — maximum number of points
- **VIEWI** — OUT view index
- **NP** — OUT number of points
- **PXA(N)**, **PYA(N)**, **PZA(N)** — OUT points in stroke in World Coordinates

### Required PHIGS Operating States

( PHOP, WSOP, *, * )

### DESCRIPTION

**Purpose**
Use SAMPLE STROKE 3 to sample the current measure of the specified stroke device without waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET STROKE MODE and INITIALIZE STROKE 3 for more information.

**C Input Parameters**
- **ws** — Workstation identifier. An integer specifying the workstation with which the specified stroke device is associated.
- **dev** — The device number of the stroke device to be sampled. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available devices.

**C Output Parameters**
- **view_ind** — The view index used to transform the Device Coordinate (DC) positions to World Coordinate (WC) points.
- **stroke** — Ppoint_list3 is defined in phigs.h as follows:
  ```c
typedef struct {
    Pint num_points;
    Ppoint3 *points;
  } Ppoint_list3;
  ```
  - **num_points** is the number of points in **points**.
  - **points** is the array of Ppoint3 structures specifying the points in WC.

**Note:** This array must be allocated by the calling program and the array pointer.
assigned to this field before calling this function.

The array must be at least as large as the buffer of the STROKE device. This buffer size is set when the device is initialized. Ppoint3 is defined in phigs.h as follows:

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

**FORTRAN Input Parameters**

- **WKID** The workstation identifier of the workstation associated with the device.
- **SKDNR** The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available devices.
- **N** The maximum number of points to store in PXA, PYA, and PZA.

**FORTRAN Output Parameters**

- **VIEWI** The view index used to transform the Device Coordinate (DC) positions to World Coordinate (WC) points.
- **NP** The number of points in the measure.
- **PXA, PYA, PZA** The arrays in which to store the points in WC. The arrays must be at least as large as NP.

**Execution**

SAMPLE STROKE 3 samples the specified stroke device. The device must be in SAMPLE mode. See SET STROKE MODE and INITIALIZE STROKE 3 for more information.

When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE STROKE 3 is called, the current value of the measure process for the specified device is retrieved without waiting for the device’s trigger to fire.

A STROKE device measure consists of a list of WC points and a view index. The points correspond to positions on the workstation selected by the operator. The view index is the index of the view used to transform these positions from DC to WC. See INITIALIZE STROKE 3 and SET VIEW TRANSFORMATION INPUT PRIORITY for a description of how this view representation is determined. The workstation transformation is used to map the DC position to a Normalized Projection Coordinate (NPC) position.

**Note:** The 2D and 3D stroke measure processes are the same except that the 2D process discards the z coordinate. The 2D version of this function, SAMPLE STROKE, can be used if the value of the z coordinate is not needed.

**ERRORS**

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **061** Ignoring function, category of the specified workstation is not INPUT or OUTIN
- **250** Ignoring function, the specified device is not available on the specified
Ignoring function, the function requires the input device to be in SAMPLE mode.

**SEE ALSO**

- INITIALIZE STROKE 3 (3P)
- SET STROKE MODE (3P)
- REQUEST STROKE 3 (3P)
- GET STROKE 3 (3P)
- INQUIRE STROKE DEVICE STATE 3 (3P)
- SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
- SAMPLE STROKE (3P)
### NAME

SAMPLE VALUATOR – sample current measure of specified valuator device

### SYNOPSIS

**C Syntax**

```c
void
psample_val ( ws, dev, value )
Pint ws;  \hspace{1cm} \text{workstation identifier}
Pint dev; \hspace{1cm} \text{valuator device number}
Pfloat *value; \hspace{1cm} \text{OUT value}
```

**FORTRAN Syntax**

```fortran
SUBROUTINE psmvl ( WKID, VLDNR, VAL )
INTEGER WKID \hspace{1cm} \text{workstation identifier}
INTEGER VLDNR \hspace{1cm} \text{valuator device number}
REAL VAL \hspace{1cm} \text{OUT value}
```

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

**Purpose**

Use SAMPLE VALUATOR to sample the current measure of the specified valuator device without waiting for the device trigger to fire. The device must be in SAMPLE mode. See SET VALUATOR MODE and INITIALIZE VALUATOR for more information.

**C Input Parameters**

- `ws` Workstation identifier. An integer specifying the workstation with which the specified valuator device is associated.
- `dev` The device number of the valuator device to be sampled. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available devices.

**C Output Parameter**

- `value` A pointer to a Pfloat variable in which to store the VALUATOR measure.

**FORTRAN Input Parameters**

- `WKID` The workstation identifier of the workstation associated with the device.
- `VLDNR` The device number of the VALUATOR device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available devices.

**FORTRAN Output Parameter**

- `VAL` A variable in which to store the VALUATOR measure.

**Execution**

SAMPLE VALUATOR samples the specified valuator device. The device must be in SAMPLE mode. See SET VALUATOR MODE and INITIALIZE VALUATOR for more information.

When a device is set to SAMPLE mode, a measure process is created for the device and the device is activated. When SAMPLE VALUATOR is called, the current value of the measure process for the specified device is retrieved without waiting for the device trigger to fire.

modified 2 April 1993
## ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, category of specified workstation is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>252</td>
<td>Ignoring function, the function requires the input device to be in SAMPLE mode</td>
</tr>
</tbody>
</table>

## SEE ALSO

- SET VALUATOR MODE (3P)
- INITIALIZE VALUATOR 3 (3P)
- REQUEST VALUATOR (3P)
- GET VALUATOR (3P)
- INQUIRE VALUATOR DEVICE STATE (3P)
<table>
<thead>
<tr>
<th>NAME</th>
<th>SCALE – calculate 2D transformation matrix to perform specified scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td></td>
</tr>
<tr>
<td>C Syntax</td>
<td></td>
</tr>
<tr>
<td>void pscale ( scale_vector, error_ind, m )</td>
<td></td>
</tr>
<tr>
<td>Pvec *scale_vector; scale factor vector</td>
<td></td>
</tr>
<tr>
<td>Pint *error_ind; OUT error indicator</td>
<td></td>
</tr>
<tr>
<td>Pmatrix m; OUT transformation matrix</td>
<td></td>
</tr>
<tr>
<td>FORTRAN Syntax</td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE psc ( FX, FY, ERRIND, XFRMT )</td>
<td></td>
</tr>
<tr>
<td>REAL FX, FY scale factor vector</td>
<td></td>
</tr>
<tr>
<td>INTEGER ERRIND OUT error indicator</td>
<td></td>
</tr>
<tr>
<td>REAL XFRMT(3, 3) OUT transformation matrix</td>
<td></td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, *, *, *)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td>Use SCALE to generate a 2D homogeneous (3 x 3) transformation matrix that performs a 2D scaling. The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION or SET GLOBAL TRANSFORMATION to modify the modelling transformation that is applied to output primitives during traversal.</td>
</tr>
<tr>
<td>C Input Parameter</td>
<td>scale_vector</td>
</tr>
<tr>
<td>A pointer to a Pvec data structure containing the scale factors to be applied to the x and y dimensions. Pvec is defined in phigs.h as follows:</td>
<td></td>
</tr>
<tr>
<td>typedef struct</td>
<td></td>
</tr>
<tr>
<td>{ Pfloat delta_x; /* x coordinate */</td>
<td></td>
</tr>
<tr>
<td>Pfloat delta_y; /* y coordinate */</td>
<td></td>
</tr>
<tr>
<td>} Pvec;</td>
<td></td>
</tr>
<tr>
<td>C Output Parameters</td>
<td>error_ind</td>
</tr>
<tr>
<td>A pointer to the location to store the error number of any error this function detects.</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>The 3 x 3 homogeneous transformation matrix that performs the specified scaling. Pmatrix is defined in phigs.h as follows:</td>
</tr>
<tr>
<td>typedef Pfloat Pmatrix[3][3];</td>
<td></td>
</tr>
<tr>
<td>FORTRAN Input Parameter</td>
<td>FX, FY The scale factors to be applied to the x and y dimensions.</td>
</tr>
</tbody>
</table>

modified 2 April 1993
**FORTRAN Output**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td>XFRMT</td>
<td>The (3 \times 3) homogeneous transformation matrix that performs the specified scaling.</td>
</tr>
</tbody>
</table>

**Execution**

SCALE returns a 2D homogeneous \((3 \times 3)\) transformation matrix that performs the scaling specified by the *scale factor vector*. Scaling is relative to the origin of the current modelling coordinate system.

The scale factor vector specifies \(sx\) and \(sy\) scaling factors that control scaling in the \(x\) and \(y\) directions.

**ERRORS**

002 Ignoring function, function requires state \((\text{PHOP, }*,*,*)\)

**SEE ALSO**

- SET LOCAL TRANSFORMATION (3P)
- ROTATE (3P)
- TRANSLATE (3P)
- BUILD TRANSFORMATION MATRIX (3P)
- COMPOSE MATRIX (3P)
- SCALE 3 (3P)
- TRANSFORM POINT (3P)
**NAME**  
SCALE 3 – calculate 3D transformation matrix to perform specified scaling

**SYNOPSIS**

C Syntax

```c
void pscale3 ( scale_vector, error_ind, m )
Pvec3 *scale_vector;  // scale factor vector
Pint *error_ind;      // OUT error indicator
Pmatrix3 m;           // OUT transformation matrix
```

FORTRAN Syntax

```fortran
SUBROUTINE psc3 ( FX, FY, FZ, ERRIND, XFRMT )
REAL FX, FY, FZ  // scale factor vector
INTEGER ERRIND  // OUT error indicator
REAL XFRMT(4, 4) // OUT transformation matrix
```

**FORTRAN Syntax**

SUBROUTINE psc3 ( FX, FY, FZ, ERRIND, XFRMT )
REAL FX, FY, FZ  // scale factor vector
INTEGER ERRIND  // OUT error indicator
REAL XFRMT(4, 4) // OUT transformation matrix

**Required PHIGS Operating States**

( PHOP, * , * , * )

**DESCRIPTION**

**Purpose**

Use SCALE 3 to generate a 3D homogeneous (4 × 4) transformation matrix that performs a 3D scaling.

The returned matrix may be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation that is applied to output primitives during traversal.

**C Input Parameter**

`scale_vector`

A pointer to a Pvec3 data structure containing the scale factors to be applied to the x, y, and z dimensions. Pvec3 is defined in phigs.h as follows:

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

**C Output Parameters**

`error_ind`

A pointer to the location to store the error number of any error this function detects.

`m`

The 4 × 4 homogeneous transformation matrix that performs the specified scaling. Pmatrix3 is defined in phigs.h as follows:

```c
typedef Pfloat Pmatrix3[4][4];
```
FORTRAN Input Parameter

$FX$, $FY$, $FZ$

The scale factors to be applied to the $x$, $y$, and $z$ dimensions.

FORTRAN Output Parameters

$ERRIND$

The error number of any error detected by this function.

$XFRMT$ The $4 \times 4$ homogeneous transformation matrix that performs the specified scaling.

Execution

$SCALE\ 3$ returns a 3D homogeneous ($4 \times 4$) transformation matrix that performs the scaling specified by the scale factor vector. Scaling is relative to the origin of the current modelling coordinate system.

The scale factor vector specifies $sx$, $sy$, and $sz$ scaling factors that control scaling in the $x$, $y$, and $z$ directions.

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

SEE ALSO

SET LOCAL TRANSFORMATION 3 (3P)
ROTATE X (3P)
ROTATE Y (3P)
ROTATE Z (3P)
TRANSLATE 3 (3P)
BUILD TRANSFORMATION MATRIX 3 (3P)
COMPOSE MATRIX 3 (3P)
SCALE (3P)
TRANSFORM POINT 3 (3P)
NAME

TEXT – create structure element specifying 2D text primitive

SYNOPSIS

C Syntax

```c
void ptext ( text_pt, text )
Ppoint text_pt; /* text point */
char text; /* text string */
```

FORTRAN Syntax

```fortran
SUBROUTINE ptx ( PX, PY, CHARS )
REAL PX, PY /* text point (MC) */
CHARACTER*(*) CHARS /* string of characters */
```

FORTRAN Subset Syntax

```fortran
SUBROUTINE ptxs ( PX, PY, LSTR, CHARS )
REAL PX, PY /* text point (MC) */
INTEGER LSTR /* length of string (in characters) */
CHARACTER*80 CHARS /* string of characters */
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

The TEXT subroutine places a structure element containing the full specification of a two-dimensional TEXT primitive into the currently-open structure.

The TEXT primitive is a character string. The text point subroutine parameter, specified in Modelling Coordinates (MC), controls the location of the string in the display. The z coordinate is assumed to be 0. Aspects of the text display such as the font, colour, spacing, height, and alignment are controlled by the current values of the primitive attributes listed below.

When the current edit mode is INSERT, the structure element created by the TEXT subroutine is inserted into the open structure after the element pointed to by the current element pointer. When the current edit mode is REPLACE, the new TEXT element replaces the element in the structure pointed to by the element pointer. In either case, the element pointer is updated to point to the new TEXT element.

C Input Parameters

`text_pt` A pointer to a Ppoint structure containing the x and y coordinates locating the text string. The Ppoint structure is defined in phigs.h as follows:

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

`text` A pointer to the ASCII character string to be written into the display.
**TEXT (3P)**

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>PX</th>
<th>The x coordinate of the point locating the text string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PY</td>
<td>The y coordinate of the point locating the text string.</td>
</tr>
</tbody>
</table>

**CHARS** A character array containing the ASCII string to be written to the display.

It is recommended that you use substrings or constants, use only the portion desired, and avoid using the blank-padded portion. An example of a substring is `LABEL(1:8)`, where label may be declared as `CHARACTER*256`. An example of a character constant is `abcdefg`. Another way to achieve the same result is to null terminate the string (for example, use `hello\0`). However, a string returned from a SunPHIGS inquiry function is not null-terminated, although it may have been originally specified that way.

**Execution**

When the structure is traversed, the TEXT element draws the specified character string. The position of the text is defined in relation to text point by the current values of the text primitive attributes `CHARACTER UP VECTOR`, `TEXT PATH`, and `TEXT ALIGNMENT`.

Other aspects of the appearance of the text are controlled by the attributes `TEXT FONT`, `TEXT PRECISION`, `CHARACTER HEIGHT`, `CHARACTER WIDTH`, `CHARACTER EXPANSION FACTOR`, `CHARACTER SPACING`, `TEXT COLOUR INDEX` and `CHARACTER SETS`.

The text point is specified in Modelling Coordinates (MC). The text primitive is subject to the current transformations in the transformation pipeline from the MC system to the workstation display.

**Note:** TEXT is limited to ASCII character strings. If an application requires other character sets, use `GENERALIZED DRAWING PRIMITIVE -17`.

**Attributes Applied**

The attributes listed below are used to display the TEXT primitive when the structure is traversed. The Aspect Source Flags (ASFs) identify where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be indirectly accessed by using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

```plaintext
- text font
- text precision
- character expansion factor
- character spacing
- text colour
- character height
- character up vector
- text path
- text alignment
- text index
- depth cue index
- name set
```

SunPHIGS Release 3.0

modified 2 April 1993
Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ESCAPE -12 (3P)
INQUIRE TEXT EXTENT (3P)
INQUIRE TEXT FACILITIES (3P)
GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
TEXT 3 (3P)
NAME
TEXT 3 – create structure element specifying 3D text primitive

SYNOPSIS
C Syntax

```c
void ptext3 ( text_pt, dir, text )
Ppoint3 *text_pt;    // text point
Pvec3 dir[2];       // direction vectors
char *text;         // text string
```

FORTRAN Syntax

```fortran
SUBROUTINE ptx3 ( PX, PY, PZ, TDX, TDY, TDZ, CHAR )
REAL PX, PY, PZ       // text point (MC)
REAL TDX(2), TDY(2), TDZ(2)  // text direction vectors (MC)
CHARACTER*(*) CHAR    // string of characters
```

FORTRAN Subset Syntax

```fortran
SUBROUTINE ptx3s ( PX, PY, PZ, TDX, TDY, TDZ, LSTR, CHAR )
REAL PX, PY, PZ       // text point (MC)
REAL TDX(2), TDY(2), TDZ(2)  // text direction vectors (MC)
INTEGER LSTR          // length of string (in characters)
CHARACTER*80 CHAR    // string of characters
```

Required PHIGS Operating States

( PHOP, *, STOP, * )

DESCRIPTION

Purpose

The TEXT 3 subroutine places a structure element containing the full specification of a three-dimensional TEXT 3 primitive into the currently-open structure.

The TEXT 3 primitive is an ASCII character string. The text point and direction vectors subroutine parameters, specified in Modelling Coordinates (MC), control the location and orientation of the string in the display. Other aspects of the text display, such as the font, colour, spacing, height, and alignment, are controlled by the current values of the primitive attributes listed below.

When the current edit mode is INSERT, the structure element created by the TEXT 3 subroutine is inserted into the open structure after the element pointed to by the current element pointer. When the current edit mode is REPLACE, the TEXT 3 element replaces the element in the structure pointed to by the element pointer. In either case, the element pointer is updated to point to the new TEXT 3 element.

C Input Parameters

text_pt A pointer to a Ppoint3 structure that specifies the x, y, and z coordinates that locate the text string. The Ppoint3 structure is defined in phigs.h as follows:

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

modified 2 April 1993
An array of two Pvec3 structures containing direction vectors. The Pvec3 structure is defined in phigs.h as follows:

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

text is a pointer to the ASCII character string to be written to the display.

FORTRAN Input Parameters

- PX: The x coordinate of the point locating the text string.
- PY: The y coordinate of the point locating the text string.
- PZ: The z coordinate of the point locating the text string.
- TDX: An array giving the x coordinates of two text direction vectors.
- TDY: An array giving the y coordinates of two text direction vectors.
- TDZ: An array giving the z coordinates of two text direction vectors.

It is recommended that you use substrings or constants, use only the portion desired, and avoid using the blank-padded portion. An example of a substring is `LABEL(1:8)`, where `label` may be declared as `CHARACTER*256`. An example of a character constant is `abcdefg`. Another way to achieve the same result is to null terminate the string (for example, `hello\0`). However, a string returned from a SunPHIGS inquiry function is not null-terminated, although it may have been originally specified that way.

Execution

When the structure is traversed, the TEXT 3 element draws the specified character string on the plane in the MC system defined by the text point and the two direction vectors. These parameters define a Text Local Coordinate (TLC) system in the MC system. The text point parameter defines the origin of the TLC system. The first direction vector defines the positive x axis, and the second direction vector defines the positive y axis. Only the directions, not the lengths, of these vectors are relevant.

The precise position of the text is defined in relation to this plane by the current values of the text primitive attributes CHARACTER UP VECTOR, TEXT PATH, and TEXT ALIGNMENT. The text primitive is subject to the current transformations in the transformation pipeline from the MC system to the workstation display.

Other aspects of the text appearance are controlled by the TEXT FONT, TEXT PRECISION, CHARACTER HEIGHT, CHARACTER WIDTH, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, TEXT COLOUR INDEX, and CHARACTER SETS attributes.
Note: TEXT 3 is limited to ASCII character strings. If an application requires other character sets, use GENERALIZED DRAWING PRIMITIVE 3 -17.

Attributes Applied

The attributes listed below are used to display the TEXT 3 primitive when the structure is traversed. The Aspect Source Flags (ASFs) identify the location of the output display attributes for access. These attributes can come directly from the traversal state list, or they can be indirectly accessed, by using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

- text font
- text precision
- character expansion factor
- character spacing
- text colour
- character height
- character up vector
- text path
- text alignment
- text index
- depth cue index
- name set

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ESCAPE -12 (3P)
GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
INQUIRE TEXT EXTENT (3P)
INQUIRE TEXT FACILITIES (3P)
TEXT (3P)
### NAME
TRANSFORM POINT – apply a 2D transformation matrix to a specified 2D point

### SYNOPSIS

#### C Syntax

```c
void ptran_point ( p, m, error_ind, r )
Ppoint *p;     /* point */
Pmatrix m;     /* transformation matrix */
Pint *error_ind; /* OUT error indicator */
Ppoint *r;     /* OUT transformed point */
```

#### FORTRAN Syntax

```fortran
SUBROUTINE ptp ( XI, YI, XFRMT, ERRIND, XO, YO )
REAL XI, YI     /* point */
REAL XFRMT(3, 3) /* transformation matrix */
INTEGER ERRIND  /* OUT error indicator */
REAL XO, YO     /* OUT transformed point */
```

### Required PHIGS Operating States

(PHOP, *, *, *)

### DESCRIPTION

**Purpose**
Use `TRANSFORM POINT` to calculate the transformed coordinates of a 2D point.

**C Input Parameters**
- `p` A pointer to a `Ppoint` data structure containing the coordinates of the point to be transformed. `Ppoint` is defined in `phigs.h` as follows:
  ```c
typedef struct {
    Pfloat x;        /* x coordinate */
    Pfloat y;        /* y coordinate */
  } Ppoint;
  ```
- `m` The $3 \times 3$ homogeneous transformation matrix to apply to $p$. `Pmatrix` is defined in `phigs.h` as follows:
  ```c
typedef Pfloat Pmatrix[3][3];
  ```

**C Output Parameters**
- `error_ind` A pointer to the location in which to store the error number of any error detected by this function.
- `r` A pointer to a `Ppoint` structure in which to store the coordinates of the transformed point.

**FORTRAN Input Parameters**
- `XI, YI` The coordinates of the point to be transformed.
- `XFRMT` The $3 \times 3$ homogeneous transformation matrix to apply to the point defined by `(XI, YI)`.
<table>
<thead>
<tr>
<th>FORTRAN Output</th>
<th>PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERRIND</strong></td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td><strong>XO, YO</strong></td>
<td>The coordinates of the transformed point.</td>
</tr>
</tbody>
</table>

**Execution**
TRANSFORM POINT applies the $3 \times 3$ homogeneous *transformation matrix* to the 2D point, and returns the coordinates of the resulting point:

$$\text{transformed point} = \text{matrix} \times \text{point}$$

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

**SEE ALSO**
- SET LOCAL TRANSFORMATION 3 (3P)
- ROTATE (3P)
- SCALE (3P)
- TRANSLATE (3P)
- BUILD TRANSFORMATION MATRIX (3P)
- COMPOSE TRANSFORMATION MATRIX (3P)
- COMPOSE MATRIX (3P)
- TRANSFORM POINT 3 (3P)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>TRANSFORM POINT 3 – apply a 3D transformation matrix to a specified 3D point</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNOPSIS</strong></td>
<td><strong>C Syntax</strong></td>
</tr>
<tr>
<td></td>
<td>void ptran_point3 ( p, m, error_ind, r )</td>
</tr>
<tr>
<td></td>
<td>Ppoint3 *p; point</td>
</tr>
<tr>
<td></td>
<td>Pmatrix3 m; transformation matrix</td>
</tr>
<tr>
<td></td>
<td>Pint *error_ind; OUT error indicator</td>
</tr>
<tr>
<td></td>
<td>Ppoint3 *r; OUT transformed point</td>
</tr>
<tr>
<td><strong>FORTRAN Syntax</strong></td>
<td>SUBROUTINE ptp3 ( XI, YI, ZI, XFRMT, ERRIND, XO, YO, ZO )</td>
</tr>
<tr>
<td></td>
<td>REAL XI, YI, ZI point</td>
</tr>
<tr>
<td></td>
<td>REAL XFRMT(4, 4) transformation matrix</td>
</tr>
<tr>
<td></td>
<td>INTEGER ERRIND OUT error indicator</td>
</tr>
<tr>
<td></td>
<td>REAL XO, YO, ZO OUT transformed point</td>
</tr>
<tr>
<td><strong>Required PHIGS Operating States</strong></td>
<td>(PHOP, *, *, *)</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td></td>
<td>Use TRANSFORM POINT 3 to calculate the transformed coordinates of a 3D point.</td>
</tr>
<tr>
<td><strong>C Input Parameters</strong></td>
<td>p A pointer to a Ppoint3 data structure containing the coordinates of the point to be transformed. Ppoint3 is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef struct {</td>
</tr>
<tr>
<td></td>
<td>Pfloat x; /* x coordinate */</td>
</tr>
<tr>
<td></td>
<td>Pfloat y; /* y coordinate */</td>
</tr>
<tr>
<td></td>
<td>Pfloat z; /* z coordinate */</td>
</tr>
<tr>
<td></td>
<td>} Ppoint3;</td>
</tr>
<tr>
<td></td>
<td>m The 4 × 4 homogeneous transformation matrix to apply to p. Pmatrix3 is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef Pfloat Pmatrix3[4][4];</td>
</tr>
<tr>
<td><strong>C Output Parameters</strong></td>
<td>error_ind A pointer to the location in which to store the error number of any error detected by this function.</td>
</tr>
<tr>
<td></td>
<td>r A pointer to a Ppoint3 structure in which to store the coordinates of the transformed point.</td>
</tr>
</tbody>
</table>

modified 2 April 1993
FORTRAN Input Parameters

\( XI, YI, ZI \)
The coordinates of the point to be transformed.

\( XFRMT \) The 4 × 4 homogeneous transformation matrix to apply to the point defined by \((XI, YI, ZI)\).

FORTRAN Output Parameters

\( ERRIND \) The error number of any error detected by this function.

\( XO, YO, ZO \) The coordinates of the transformed point.

Execution

TRANSFORM POINT applies the 4 × 4 homogeneous transformation matrix to the 3D point and returns the coordinates of the resulting point:

\[
\text{transformed point} = \text{matrix} \times \text{point}
\]

ERRORS

002 Ignoring function, function requires state (\text{PHOP}, *, *, *)

SEE ALSO

SET LOCAL TRANSFORMATION 3 (3P)
ROTATE X (3P)
ROTATE Y (3P)
ROTATE Z (3P)
SCALE 3 (3P)
TRANSLATE 3 (3P)
BUILD TRANSFORMATION MATRIX 3 (3P)
COMPOSE TRANSFORMATION MATRIX 3 (3P)
COMPOSE MATRIX 3 (3P)
TRANSFORM POINT (3P)
NAME
TRANSLATE – calculate a 2D transformation matrix to perform a specified translation

SYNOPSIS
void
pttranslate ( trans_vector, error_ind, m )

Pvec *trans_vector; translation vector
Pint *error_ind; OUT error indicator
Pmatrix m; OUT transformation matrix

FORTRAN Syntax
SUBROUTINE ptr ( DX, DY, ERRIND, XFRMT )
REAL DX, DY translation vector
INTEGER ERRIND OUT error indicator
REAL XFRMT(3, 3) OUT transformation matrix

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use TRANSLATE to generate a 2D homogeneous (3 × 3) transformation matrix that
performs a 2D translation.
The returned matrix can be passed as an argument to SET LOCAL TRANSFORMATION or SET
GLOBAL TRANSFORMATION to modify the modelling transformation that is applied to
output primitives during traversal.

C Input Parameter
trans_vector
A pointer to a Pvec structure containing the Modelling Coordinate (MC)
translation values to be applied in the x and y dimensions. Pvec is defined in
phigs.h as follows:
typedef struct {
    Pfloat delta_x; /* the x axis translation value */
    Pfloat delta_y; /* the y axis translation value */
} Pvec;

C Output Parameters
error_ind
A pointer to the location in which to store the error number of any error detected
by this function.

m
The 3 × 3 homogeneous transformation matrix that performs the specified
translation. Pmatrix is defined in phigs.h as follows:
typedef Pfloat Pmatrix[3][3];

FORTRAN Input Parameter
DX, DY The translation values to be applied in the x and y dimensions.

modified 2 April 1993
<table>
<thead>
<tr>
<th>FORTRAN Output</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error number of any error detected by this function.</td>
</tr>
<tr>
<td>XFRMT</td>
<td>The $3 \times 3$ homogeneous transformation matrix that performs the specified translation.</td>
</tr>
</tbody>
</table>

**Execution**

TRANSLATE returns a 2D homogeneous $(3 \times 3)$ transformation matrix that performs the translation specified by *translation vector*.

The translation vector specifies the translation distance in the $x$ and $y$ directions.

**ERRORS**

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

**SEE ALSO**

- SET LOCAL TRANSFORMATION (3P)
- SET GLOBAL TRANSFORMATION (3P)
- SET VIEW REPRESENTATION (3P)
- ROTATE (3P)
- SCALE (3P)
- BUILD TRANSFORMATION MATRIX (3P)
- COMPOSE TRANSFORMATION MATRIX (3P)
- COMPOSE MATRIX (3P)
- TRANSFORM POINT (3P)
- TRANSLATE 3 (3P)
**NAME**
TRANSLATE 3 – calculate a 3D transformation matrix to perform a specified translation

**SYNOPSIS**

**C Syntax**
```c
void
ptr3 ( trans_vector, error_ind, m )
```
```c
Pvec3 *trans_vector;  // translation vector
Pint *error_ind;      // OUT error indicator
Pmatrix3 m;           // OUT transformation matrix
```

**FORTRAN Syntax**
```fortran
SUBROUTINE ptr3 ( DX, DY, DZ, ERRIND, XFRMT )
```
```fortran
REAL DX, DY, DZ     // translation vector
INTEGER ERRIND      // OUT error indicator
REAL XFRMT(4, 4)    // OUT transformation matrix
```

**Required PHIGS Operating States**
(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**
Use TRANSLATE 3 to generate a 3D homogeneous (4 x 4) transformation matrix that performs a 3D translation.

The returned matrix can be passed as an argument to SET LOCAL TRANSFORMATION 3 or SET GLOBAL TRANSFORMATION 3 to modify the modelling transformation that is applied to output primitives during traversal.

**C Input Parameter**
`trans_vector`
A pointer to a Pvec3 structure containing the Modelling Coordinate (MC) translation values to be applied in the x, y, and z dimensions. Pvec3 is defined in phigs.h as follows:
```c
typedef struct {
    Pfloat delta_x;    /* the x axis translation value */
    Pfloat delta_y;    /* the y axis translation value */
    Pfloat delta_z;    /* the z axis translation value */
} Pvec3;
```

**C Output Parameters**
`error_ind`
A pointer to the location in which to store the error number of any error detected by this function.

`m`
The 4 x 4 homogeneous transformation matrix that performs the specified translation. Pmatrix is defined in phigs.h as follows:
```c
typedef Pfloat Pmatrix3[4][4];
```

modified 2 April 1993
### FORTRAN Input Parameters

- $DX$, $DY$, $DZ$
  
  The translation values to be applied in the $x$, $y$, and $z$ dimensions.

### FORTRAN Output Parameters

- $ERRIND$
  
  The error number of any error detected by this function.

- $XFRMT$ The $4 \times 4$ homogeneous transformation matrix that performs the specified translation.

### Execution

TRANSLATE 3 returns a 3D homogeneous ($4 \times 4$) transformation matrix that performs the translation specified by translation vector.

The translation vector specifies the translation distance in the $x$, $y$, and $z$ directions.

### ERRORS

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

### SEE ALSO

- SET LOCAL TRANSFORMATION 3 (3P)
- SET GLOBAL TRANSFORMATION 3 (3P)
- SET VIEW REPRESENTATION 3 (3P)
- ROTATE X (3P)
- ROTATE Y (3P)
- ROTATE Z (3P)
- SCALE 3 (3P)
- BUILD TRANSFORMATION MATRIX 3 (3P)
- COMPOSE TRANSFORMATION MATRIX 3 (3P)
- COMPOSE MATRIX 3 (3P)
- TRANSFORM POINT 3 (3P)
- TRANSLATE (3P)
UNPACK DATA RECORD – unpack values from a data record into FORTRAN arrays

**SUBROUTINE purec ( LDR, DATREC, IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )**

- **INTEGER** LDR number of array elements used in **DATREC**
- **INTEGER** IIL dimension of integer array
- **INTEGER** IRL dimension of real array
- **INTEGER** ISL dimension of character array
- **INTEGER** ERRIND OUT error indicator (zero if no error)
- **INTEGER** IL OUT number of integer entries
- **INTEGER** IA(IIL) OUT array containing integer entries
- **INTEGER** RL OUT number of real entries
- **REAL** RA(IRL) OUT array containing real entries
- **INTEGER** SL OUT number of character string entries
- **INTEGER** LSTR(ISL) OUT length of each character string entry
- **CHARACTER**(*) STR(ISL) OUT character string entries

**FORTRAN Subset Syntax**

```
SUBROUTINE purec ( LDR, DATREC, IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )
```

- **INTEGER** LDR number of array elements used in **DATREC**
- **INTEGER** IIL dimension of integer array
- **INTEGER** IRL dimension of real array
- **INTEGER** ISL dimension of character array
- **INTEGER** ERRIND OUT error indicator (zero if no error)
- **INTEGER** IL OUT number of integer entries
- **INTEGER** IA(IIL) OUT array containing integer entries
- **INTEGER** RL OUT number of real entries
- **REAL** RA(IRL) OUT array containing real entries
- **INTEGER** SL OUT number of character string entries
- **INTEGER** LSTR(ISL) OUT length of each character string entry
- **CHARACTER**(*) STR(ISL) OUT character string entries

**Required PHIGS Operating States**

\((*, *, *, *)\)

modified 2 April 1993
DESCRIPTION

Purpose

UNPACK DATA RECORD is a PHIGS FORTRAN utility function. It unpacks variable or implementation-dependent data values into FORTRAN arrays from a data record.

FORTRAN Input Parameters

LDR The number of 80-character strings used in the DATREC array.

DATREC The data record is unpacked from this array of 80-character strings.

IIL The dimension of the INTEGER array IA.

IRL The dimension of the REAL array RA.

ISL The dimension of the CHARACTER array STR and of the string length array LSTR.

FORTRAN Output Parameters

ERRIND The error number of any error detected by this function.

IL The number of INTEGER entries in the data record.

IA The IL INTEGER values are unpacked from the data record into this array.

RL The number of REAL entries in the data record.

RA The RL REAL values are unpacked from the data record into this array.

SL The number of character string entries in the data record.

LSTR The lengths of the SL character strings unpacked from the data record into this array.

STR The SL character strings are unpacked from the data record into this array. The FORTRAN subset syntax restricts the STR array to be CHARACTER*80. The STR argument is required, even if the number of strings in the data record is 0.

Execution

UNPACK DATA RECORD unpacks multiple INTEGER, REAL, and CHARACTER string values from a single data record held in contiguous 80-character elements of the DATREC array. The data record DATREC and the INTEGER LDR typically were output parameters from the same call to a PHIGS FORTRAN SUBROUTINE.

This scheme allows the FORTRAN function to return variable or implementation dependent data, in simple fixed-format parameters. An example of a PHIGS FORTRAN function that can return a data record is the FORTRAN ESCAPE function, pesc.

If the data record is invalid, nothing but ERRIND is returned. If any of the IA, RA, or LSTR output arrays is not large enough, error 2001 is issued, and only the counts IL, RL, and SL are returned (unless all three output array sizes were specified as zero; in this case, all counts are returned, but no error is issued). If any string in the data record is longer than an element of the STR output array, the counts and string lengths are returned, and no strings are returned.

ERRORS

2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

380 modified 2 April 1993
| 2003 | FORTRAN: Ignoring function, invalid data record — the data record passed PHIGS routine cannot be decoded, or there was a problem encountered when PHIGS was creating a data record, making the result invalid. |

**SEE ALSO**

PACK DATA RECORD (3P)
UNPOST ALL STRUCTURES

NAME
UNPOST ALL STRUCTURES – unpost all structures posted to the specified workstation

SYNOPSIS
C Syntax
void punpost_all structs ( ws_id )

Pint ws_id;    workstation identifier

FORTRAN Syntax
SUBROUTINE pupast ( WKID )

INTEGER WKID    workstation identifier

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
UNPOST ALL STRUCTURES unposts all of the structures posted to a specified workstation.
Following UNPOST ALL STRUCTURES, no structure networks remain eligible for display on
the workstation. The immediate effect of the function on the display surface is dependent
on the workstation’s display update state.

C Input Parameter
ws_id    All structures are to be unposted from the workstation with this identifier.

FORTRAN Input
Parameter
WKID    All structures are to be unposted from the workstation with this identifier.

Execution
UNPOST ALL STRUCTURES removes all the structures from the table of posted structures on the
specified workstation.
The unposted structures are not deleted from the Central Structure Store (CSS), and can,
again, be posted to this or another workstation. No other workstations are affected.

ERRORS
003    Ignoring function, function requires state (PHOP, WSOP, *, *)
054    Ignoring function, the specified workstation is not open
059    Ignoring function, the specified workstation does not have output capability
(that, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO
POST STRUCTURE (3P)
INQUIRE POSTED STRUCTURES (3P)
CLOSE WORKSTATION (3P)
UNPOST STRUCTURE (3P)
UNPOST STRUCTURE

NAME
UNPOST STRUCTURE – unpost a structure from the specified workstation

SYNOPSIS
C Syntax
void
punpost_struct ( ws_id, struct_id )
Pint ws_id;  /* workstation identifier */
Pint struct_id; /* structure identifier */

FORTRAN Syntax
SUBROUTINE pupost ( WKID, STRID )
INTEGER WKID  /* workstation identifier */
INTEGER STRID  /* structure identifier */

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
UNPOST STRUCTURE unposts a single structure from a workstation. The structure is not
deleted from the Central Structure Store (CSS), but it is no longer eligible for display on
the workstation.
The immediate effect of the function on the display surface depends on the workstation’s
display update state.

C Input Parameters
ws_id  The structure is to be unposted from the workstation with this identifier.
struct_id  The identifier of the structure to be unposted.

FORTRAN Input Parameters
WKID  The structure is to be unposted from the workstation with this identifier.
STRID  The identifier of the structure to be unposted.

Execution
UNPOST STRUCTURE removes the structure specified from the table of posted structures on
the workstation. Unless the structure is part of a structure network posted to the
workstation, the structure is no longer eligible for display on the workstation, and
changes to the structure no longer affect the workstation.
The structure is not deleted from the CSS and can, again, be posted to the same or another
workstation. Other workstations are unaffected.
If structure identifier is not listed in the table of posted structures or does not exist in the CSS
when UNPOST STRUCTURE is called, then the function takes no action.
The effect of unposting the structure may be immediately visible on the workstation’s
display, depending on its display update state.

ERRORS
003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open

modified 2 April 1993
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO

POST STRUCTURE (3P)
INQUIRE POSTED STRUCTURES (3P)
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED (3P)
UNPOST ALL STRUCTURES (3P)
**NAME**
UPDATE WORKSTATION – execute deferred workstation actions and optionally correct the display

**SYNOPSIS**

**C Syntax**

```c
void pupd_ws ( ws, regen_flag )
Pint ws;
Ppregen_flag regen_flag; when to do the regeneration
```

**FORTRAN Syntax**

```fortran
SUBROUTINE puwk ( WKID, REGFL )
INTEGER WKID workstation identifier
INTEGER REGFL regeneration flag (PPOSTP, PPERFO)
```

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**
UPDATE WORKSTATION completes any actions in progress on the specified workstation; then, depending on the value of the regeneration flag and the correctness of the workstation’s display, the display is regenerated to make it CORRECT.

**C Input Parameters**

ws The identifier of the workstation to be updated.
regen_flag The regeneration flag controls whether or not the display should be corrected. Valid values are:
- PFLAG_PERFORM Make picture PVISUAL_ST_CORRECT, if it is not
- PFLAG_POSTPONE Postpone regeneration; complete only deferred workstation actions

**FORTRAN Input Parameters**

WKID The identifier of the workstation to be updated.
REGFL The regeneration flag controls whether or not the display should be corrected. Valid values are:
- PPOSTP Postpone regeneration, complete only deferred workstation actions
- PPERFO Make picture CORRECT, if it is not

**Execution**
UPDATE WORKSTATION always initiates transmission of data buffered from previous actions on the specified workstation, without first clearing the display surface.
UPDATE WORKSTATION may then regenerate the display, depending on the value of the regeneration flag parameter and the correctness of the workstation’s display. If regeneration flag is set to PERFORM, and the workstation’s display is not entirely CORRECT, then UPDATE WORKSTATION regenerates the display, performing the following actions:

modified 2 April 1993
UPDATE WORKSTATION (3P)

1. If the workstation’s display is not empty, then the display surface is cleared.
2. For every view representation in the workstation’s state list, if the view transformation update state is PENDING, then the current view representation is loaded from the requested view representation, and the update state is set to NOTPENDING.
3. If the workstation transformation update state is PENDING, then the current workstation window and current workstation viewport are loaded with the requested values for each; and the update state is set to NOTPENDING.
4. If the workstation’s HLHSR update state is PENDING, then the current HLHSR mode is updated to the value of the requested HLHSR mode; and the update state is set to NOTPENDING.
5. All structure networks posted for this workstation are redisplayed in their priority order.
6. The workstation’s state of visual representation is set to CORRECT.

The workstation’s state of visual representation indicates that the display is CORRECT, SIMULATED, or DEFERRED. INQUIRE DISPLAY UPDATE STATE returns the workstation’s display surface empty and state of visual representation state list entries. Normally, the traversal in step 5 causes the display surface empty workstation state to become NOTEMPTY. (If all the posted structures are empty, or contain elements but no output primitive elements, then the display surface empty may be EMPTY or NOTEMPTY.)

UPDATE WORKSTATION performs the same sequence of actions as REDRAW ALL STRUCTURES when:

- regeneration flag is PERFORM.
- The workstation’s state of visual representation is DEFERRED or SIMULATED.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO

INQUIRE DISPLAY UPDATE STATE (3P)
SET DISPLAY UPDATE STATE (3P)
REDRAW ALL STRUCTURES (3P)
**NAME**
WRITE ITEM TO METAFILE – write application-supplied data to metafile

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pwitm ( WKID, TYPE, IDRL, LDR, DATREC )
INTEGER WKID  // workstation identifier
INTEGER TYPE  // item type
INTEGER IDRL  // number of significant characters in data record
INTEGER LDR  // dimension of data record array
CHARACTER*80 DATREC(LDR)  // data record
```

**Required PHIGS Operating States**

PHOP, WSOP, *, *

**DESCRIPTION**

*Note:* This function has C and FORTRAN bindings, but its functionality is not implemented.

**ERRORS**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>056</td>
<td>Ignoring function, specified workstation is not of category MO</td>
</tr>
<tr>
<td>300</td>
<td>Ignoring function, item type is not allowed for user items</td>
</tr>
<tr>
<td>301</td>
<td>Ignoring function, item length is invalid</td>
</tr>
</tbody>
</table>

**SEE ALSO**

OPEN WORKSTATION (3P)
GET ITEM TYPE FROM METAFILE (3P)
READ ITEM FROM METAFILE (3P)
INTERPRET ITEM (3P)
NAME

INQUIRE ALL CONFLICTING STRUCTURES – obtain all conflicting structures in both Central Structure Store and specified archive file

SYNOPSIS

C Syntax

```c
void pinq_all_conf_structs ( ar_id, length, start, error_ind, ids, total_length )
```

- Pint `ar_id`; archive identifier
- Pint `length`; length of application list
- Pint `start`; starting position
- Pint `*error_ind`; OUT error indicator
- Pint_list `*ids`; OUT list of conflicting structure ids
- Pint `*total_length`; OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqcst ( AFID, N, ERRIND, OL, OSTRID )
```

- INTEGER `AFID` archive file identifier
- INTEGER `N` element of the structure identifier list to return
- INTEGER `ERRIND` OUT error indicator
- INTEGER `OL` OUT number of structure identifiers in list
- INTEGER `OSTRID` OUT Nth structure identifier in list

Required PHIGS Operating States

( PHOP, *, *, AROP )

DESCRIPTION

Purpose

INQUIRE ALL CONFLICTING STRUCTURES obtains a list of the identifiers of all structures which exist in both the Central Structure Store (CSS) and the specified open archive file.

C Input Parameters

- `ar_id` The archive identifier specifying the open archive file to use.
- `length` The number of `ints` items in the `ids` output parameter for which the application has allocated memory. `length` is the number of list elements (structure identifiers) that the system can return in `ints`. If a value of 0 is used here, no data will be returned in the `ints` list, but the total number of conflicting structures will be returned in `total_length`.
- `start` Starting position in the list of identifiers of conflicting structures at which to begin the inquiry. The elements of the list of structure identifiers, beginning with the item number specified by `start`, are copied sequentially into `ints` until `ints` is full or all the structure identifiers have been copied.

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error this function detects.
- `ids` A pointer to a Pint_list structure in which the list of identifiers of conflicting structures is returned. The Pint_list structure is defined in phigs.h as follows:

modified 2 April 1993
typedef struct {
  Pint num_ints;  /* number of Pints in list */
  Pint *ints;    /* list of integers */
} Pint_list;

The num_ints component specifies the number of structure identifiers in the list. The int component is a pointer to a list, num_ints long, of the structure identifiers.

The application must allocate memory for length elements in the ints list.

total_length
A pointer to an integer in which the total number of conflicting structures is returned. This is the value required for length if all structure identifiers are to be returned.

FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFID</td>
<td>The archive identifier specifying the open archive file to use.</td>
</tr>
<tr>
<td>N</td>
<td>Element of the list of identifiers of conflicting structures to return; only one identifier may be inquired per subroutine call. If a value of 0 is used here, no structure identifier will be returned, but the total number of conflicting structures will be returned in OL.</td>
</tr>
</tbody>
</table>

FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error number of any error this function detects.</td>
</tr>
<tr>
<td>OL</td>
<td>The total number of conflicting structures.</td>
</tr>
<tr>
<td>OSTRID</td>
<td>The Nth structure identifier from the list of conflicting structures.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>Ignoring function, function requires state (PHOP, *, *, AROP)</td>
</tr>
<tr>
<td></td>
<td>Ignoring function, the specified archive file is not open</td>
</tr>
</tbody>
</table>

SEE ALSO
INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)

modified 2 April 1993
NAME
INQUIRE ANNOTATION FACILITIES – obtain annotation text facilities on workstation type

SYNOPSIS

C Syntax

```c
void pinq_anno_facs ( type, length, start, error_ind, styles, total_length, num_char_hts,
                     min_char_ht, max_char_ht )
```

- `Pint type;` workstation type
- `Pint length;` length of application list
- `Pint start;` starting position
- `Pint *error_ind;` OUT error indicator
- `Pint_list *styles;` OUT list annotation styles
- `Pint *total_length;` OUT length of list in PHIGS
- `Pint *num_char_hts;` OUT number of character heights
- `Pfloat *min_char_ht;` OUT minimum character height
- `Pfloat *max_char_ht;` OUT maximum character height

FORTRAN Syntax

```fortran
SUBROUTINE pqanf ( WTYPE, N, ERRIND, NAS, AS, NCHH, MINCHH, MAXCHH )
```

- `INTEGER WTYPE` workstation type
- `INTEGER N` list element of annotation styles requested
- `INTEGER ERRIND` OUT error indicator
- `INTEGER NAS` OUT number of available annotation styles
- `INTEGER AS` OUT nth element of list of available annotation styles
- `INTEGER NCHH` OUT number of character heights
- `REAL MINCHH` OUT minimum character height
- `REAL MAXCHH` OUT maximum character height

Required PHIGS Operating States

```plaintext
(PHOP, *, *, *)
```

DESCRIPTION

Purpose

INQUIRE ANNOTATION FACILITIES obtains a list of the annotation text facilities available on the specified workstation type.

C Input Parameters

- `type` Workstation type.
- `length` The number of `ints` items in the `styles` output parameter for which the application has allocated memory. The `length` is the number of list elements (annotation styles) that the system can return in the list of integers in the `Pint_list` structure under `styles`. If a value of zero is used here, no data will be returned in this list, but the total number of annotation styles available on the specified workstation type will be returned in `total_length`.
- `start` Starting position in the workstation type list of available annotation styles at which the inquiry is to begin. The elements of the list of annotation styles, beginning with the item number specified by `start`, are copied sequentially into
the list of integers until it is full or all the annotation styles have been copied.

**C Output Parameters**

**error_ind**
A pointer to the location to store the error number of any error detected by this function.

**styles**
A pointer to a Pint_list structure in which the system returns the list of annotation styles available on the specified type of workstation. Pint_list is defined in phigs.h as follows:

```c
Pint num_ints; /* number of Pints in list */
Pint *ints;    /* list of integers */
}
Pint_list;
```

The `num_ints` component specifies the number of annotation styles in the list. The `ints` component is a pointer to a list, `num_ints` long, of the annotation styles.

The application must allocate memory for `length` elements in the list of `ints`.

The defined values for the annotation styles are:

1. PANNO_STYLE_UNCONNECTED
2. PANNO_STYLE_LEAD_LINE

**total_length**
A pointer to an integer in which the total number of elements in the specified workstation type list of annotation styles is returned. This is the value required for `length` if all annotation styles are to be returned.

**num_char_hts**
A pointer to an integer in which the number of character heights is returned.

**min_char_ht**
A pointer to an integer in which the minimum character height is returned.

**max_char_ht**
A pointer to an integer in which the maximum character height is returned.

**FORTRAN Input Parameters**

**WTYPE** Workstation type.

**N** Element of the specified workstation type list of available annotation styles to return. Only one annotation style may be inquired upon per subroutine call. If a value of zero is used here, no annotation style data will be returned, but the total number of elements in the workstation type list of available annotation styles will be returned in `NAS`.

**FORTRAN Output Parameters**

**ERRIND**
The error number of any error this function detects.

**NAS** The total number of elements in the specified workstation type list of available annotation styles.

modified 2 April 1993
INQUIRE ANNOTATION FACILITIES (3P)

The Nth annotation style from the specified workstation type list of available annotation styles.

NCHH A pointer to an integer in which the number of character heights is returned.

MINCHH A pointer to an integer in which the minimum character height is returned.

MAXCHH A pointer to an integer in which the maximum character height is returned.

ERRORS
002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO
ANNOTATION TEXT RELATIVE (3P)
ANNOTATION TEXT RELATIVE 3 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NAME
INQUIRE ARCHIVE FILES – obtain currently-open archive file identifiers and names

SYNOPSIS
C Syntax

void
pinq_ar_files ( store, error_ind, ar_files )

Pstore store; handle to Store object
Pint *error_ind; OUT error indicator
Par_file_list **ar_files; OUT list of archive file

FORTRAN Syntax

SUBROUTINE pqarf ( N, ERRIND, NUMBER, AFID, ARCFIL )

INTEGER N list element requested
INTEGER ERRIND OUT error indicator
INTEGER NUMBER OUT number of archive files open
INTEGER AFID OUT Nth open archive file identifier
INTEGER ARCFIL OUT Nth open archive file name

Required PHIGS Operating States
( PHOP, *, *, * )

DESCRIPTION
Purpose
Use INQUIRE ARCHIVE FILES to determine the names and identifiers of all currently-open archive files.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by ar_files.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters
error_ind A pointer to the location to store the error number of any error detected by this function.

ar_files A pointer to a structure Par_file_list, containing a list of the currently-open archive file identifiers and names. Par_file_list is defined in phigs.h as follows:
typedef struct{
    Pint num_ar_files; /* number of archive files */
    Par_file *ar_files; /* list of archive files */
} Par_file_list;
The num_ar_files component specifies the number of currently-open archive files.

modified 2 April 1993
The *ar_files* component is a pointer to a list, *num_ar_files* long, of *Par_file* structs listing the identifiers and names of currently-open archive files. *Par_file* is defined in *phigs.h* as follows:

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

The *id* component is the archive file identifier and the *name* component is a pointer to a character string giving the file name of the archive file with identifier *id*.

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>N</em></td>
<td>Element of the list of open archive files to return; only one such element may be inquired per subroutine call. If a value of 0 is used here, no archive file data will be returned, but the total number of open archive files will be returned in <em>NUMBER</em>.</td>
</tr>
</tbody>
</table>

**FORTRAN Output Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ERRIND</em></td>
<td>The error number of any error this function detects.</td>
</tr>
<tr>
<td><em>NUMBER</em></td>
<td>The total number of open archive files.</td>
</tr>
<tr>
<td><em>AFID</em></td>
<td>The <em>N</em>th archive file identifier from the list of currently-open archive files.</td>
</tr>
<tr>
<td><em>ARCFIL</em></td>
<td>The <em>N</em>th archive file logical unit number from the list of currently-open archive files.</td>
</tr>
</tbody>
</table>

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- OPEN ARCHIVE FILE (3P)
- CLOSE ARCHIVE FILE (3P)
- INQUIRE ARCHIVE STATE VALUE (3P)

---

modified 2 April 1993
<table>
<thead>
<tr>
<th>NAME</th>
<th>INQUIRE ARCHIVE STATE VALUE – obtain state value of archive file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td><strong>C Syntax</strong></td>
</tr>
<tr>
<td>void</td>
<td>pinq_ar_st ( archive_state )</td>
</tr>
<tr>
<td>Par_st *archive_state;</td>
<td>OUT archive state</td>
</tr>
<tr>
<td><strong>FORTRAN Syntax</strong></td>
<td>SUBROUTINE pqars ( ARSTA )</td>
</tr>
<tr>
<td>INTEGER ARSTA</td>
<td>OUT archive state value (PARCL, PAROP)</td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(*)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Use INQUIRE ARCHIVE STATE VALUE to determine the current state of the archive file. The state value may be either archive open (AROP) or archive closed (ARCL).</td>
</tr>
<tr>
<td>C Output Parameter</td>
<td>archive_state</td>
</tr>
<tr>
<td>A pointer to a Par_st enumerated variable in which the current state of the archive file is returned. Archive state values are defined in phigs.h as follows:</td>
<td></td>
</tr>
<tr>
<td>PST_ARCL</td>
<td>(archive closed)</td>
</tr>
<tr>
<td>PST_AROP</td>
<td>(archive open)</td>
</tr>
<tr>
<td>FORTRAN Output Parameters</td>
<td>ARSTA An integer in which the current state of the archive file is returned. Archive state values are defined in phigs77.h as follows:</td>
</tr>
<tr>
<td>PARCL</td>
<td>(archive closed)</td>
</tr>
<tr>
<td>PAROP</td>
<td>(archive open)</td>
</tr>
<tr>
<td>ERRORS</td>
<td>No Error</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>OPEN ARCHIVE FILE (3P)</td>
</tr>
<tr>
<td></td>
<td>CLOSE ARCHIVE FILE (3P)</td>
</tr>
<tr>
<td></td>
<td>INQUIRE ARCHIVE FILES (3P)</td>
</tr>
</tbody>
</table>

modified 2 April 1993 395
NAME

INQUIRE CHOICE DEVICE STATE – inquire state of a choice device

SYNOPSIS

void

pinq_choice_st ( ws, dev, store, err, op_mode, echo_switch, init_status, init_choice, prompt_echo, echo_area, choice_data )

Pint ws;    // workstation identi®er
Pint dev;   // choice device number
Pstore store; // OUT handle to Store object
Pint *err;  // OUT error indicator
Pop_mode *op_mode; // OUT operating mode
Pecho_switch *echo_switch; // OUT echo switch
Pin_status *init_status; // OUT initial choice status
Pint *init_choice; // OUT initial choice
Pint *prompt_echo; // OUT prompt/echo type
Plimit *echo_area; // OUT echo area
Pchoice_data **choice_data; // OUT data record

FORTRAN Syntax

SUBROUTINE pqchs ( WKID, CHDNR, MLDR, ERRIND, MODE, ESW, ISTAT, ICHNR, PET, EAREA, LDR, DATREC, DATREC_LEN )

INTEGER WKID           // workstation identi®er
INTEGER CHDNR           // choice device number
INTEGER MLDR            // dimension of data record array
INTEGER ERRIND          // OUT error indicator
INTEGER MODE            // OUT operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW             // OUT echo switch (PNECHO, PECHO)
INTEGER ISTAT           // OUT initial status (POK, PNCHOI)
INTEGER ICHNR           // OUT initial choice number
INTEGER PET             // OUT prompt/echo type
REAL EAREA(4)           // OUT echo area in device coordinates
INTEGER LDR             // OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR) // OUT data record
INTEGER DATREC(MLDR)  // OUT data record

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE CHOICE DEVICE STATE to determine the current state of the specified choice device.
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The choice device data record within the store buffer is accessed via the pointer pointed to by `choice_data`.

- `ws`: Workstation identifier. An integer specifying the workstation with which the specified choice device is associated.
- `dev`: The device number of the choice device. See the `AVAILABLE DEVICES` section of `INITIALIZE CHOICE` for a description of the available devices.
- `store`: The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see `CREATE STORE`).

Output Parameters

- `err`: The error indicator. See the `Execution` section below for a description of its use. See the `ERRORS` section below for the possible values it may return.
- `op_mode`: A pointer to a variable of type `Pop_mode`, which contains the current operating mode of the device. `Pop_mode` is enumerated in `phigs.h` as follows:

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

- `echo_switch`: A pointer to a variable of type `Pecho_switch`, which contains the state of the device’s echo switch. The value returned for `echo_switch` will be either `PSWITCH_ECHO` or `PSWITCH_NO_ECHO`.

- `init_status`: A pointer to a variable of type `Pin_status`, which contains the initial choice status of the device. Possible values for `init_status` are `PIN_STATUS_NONE`, `PIN_STATUS_OK`, or `PIN_STATUS_NO_IN`.

- `init_choice`: A pointer to an integer that contains the value of the initial choice.

- `prompt_echo`: A pointer to an integer that contains the value of the prompt/echo type.

- `echo_area`: A pointer to a variable of type `Plimit` that contains the echo area of the device. `Plimit` is defined in `phigs.h` as follows:

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

choice_data

A pointer to a Pchoice_data pointer. PHIGS assigns this pointer to the location in the Pstore structure that contains the device’s data record contents. Pchoice_data is defined in phigs.h as follows:

typedef union {
    struct {
        Pint unused;
    } pet_r1;

    struct {
        Pint num_prompts; /* number of alternatives */
        Ppr_switch *prompts; /* array of prompts */
    } pet_r2;

    struct {
        Pint num_strings; /* number of choice strings */
        char **strings; /* array of choice strings */
    } pet_r3;

    struct {
        Pint num_strings; /* number of alternatives */
        char **strings; /* array of strings */
    } pet_r4;

    struct {
        Pint struct_id; /* struct identifier */
        Pint num_pick_ids; /* number of alternatives */
        Pint *pick_ids; /* array of pick identifiers */
    } pet_r5;
} Pchoice_data;

Ppr_switch is an enumerated type with the following values:

typedef enum {
    PPR_OFF
    PPR_ON
} Ppr_switch;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the IMLDR argument. The dimension needed is returned in the LDR argument. The caller can
determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial choice status and number, the prompt/echo type and the echo area.

Error 2001 is returned if MLDR is too small, but not if it’s zero.

**WKID**  The workstation identifier of the workstation associated with the device.

**CHDNR**  The device number of the CHOICE device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE for a description of the available devices.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MODE**  The operating mode.

**ESW**  The echo switch.

**ISTAT**  The initial choice status.

**ICHNR**  The initial choice number.

**PET**  The prompt/echo type.

**EAREA**  An array containing the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array.

**DATREC_LEN**  The length of the data record array.

**Execution**  

INQUIRE CHOICE DEVICE STATE returns the current state of the specified choice device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial choice, prompt/echo type, echo area and data record. See SET CHOICE MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE CHOICE for a description of the initial choice, prompt/echo type, echo area and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.
### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not category INPUT or category OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td><strong>FORTRAN:</strong> Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

### SEE ALSO

- **SET CHOICE MODE (3P)**
- **INITIALIZE CHOICE (3P)**
- **INQUIRE CHOICE DEVICE STATE 3 (3P)**

400 modified 2 April 1993
NAME INQUIRE CHOICE DEVICE STATE 3 – inquire state of a choice device

SYNOPSIS
C Syntax

```c
void
pinq_choice_st3 ( ws, dev, store, err, op_mode, echo_switch, init_status, init_choice,
                 prompt_echo, echo_vol, choice_data )
```

- `Pint ws;` workstation identifier
- `Pint dev;` choice device number
- `Pstore store;` OUT pointer to buffer
- `Pint *err;` OUT error indicator
- `Ppop_mode *op_mode;` OUT operating mode
- `Pecho_switch *echo_switch;` OUT echo switch
- `Pin_status *init_status;` OUT initial choice status
- `Pint *init_choice;` OUT initial choice
- `Pint *prompt_echo;` OUT prompt/echo type
- `Plimit3 *echo_vol;` OUT echo volume
- `Pchoice_data3 **choice_data;` OUT data record

FORTRAN Syntax

```fortran
SUBROUTINE pqchs3 ( WKID, CHDNR, MLDNR, ERRIND, MODE, ESW, ISTAT,
                    ICHNR, PET, EVOL(6), LDR, DATREC, DATREC_LEN )
```

- `INTEGER WKID` workstation identifier
- `INTEGER CHDNR` choice device number
- `INTEGER MLDNR` dimension of data record array
- `INTEGER ERRIND` OUT error indicator
- `INTEGER MODE` OUT operating mode (PREQU, PSAMPL, PEVENT)
- `INTEGER ESW` OUT  echo switch (PNECHO, PECHO)
- `INTEGER ISTAT` OUT initial status (POK, PNCHOI)
- `INTEGER ICHNR` OUT initial choice number
- `INTEGER PET` OUT prompt/echo type
- `REAL EVOL(6)` OUT echo volume in device coordinates
- `INTEGER LDR` OUT number of array elements used in data record
- `CHARACTER*80 DATREC(MLDR)` OUT data record
- `INTEGER DATREC_LEN` OUT data record

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE CHOICE DEVICE STATE 3 to determine the current state of the specified choice device.

modified 2 April 1993
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE_STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The choice device data record within the store buffer is accessed via the pointer pointed to by choice_data.

ws Workstation identifier. An integer specifying the workstation with which the specified choice device is associated.

dev The device number of the choice device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE 3 for a description of the available devices.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

err The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**op_mode**
A pointer to a variable of type Pop_mode, which will contain the current operating mode of the device. Pop_mode is enumerated in phigs.h as follows:

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

echo_switch
A pointer to a variable of type Pecho_switch, which will contain the state of the device's echo switch. The value returned for echo_switch will be either PSWITCH_ECHO or PSWITCH_NO_ECHO.

**init_status**
A pointer to a variable of type Pin_status, which contains the initial choice status of the device. Possible values for init_status are PIN_STATUS_OK and PIN_STATUS_NONE.

**init_choice**
A pointer to an integer that contains the value of the initial choice.

**prompt_echo**
A pointer to an integer that contains the value of the prompt/echo type.

**echo_vol**
A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat y_min;
    Pfloat z_min;
    Pfloat x_max;
    Pfloat y_max;
    Pfloat z_max;
} Plimit3;
```
Pfloat x_max; /* maximum x coordinate value */
Pfloat y_min; /* minimum y coordinate value */
Pfloat y_max; /* maximum y coordinate value */
Pfloat z_min; /* minimum z coordinate value */
Pfloat z_max; /* maximum z coordinate value */

A pointer to a Pchoice_data3 pointer. PHIGS assigns this pointer to the location in the Pstore structure that contains the device’s data record contents.

Pchoice_data3 is defined in phigs.h as follows:

typedef union {
  struct {
    Pint unused;
  } pet_r1;
  struct {
    Pint num_prompts; /* number of alternatives */
Ppr_switch *prompts; /* array of prompts */
  } pet_r2;
  struct {
    Pint num_strings; /* number of choice strings */
    char **strings; /* array of choice strings */
  } pet_r3;
  struct {
    Pint num_strings; /* number of alternatives */
    char **strings; /* array of strings */
  } pet_r4;
  struct {
    Pint struct_id; /* struct identifier */
Pint num_pick_ids; /* number of alternatives */
Pint *pick_ids; /* array of pick identifiers */
  } pet_r5;
} Pchoice_data3;

Ppr_switch is an enumerated type with the following values:

typedef enum {
  PPR_OFF,
  PPR_ON
} Ppr_switch;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can
determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial choice status and number, the prompt/echo type and the echo volume.

Error 2001 is returned if MLDR is too small, but not if it’s zero.

**WKID**  The workstation identifier of the workstation associated with the device.

**CHDNR**  The device number of the CHOICE device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE 3 for a description of the available devices.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The *error indicator*. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MODE**  The operating mode.

**ESW**  The echo switch.

**ISTAT**  The initial choice status.

**ICHNR**  The initial choice number.

**PET**  The prompt/echo type.

**EVOL**  An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array.

**DATREC**  The length of the data record array.

**Execution**  INQUIRE CHOICE DEVICE STATE 3 returns the current state of the specified choice device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial choice, prompt/echo type, echo volume and data record. See SET CHOICE MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE CHOICE 3 for a description of the initial choice, prompt/echo type, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the *error indicator* will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the *error indicator* will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.
ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation is neither of category INPUT nor of category OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
2200 C: Buffer overflow in input or inquiry function

FORTRAN:
Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

SET CHOICE MODE (3P)
INITIALIZE CHOICE 3 (3P)
INQUIRE CHOICE DEVICE STATE (3P)

modified 2 April 1993
NAME
INQUIRE COLOUR FACILITIES – obtain workstation type’s colour facilities

SYNOPSIS
void

C Syntax

void
pinq_colr_facs ( type, error_ind, facilities )
Pint type;          workstation type
Pint *error_ind;    OUT error indicator
Pcolr_facs *facilities;  OUT colour facilities

FORTRAN Syntax

SUBROUTINE pqcf ( WTYPE, ERRIND, NCOLI, COLA, NPCI, CC )
INTEGER WTYPE    workstation type
INTEGER ERRIND   OUT error indicator
INTEGER NCOLI    OUT number of colours
INTEGER COLA     OUT colour available (PMONOC, PCOLOR)
INTEGER NPCI     OUT number of predefined colour indices
REAL CC(9)       OUT primary colour chromaticity coefficients and luminance value

Required PHIGS
Operating States (PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE COLOUR FACILITIES to obtain a description of colour facilities for the specified
type of workstation.

C Input Parameter
type    Get the colour facilities for this workstation type.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error this function
detects.

facilities
A pointer to the structure in which to store the information, defined as:

typedef struct {
    Pint num_colrs;        /* number of colours */
    Pcolr_avail colr_avail;    /* colour availability */
    Pint num_pred_inds;    /* number of predefined bundles */
    Pcieluv prim_colrs;        /* primary colours */
} Pcolr_facs;

colr_avail returns the colour availability, one of the following enumeration
values:

PAVAIL_COLR Colour
PAVAIL_MONOCHR Monochrome
num_pred_ind returns the number of predefined colour bundle indices.

prim_cols returns the chromaticity coefficients of the three primary colours (according to the CIE (Commission Internationale de l’Eclairage) universal colour definition system), defined as:

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

The z primary chromaticity coefficients can be calculated by 1.0 - x - y.

Note: SunPHIGS does not support the CIE colour model, so 0.0 is returned for all the chromaticity coefficients.

FORTRAN Input

Parameter

WTYPE Get the colour facilities for this workstation type.

FORTRAN Output

Parameters

ERRIND The error number of any error this function detects.

NCOLI NCOLI is either the number of colours or intensities displayable, or zero to indicate that a continuous range of colours is supported.

COLA COLA is the colour availability, one of the following enumeration values:

    PMONOC   Monochrome
    PCOLOR   Colour

NPCI NPCI is the number of predefined colour bundles.

CC The three primary colours’ CIE Commission Internationale de l’Eclairage’s universal colour definition system); chromaticity coefficients. They are ordered: x coefficient of red, y coefficient of red, x coefficient of green, y coefficient of green, x coefficient of blue, y coefficient of blue. The z primary chromaticity coefficients can be calculated by 1.0 - x - y.

Note: SunPHIGS does not support the CIE colour model, so 0.0 is returned for all the chromaticity coefficients.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062 Ignoring function, this information is not available for this MO workstation type

modified 2 April 1993
SEE ALSO

INQUIRE WORKSTATION CONNECTION AND TYPE (3P)
WORKSTATION TYPE SET (3P)
INQUIRE PREDEFINED COLOUR REPRESENTATION (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
INQUIRE COLOUR MAPPING STATE

NAME
INQUIRE COLOUR MAPPING STATE – inquire the current colour mapping state for a specified workstation

SYNOPSIS
C Syntax

```c
void pinq_colr_map_st ( ws, map_method, error_ind, map_st )
```

Pint ws;  // workstation identifier
Pint map_method;  // mapping method
Pint *error_ind;  // OUT error indicator
Pcolr_map_st *map_st;  // OUT method-specific state

FORTRAN Syntax

```fortran
SUBROUTINE pqcmms ( WTYPE, METH, ERRIND, MAPST )
```

INTEGER WTYPE  // workstation type
INTEGER METH  // colour mapping method of interest
INTEGER ERRIND  // OUT error indicator
INTEGER MAPST  // OUT method-specific state

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE COLOUR MAPPING STATE to determine the currently-available colour mapping facilities for a specified colour mapping method and a specified workstation.

C Input Parameters

`ws`  // The workstation identifier.
`map_method`  // The colour mapping method.

C Output Parameters

`error_ind`  // A pointer to the location to store the error number of any error this function detects.
`map_st`  // A pointer to a location in which the system returns the current colour mapping state for the specified colour mapping method. Pcolr_map_st is defined in phigs.h as:

typedef struct {
    Pint int_data;  // for mapping methods 1 and 3
} Pcolr_map_st;

For colour mapping method PCOLR_MAP_TRUE, `int_data` will contain the number of available true colours. For colour mapping method PCOLR_MAP_PSEUDO, `int_data` will contain the number of pseudo colour entries available. There is no information returned for colour mapping method PCOLR_MAP_PSEUDO_N.

modified 2 April 1993

409
### FORTRAN Parameters

For **METH**, the colour mapping methods defined in phigs77.h are as follows:

1. **PCMAPTRUE**
2. **PCMAPPSEUDO**
3. **PCMAPPSEUDON**

Other parameter descriptions will be provided at a later time.

### Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, **ERROR HANDLING** is not invoked when this function detects an error.

### ERRORS

- **003** Ignoring function, function requires state (*PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT or OUTIN)
- **126** Ignoring function, the specified colour mapping method is not available on the specified workstation.

### SEE ALSO

- SET COLOUR MAPPING REPRESENTATION (3PP)
- INQUIRE COLOUR MAPPING METHOD FACILITIES (3PP)
NAME
INQUIRE COLOUR MODEL – obtain workstation’s current colour model

SYNOPSIS
C Syntax
void
ping_colr_model ( ws, error_ind, model )
Pint ws;          workstation identi®er
Pint *error_ind;  OUT error indicator
Pint *model;      OUT current colour model

FORTRAN Syntax
SUBROUTINE pqcmd ( WKID, ERRIND, CMODEL )
INTEGER WKID    workstation identi®er
INTEGER ERRIND  OUT error indicator
INTEGER CMODEL  OUT current colour model

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE COLOUR MODEL to determine the current colour model from a specified
workstation’s state list.

C Input Parameters
ws           The workstation identi®er of the workstation whose workstation state list is
queried.

C Output Parameters
error_ind     A pointer to the location to store the error number of any error this function
detects.
model         A pointer to a location in which the system returns the current colour model.
The possible colour models are as follows:
    1  PMODEL_RGB
    2  PMODEL_CIELUV
    3  PMODEL_HSV
    4  PMODEL_HLS

FORTRAN Input
Parameters
WKID         The workstation identi®er of the workstation whose state list is queried.

FORTRAN Output
Parameters
ERRIND      The error number of any error this function detects.
CMODEL      The current colour model.

modified 2 April 1993
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
</tbody>
</table>

**SEE ALSO**

INQUIRE COLOUR MODEL FACILITIES (3P)
SET COLOUR MODEL (3P)

modified 2 April 1993
INQUIRE COLOUR MODEL FACILITIES

NAME
INQUIRE COLOUR MODEL FACILITIES – obtain list of workstation colour model facilities

SYNOPSIS
C Syntax

```c
void pinq_colr_model_facs ( type, length, start, error_ind, models, total_length, def )
```

- `Pint  type;`  workstation type
- `Pint  length;`  length of application list
- `Pint  start;`  starting position
- `Pint *error_ind;`  OUT error indicator
- `Pint_list *models;`  OUT list of colour models
- `Pint *total_length;`  OUT length of list in PHIGS
- `Pint *def;`  OUT default colour model

FORTRAN Syntax

```fortran
SUBROUTINE pqcmdf ( WTYPE, N, ERRIND, OL, CMOD, DFCMOD )
```

- `INTEGER WTYPE`  workstation type
- `INTEGER N`  element of list of available colour models
- `INTEGER ERRIND`  OUT error indicator
- `INTEGER OL`  OUT number of available colour models
- `INTEGER CMOD`  OUT Nth available colour model
- `INTEGER DFCMOD`  OUT default colour model

Required PHIGS Operating States

`(PHOP, *, *, *)`

DESCRIPTION

Purpose

Use INQUIRE COLOUR MODEL FACILITIES to obtain a list of the colour model facilities for the specified type of workstation from the workstation description table.

C Input Parameters

- `type`  Get the colour model facilities for this workstation type.
- `length`  The number of elements for which memory is allocated in the output parameter `models`. Zero may be specified in order to get the `total_length` of the list.
- `start`  Starting position in the list at which to begin the inquiry.

C Output Parameters

- `error_ind`  A pointer to the location to store the error number of any error this function detects.
- `models`  A pointer to a `Pint_list` which returns the portion of the list of available colour models from the workstation description table, starting with `start`. `Pint_list` is defined in phigs.h as:

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

modified 2 April 1993
Pint_list;

The pointer to the list of integers must be initialized to an array of length Pint elements.

total_length

A pointer to an integer in which to return the total length of the list. This is the value required for length if all the items in the list are to be returned.

def

A pointer to an integer in which to return the default colour model.

FORTRAN Input Parameters

WTYPE Get the colour model facilities for this workstation type.

N Get the Nth element from the list of available colour models.

FORTRAN Output Parameters

ERRIND The error number of any error detected by this function.

OL The total length of the list of available colour models.

CMOD The Nth colour model from the list of available colour models.

DFCMOD The default colour model.

ERRORS

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET COLOUR MODEL (3P)

INQUIRE COLOUR FACILITIES (3P)

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NAME
INQUIRE COLOUR REPRESENTATION – obtain colour representation on workstation

SYNOPSIS
C Syntax
void
pinq_colr_rep ( ws, index, type, error_ind, rep )
Pint ws;  // workstation identifier
Pint index;  // colour index
Pinq_type type;  // type of returned value
Pint *error_ind;  // OUT error indicator
Pcolr_rep *rep;  // OUT colour representation

FORTRAN Syntax
SUBROUTINE pqcr ( WKID, COLI, CCSBSZ, TYPE, ERRIND, OL, SPEC )
INTEGER WKID  // workstation identifier
INTEGER COLI  // colour index
INTEGER CCSBSZ  // colour component specification buffer size
INTEGER TYPE  // type of returned values (PSET, PREALI)
INTEGER ERRIND  // OUT error indicator
INTEGER OL  // number of colour components in the colour specification
REAL SPEC(CCSBSZ)  // colour specification

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE COLOUR REPRESENTATION to determine the current representation in a specified colour table entry from a specified workstation’s state list.

C Input Parameters
ws  The workstation identifier of the workstation whose state list is queried.
index  Entry to be returned from the workstation’s colour table; if this entry is not present in the table and the type of returned value parameter is Pinq_REALIZED, the representation for colour index 1 is returned.
type  An enumerated variable specifying whether the value desired is the value specified by the application program or the available value used by the workstation. Valid values are:

Pinq_SET  Return the values exactly as they are specified in the application program.
Pinq_REALIZED  Return the values as they are used by the workstation when the values specified in the application program are mapped to values available on the workstation.

modified 2 April 1993
C Output Parameters

\textit{error\_ind}

A pointer to the location to store the error number of any error this function detects.

\textit{rep}

A pointer to a structure in which the system returns the colour representation at entry \textit{index}. \texttt{Pcolr\_rep} is defined in \texttt{phigs.h} as follows:

typedef union {
    Prgb rrgb; /* Red Green Blue colour specification */
    Pcieluv cieluv; /* CIE L*U*V* colour specification */
    Phls hls; /* Hue Lightness Saturation colour specification */
    Phsv hsv; /* Hue Saturation Value colour specification */
    Pdata unsupp; /* Colour in unsupported colour model */
} Pcolr\_rep;

Prgb is defined in \texttt{phigs.h} as follows:

typedef struct {
    Pfloat red; /* red, hue, and so on */
    Pfloat green; /* green, saturation, lightness, and so on */
    Pfloat blue; /* blue, value, saturation, and so on */
} Prgb;

Pcieluv is defined in \texttt{phigs.h} as follows:

typedef struct {
    Pfloat cieluv\_x; /* x coefficient */
    Pfloat cieluv\_y; /* y coefficient */
    Pfloat cieluv\_y\_lum; /* y luminance */
} Pcieluv;

Phsv is defined in \texttt{phigs.h} as follows:

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

Phls is defined in \texttt{phigs.h} as follows:

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

Pdata is defined in phigs.h as follows:
typedef struct {
    size_t size; /* size of data */
    char *data /* pointer to data */
} Pdata;

FORTRAN Input Parameters

WKID The workstation identifier of the workstation whose state list is being queried.

COLI Entry to be returned from the workstation’s colour table; if this entry is not present in the table, and the type of returned value parameter is PREALI, the representation for colour index 1 is returned.

CCSBSZ The size for the SPEC buffer in which to return the colour specification.

TYPE An enumerated variable specifying whether the value desired is the value specified by the application program or the available value used by the workstation. Valid values are:

PSET Return the values exactly as they are specified in the application program.

PREALI Return the values as they are used by the workstation when the values specified in the application program are mapped to values available on the workstation.

FORTRAN Output Parameters

ERRIND Location to store the error number of any error this function detects.

OL The number of colour components in the colour specification.

SPEC The colour specification.

ERRORS 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
        054 Ignoring function, the specified workstation is not open
        059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
        113 Ignoring function, the colour index value is less than zero
        101 Ignoring function, the specified representation has not been defined

SEE ALSO INQUIRE COLOUR FACILITIES (3P)
        SET COLOUR REPRESENTATION (3P)
        INQUIRE PREDEFINED COLOUR REPRESENTATION (3P)

modified 2 April 1993
NAME
INQUIRE CONFLICT RESOLUTION – obtain current values of conflict resolution flags

SYNOPSIS
C Syntax

void
pinq_conf_res ( error_ind, archive, retrieval )

Pint *error_ind; /\ OUT error indicator
Pconf_res *archive; /\ OUT archival resolution
Pconf_res *retrieval; /\ OUT retrieval resolution

FORTRAN Syntax

SUBROUTINE pqcnrs ( ERRIND, ARCCR, RETCR )

INTEGER ERRIND /\ OUT error indicator
INTEGER ARCCR /\ OUT archival conflict resolution
INTEGER RETCR /\ OUT retrieval conflict resolution

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION
Purpose

Use INQUIRE CONFLICT RESOLUTION to determine the current values of the archival and retrieval conflict resolution flags. The flag values may be MAINTAIN, ABANDON, or UPDATE.

See the description of the subroutine SET CONFLICT RESOLUTION for information on the meaning of these values.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error this function detects.

archive
A pointer to a Pconf_res enumerated type in which the system returns the current value of the archival conflict resolution flag.

retrieval
A pointer to a Pconf_res enumerated type in which the system returns the current value of the retrieval conflict resolution flag. Values for the Pconf_res enumerated type are defined in phigs.h as follows:

PRES_MAINTAIN
PRES_ABANDON
PRES_UPD

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

ARCCR
The current value of the archival conflict resolution flag.

RETCR
The current value of the retrieval conflict resolution flag. The conflict resolution flag values are defined in phigs77.h as follows:
PCRMNT   Maintain
PCRABA   Abandon
PCRUPD   Update

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

SEE ALSO  SET CONFLICT RESOLUTION (3P)

modified 2 April 1993
INQUIRE CONFLICTING STRUCTURES IN NETWORK

NAME
INQUIRE CONFLICTING STRUCTURES IN NETWORK – obtain conflicting network structures in Central Structure Store and specified archive file.

SYNOPSIS

C Syntax

```c
void pinq_conf_structs_net ( ar_id, struct_id, src, length, start, error_ind, ids, total_length )
```

- `Pint ar_id;` archive identifier
- `Pint struct_id;` structure identifier
- `Pstruct_net_source src;` structure network source
- `Pint length;` length of application list
- `Pint start;` starting position
- `Pint *error_ind;` OUT error indicator
- `Pint_list *ids;` OUT conflicting struct id list
- `Pint *total_length;` OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqcstn ( AFID, STRID, SNSRC, N, ERRIND, OL, OSTRID )
```

- `INTEGER AFID` archive file identifier
- `INTEGER STRID` structure identifier
- `INTEGER SNSRC` structure network source (PCSS, PARCHV)
- `INTEGER N` element of the structure identifier list to return
- `INTEGER ERRIND` OUT error indicator
- `INTEGER OL` OUT number of structure identifiers in list
- `INTEGER OSTRID` OUT Nth structure identifier in list

Required PHIGS Operating States

(HP, *, *, AROP)

DESCRIPTION

Purpose
INQUIRE CONFLICTING STRUCTURES IN NETWORK obtains a list of the identifiers of all structures in a specified network that exist in both the Central Structure Store (CSS) and the specified open archive file.

C Input Parameters

- `ar_id` The archive identifier specifying the open archive file to use.
- `struct_id` The identifier of the root structure of the network.
- `src` The structure network source, defined in phigs.h as:
  - `PNET_CSS` Central Structure Store
  - `PNET_AR` Archive
- `length` The number of integers items in the `ids` output parameter for which the application has allocated memory. `length` is the number of list elements (structure identifiers) that the system can return in `ids->integers`. If a value of 0 is used here, no data will be returned in the `ids->integers` list, but the total number of conflicting structures will be returned in `total_length`.

420 modified 2 April 1993
**INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)**

- **start**
  Starting position in the list of identifiers of conflicting structures at which to begin the inquiry. The elements of the list of structure identifiers, beginning with the item number specified by *start*, are copied sequentially into *ids-*→*integers* until *ids-*→*integers* is full or all the structure identifiers have been copied.

- **C Output Parameters**
  - **error_ind**
    A pointer to the location to store the error number of any error this function detects.
  - **ids**
    A pointer to a Pint_list structure in which the list of identifiers of conflicting structures is returned. The Pint_list structure is defined in phigs.h as follows:
    ```c
    typedef struct {
        Pint num_ints; /* number of Pints in list */
        Pint *ints; /* list of integers */
    } Pint_list;
    ```
    The *num_ints* component specifies the number of structure identifiers in the list. The *ints* component is a pointer to a list, *num_ints* long, of the structure identifiers. The application must allocate memory for *length* elements in the *ids-*→*integers* list.
  - **total_length**
    A pointer to an integer in which the total number of conflicting structures is returned. This is the value required for *length* if all structure identifiers are to be returned.

- **FORTRAN Input Parameters**
  - **AFID**
    The archive identifier specifying the open archive file to use.
  - **STRID**
    The identifier of the root structure of the network.
  - **SNSRC**
    The structure network source, defined in phigs77.h as:
    ```
    PCSS Central Structure Store
    PARCHV Archive
    ```
  - **N**
    Element of the list of identifiers of conflicting structures to return; only one identifier can be queried per subroutine call. If a value of 0 is used here, no structure identifier will be returned, but the total number of conflicting structures will be returned in *OL*.

- **FORTRAN Output Parameters**
  - **ERRIND**
    The error number of any error this function detects.
  - **OL**
    The total number of conflicting structures.
  - **OSTRID**
    The Nth structure identifier from the list of conflicting structures.

- **Execution**
  If the structure network source is the Central Structure Store, the identifiers of all structures in the network defined by the root *structure identifier* in the Central Structure Store are compared to the identifiers of all structures in the specified archive file, and

Modified 2 April 1993
those which are in both are returned as conflicting structures. If the structure network source is the archive file, the identifiers of all structures in the network defined by the root structure identifier in the specified archive file are compared to the identifiers of all structures in the Central Structure Store, and those which are in both are returned as conflicting structures.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>Ignoring function, function requires state (PHOP, *, *, AROP)</td>
</tr>
<tr>
<td>201</td>
<td>Ignoring function, the specified structure does not exist</td>
</tr>
<tr>
<td>404</td>
<td>Ignoring function, the specified archive file is not open</td>
</tr>
</tbody>
</table>

**SEE ALSO**

INQUIRE ALL CONFLICTING STRUCTURES (3P)
NAME  
INQUIRE CURRENT ELEMENT CONTENT – obtain contents of current element

SYNOPSIS  
C Syntax

void

pinq_cur_elem_content ( store, error_ind, data )

Pstore store;  handle to Store object
Pint *error_ind;  OUT error indicator
Pelem_data **data;  OUT data record

FORTRAN Syntax

SUBROUTINE pqceco ( IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )

INTEGER IIL  dimension of integer array
INTEGER IRL  dimension of real array
INTEGER ISL  dimension of character array
INTEGER ERRIND  OUT error indicator
INTEGER IL  OUT number of integer entries
INTEGER IA(IIL)  OUT array containing integer entries
INTEGER RL  OUT number of real entries
REAL RA(IRL)  OUT array containing real entries
INTEGER SL  OUT number of character string entries
INTEGER LSTR(ISL)  OUT length of each character string entry
CHARACTER*80 STR(ISL)  OUT character string entries

FORTRAN Subset Syntax

SUBROUTINE pqceco ( IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )

INTEGER IIL  dimension of integer array
INTEGER IRL  dimension of real array
INTEGER ISL  dimension of character array
INTEGER ERRIND  OUT error indicator
INTEGER IL  OUT number of integer entries
INTEGER IA(IIL)  OUT array containing integer entries
INTEGER RL  OUT number of real entries
REAL RA(IRL)  OUT array containing real entries
INTEGER SL  OUT number of character string entries
INTEGER LSTR(ISL)  OUT length of each character string entry
CHARACTER*80 STR(ISL)  OUT character string entries

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION  
Purpose

INQUIRE CURRENT ELEMENT CONTENT determines the contents of the current element.

C Input Parameter

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

modified 2 April 1993
The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by data.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind A pointer to the location to store the error number of any error this function detects.

data A pointer to a Pelem_data union, in which the contents of the current element will be returned, as appropriate for the element type. See INQUIRE ELEMENT CONTENT (3P) for a description of Pelem_data.

FORTRAN Input Parameters

IIL Dimension of integer array IA in which the current element integer data will be returned. The appropriate array size for the data to be returned may be obtained by calling INQUIRE CURRENT ELEMENT TYPE AND SIZE and using the value returned in its IL parameter.

IRL Dimension of real array RA in which the current element real data will be returned. The appropriate array size for the data to be returned may be obtained by calling INQUIRE CURRENT ELEMENT TYPE AND SIZE and using the value returned in its RL parameter.

ISL Dimension of integer array LSTR and character array STR in which the current element character data will be returned. The appropriate array size for the data to be returned may be obtained by calling INQUIRE CURRENT ELEMENT TYPE AND SIZE and using the value returned in its SL parameter.

FORTRAN Output Parameters

ERRIND The error number of any error this function detects.

IL The number of entries returned in the IA array.

IA The integer values contained in the current element.

RL The number of entries returned in the RA array.

RA The real values contained in the current element.

SL The number of entries returned in the LSTR and STR arrays.

LSTR SL integers specifying the lengths of the SL character strings returned in STR.

STR The character data contained in the current element.

The contents of the various arrays are determined by the element type. See INQUIRE ELEMENT CONTENT (3P) for a description of the contents of the arrays.
Errors

005 Ignoring function, function requires state (PHOP, *, STOP, *)

See Also

inquire_current_element_content.3
INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
UNPACK DATA RECORD (3P)
INQUIRE ELEMENT CONTENT (3P)
### NAME
INQUIRE CURRENT ELEMENT TYPE AND SIZE – obtain type and size of current element

### SYNOPSIS

**C Syntax**

```c
void pinq_cur_elem_type_size ( error_ind, type, size )
Pint *error_ind;  OUT error indicator
Pelem_type *type;  OUT element type
size_t *size;  OUT element size
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pqcets ( ERRIND, ELTYPE, IL, RL, SL )
INTEGER ERRIND  OUT error indicator
INTEGER ELTYPE  OUT element type
INTEGER IL  OUT dimension of integer array (this can be passed to PQCECO as ILL)
INTEGER RL  OUT dimension of real array (this can be passed to PQCECO as IRL)
INTEGER SL  OUT dimension of character array (this can be passed to PQCECO as ISL)
```

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION

**Purpose**
INQUIRE CURRENT ELEMENT TYPE AND SIZE determines the type and size of the current element.

**C Output Parameters**

- **error_ind**
  A pointer to the location to store the error number of any error this function detects.

- **type**
  Returns the type of the current element. For example, a polyline primitive element would be returned as PELEM_POLYLINE, a character height attribute as PELEM_CHARACTER_HEIGHT, a modelling transformation as PELEM_LOCAL_MODELLING_TRANSFORMATION3, and so forth. See INQUIRE ELEMENT TYPE AND SIZE (3P) for a description of the Pelem_type enumerated type.

- **size**
  Returns the size, in bytes, that the application will have to allocate in order to call INQUIRE CURRENT ELEMENT CONTENT to retrieve the contents of the current element. If the current element type is such that it is not necessary to allocate any dynamic memory to retrieve its contents, a value of zero is returned.
FORTRAN Output Parameters

ERRIND
   The error number of any error this function detects.

ELTYPE
   Returns the type of the current element. For example, a polyline primitive
element would be returned as PEPL, a character height attribute as PECHH, a local
modelling transformation as PELMT, and so forth. The FORTRAN definitions for
element types are in phigs77.h. See INQUIRE ELEMENT TYPE AND SIZE (3P) for a
description of mapping from the six-character FORTRAN definitions to the actual
PHIGS element type names.

IL
   Returns the dimension required for the integer array argument to INQUIRE
CURRENT ELEMENT CONTENT in order to retrieve the current element contents.

RL
   Returns the dimension required for the real array argument to INQUIRE CURRENT
ELEMENT CONTENT in order to retrieve the current element contents.

SL
   Returns the dimension required for the character array argument to INQUIRE
CURRENT ELEMENT CONTENT in order to retrieve the current element contents.

ERRORS
   005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
   INQUIRE CURRENT ELEMENT CONTENT (3P)
   INQUIRE ELEMENT POINTER (3P)
   ELEMENT SEARCH (3P)
   INQUIRE ELEMENT TYPE AND SIZE (3P)
NAME
INQUIRE DEFAULT CHOICE DEVICE DATA – inquire the predefined choice data

SYNOPSIS
void

C Syntax

pinq_def_choice_data ( type, device, store, error_ind, max_choices, pets, echo_area,
choice_data )

Pint type; workstation type
Pint device; logical input device number
Pstore store; handle to Store object
Pint *error_ind; OUT error indicator
Pint *max_choices; OUT max. number of choices
Pint_list **pets; OUT list of prompt and echo types
Plimit *echo_area; OUT default echo area
Pchoice_data ***choice_data; OUT pointer to default data record

FORTRAN Syntax

SUBROUTINE pqdch ( WTYPE, DEVNO, N, MLDR, ERRIND, MALT, OL, PET,
EAREA, LDR, DATREC )

INTEGER WTYPE workstation type
INTEGER DEVNO logical input device number
INTEGER N list element requested
INTEGER MLDR dimension of data record array
INTEGER ERRIND OUT error indicator
INTEGER MALT OUT maximum number of alternatives
INTEGER OL OUT number of available prompt/echo types
INTEGER PET OUT Nth element of list of available prompt/echo types
REAL EAREA(4) OUT default echo area in device coordinates
INTEGER LDR OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR) OUT data record

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE DEFAULT CHOICE DEVICE DATA to determine the following information for a
choice input device associated with a given workstation type:
Number and list of available prompt/echo types.
Default echo area.
Default input data record.
Maximum number of available choices.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The choice device data record within the store buffer is accessed via the pointer returned in choice_data. The prompt/echo type list is accessed in the store buffer via the pointer returned in pets.

- **type**  
  The workstation type with which the device is associated.

- **device**  
  The device number of the choice device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE for a description of the available devices.

- **store**  
  The memory buffer PHIGS is to use for storing the information returned for the PChoice_data structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

- **error_ind**  
  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

- **max_choices**  
  Maximum number of choices.

- **pets**  
  A list of the available prompt/echo types. Pint_list is defined in phigs.h as follows:

  ```c
  typedef struct {
      Pint    num_ints;    /* number of Pints in list */
      Pint    *ints;      /* list of integers */
  } Pint_list;
  
  The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

- **echo_area**  
  A pointer a Plimit structure to contain the default echo area. Plimit is defined in phigs.h as follows:

  ```c
  typedef struct {
      Pfloat  x_min;    /* x min */
      Pfloat  x_max;    /* x max */
      Pfloat  y_min;    /* y min */
      Pfloat  y_max;    /* y max */
  } Plimit;
  
  A pointer to a Plimit structure.

  ```c
  typedef struct {
      Pint    num_ints;    /* number of Pints in list */
      Pint    *ints;      /* list of integers */
  } Pint_list;
  
  The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

- **choice_data**  
  A pointer to a Pchoice_data pointer. PHIGS assigns this pointer to the location in
typedef struct {
  union Pchoice_pets {
    struct Pchoice_pet_r1 {
      Pint unused;
    } pet_r1;
    struct Pchoice_pet_r2 {
      Pint num_prompts; /* number of alternatives */
      Ppr_switch *prompts; /* array of prompts */
    } pet_r2;
    struct Pchoice_pet_r3 {
      Pint num_strings; /* number of choice strings */
      char **strings; /* array of choice strings */
    } pet_r3;
    struct Pchoice_pet_r4 {
      Pint num_strings; /* number of alternatives */
      char **strings; /* array of strings */
    } pet_r4;
    struct Pchoice_pet_r5 {
      Pint struct_id; /* struct identifier */
      Pint num_pick_ids; /* number of alternatives */
      Pint *pick_ids; /* array of pick identifiers */
    } pet_r5;
  } pets;
} Pchoice_data;

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the default input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo area, and the maximum number of available choices. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  The workstation type with which the device is associated.

modified 2 April 1993
**DEVNO**
The device number of the CHOICE device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE for a description of the available devices.

**N**
The index of the prompt/echo type list entry to return.

**MLDR**
The dimension of the data record array, DATREC.

**ERRIND**
The *error indicator*. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MALT**
The maximum number of choice alternatives available.

**OL**
The number of available prompt/echo types.

**PET**
The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EAREA**
An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**
The required dimension of the data record array, DATREC.

**DATREC**
The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

---

**Execution**
INQUIRE DEFAULT CHOICE DEVICE DATA returns the default data of the specified choice device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE CHOICE for a description of the prompt/echo types, echo area and data record contents and how to set these values.

If an error is detected by this function, except in the cases described in the C and FORTRAN Parameters subsections above, then the *error indicator* indicates the error number of the error detected and no other output data is returned. If no error is detected, then the error indicator is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

---

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation’s category is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
</tbody>
</table>

modified 2 April 1993
FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

INITIALIZE CHOICE (3P)
INQUIRE DEFAULT CHOICE DEVICE DATA 3 (3P)
NAME

INQUIRE DEFAULT CHOICE DEVICE DATA 3 – inquire the predefined choice data

SYNOPSIS

C Syntax

```c
void pinq_def_choice_data3 (type, device, store, error_ind, max_choices, pets, echo_vol, choice_data )
```

- `Pint type;` workstation type
- `Pint device;` logical input device number
- `Pstore store;` handle to Store object
- `Pint *error_ind;` OUT error indicator
- `Pint *max_choices;` OUT max. number of choices
- `Pint_list **pets;` OUT list of prompt and echo types
- `Plimit3 *echo_vol;` OUT default echo volume
- `Pchoice_data3 **choice_data;` OUT default data record

FORTRAN Syntax

```fortran
SUBROUTINE pqdch3 ( WTYPE, DEVNO, N, MLDR, ERRIND, MALT, OL, PET, EVOL, LDR, DATREC )
```

- `INTEGER WTYPE` workstation type
- `INTEGER DEVNO` logical input device number
- `INTEGER N` list element requested
- `INTEGER MLDR` dimension of data record array
- `INTEGER ERRIND` OUT error indicator
- `INTEGER MALT` OUT maximum number of alternatives
- `INTEGER OL` OUT number of available prompt/echo types
- `INTEGER PET` OUT Nth element of list of available prompt/echo types
- `REAL EVOL(6)` OUT default echo volume in device coordinates
- `INTEGER LDR` OUT number of array elements used in data record
- `CHARACTER*80 DATREC(MLDR)` OUT data record

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use INQUIRE DEFAULT CHOICE DEVICE DATA 3 to determine the following information for a CHOICE input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.
- Maximum number of available choices.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

modified 2 April 1993
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The choice device data record within the store buffer is accessed via the pointer returned in choice_data. The prompt/echo type list is accessed in the store buffer via the pointer returned in pets.

type  The workstation type with which the device is associated.

device The device number of the choice device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE 3 for a description of the available devices.

store  The memory buffer PHIGS is to use for storing the information returned for the Pchoice_data3 structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

max_choices Maximum number of choices.

pets  List of available prompt/echo types. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

echo_vol A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

typedef struct {
    Pfloat  x_min;       /* minimum x coordinate value */
    Pfloat  x_max;       /* maximum x coordinate value */
    Pfloat  y_min;       /* minimum y coordinate value */
    Pfloat  y_max;       /* maximum y coordinate value */
    Pfloat  z_min;       /* minimum z coordinate value */
    Pfloat  z_max;       /* maximum z coordinate value */
} Plimit3;

choice_data A pointer to a Pchoice_data3 pointer. PHIGS assigns this pointer to the location in
the Pstore structure that contains the device’s data record contents.

Pchoice_data3 is defined in phigs.h as follows:

```c
typedef struct {
    union Pchoice_pets {
        struct Pchoice_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pchoice_pet_r2 {
            Pint num_prompts; /* number of alternatives */
            Ppr_switch *prompts; /* array of prompts */
        } pet_r2;
        struct Pchoice_pet_r3 {
            Pint num_strings; /* number of choice strings */
            char **strings; /* array of choice strings */
        } pet_r3;
        struct Pchoice_pet_r4 {
            Pint num_strings; /* number of alternatives */
            char **strings; /* array of strings */
        } pet_r4;
        struct Pchoice_pet_r5 {
            Pint struct_id; /* struct identifier */
            Pint num_pick_ids; /* number of alternatives */
            Pint *pick_ids; /* array of pick identifiers */
        } pet_r5;
    } pets;
} Pchoice_data3;
```

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the contents of the default input data record are placed. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo volume, and the maximum number of available choices. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE** The workstation type with which the device is associated.
**DEVNO**
The device number of the CHOICE device. See the AVAILABLE DEVICES section of INITIALIZE CHOICE 3 for a description of the available devices.

**N**
The index of the prompt/echo type list entry to return.

**MLDR**
The dimension of the data record array, DATREC.

**ERRIND**
The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MALT**
The maximum number of choice alternatives available.

**OL**
The number of available prompt/echo types.

**PET**
The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EVL**
An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**
The required dimension of the data record array, DATREC.

**DATREC**
The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**
INQUIRE DEFAULT CHOICE DEVICE DATA 3 returns the default data of the specified choice device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE CHOICE 3 for a description of the prompt/echo types, echo volume and data record contents and how to set these values.

If this function detects an error, except in the cases described in the C and FORTRAN Parameters sections above, then the error indicator indicates the error number of the detected error and no other output data is returned. If no error is detected, then the error indicator is set to zero and the information inquired for is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**
002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
061 Ignoring function, specified workstation’s category is not INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
2200 C: Buffer overflow in input or inquiry function
2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the

modified 2 April 1993
returned value.

SEE ALSO

INITIALIZE CHOICE 3 (3P)
INQUIRE DEFAULT CHOICE DEVICE DATA (3P)
NAME
INQUIRE DEFAULT DISPLAY UPDATE STATE – inquire the default display update state for a specified workstation type

SYNOPSIS
C Syntax
void
pinq_def_disp_upd_st ( type, error_ind, def_mode, mod_mode )

FORTRAN Syntax
SUBROUTINE pqddus ( WTYPE, ERRIND, DEFMOD, MODMOD )

C Input Parameters
type Type of workstation.

C Output Parameters
error_ind A pointer to the location to store the error number of any error this function detects.
def_mode The deferral mode, def_mode, is defined in phigs.h as:

typedef enum {
PDEFER_ASAP, As Soon As Possible
PDEFER_BNIG, Before the Next Interaction Globally
PDEFER_BNIL, Before the Next Interaction Locally
PDEFER_ASTI, At Some Time
PDEFER_WAIT, When the Application Requests It
} Pdefer_mode;

DESCRIPTION
Purpose
Use INQUIRE DEFAULT DISPLAY UPDATE STATE to determine the default display update state for a specified workstation type. The display update state consists of a deferral mode and a modification mode.

See the description of the subroutine SET DISPLAY UPDATE STATE for information on the meaning of these modes.

Required PHIGS Operating States
(PHOP, *, *, *)

modified 2 April 1993
mod_mode

The modification mode, mod_mode, is defined in phigs.h as:

typedef enum {
    PMODE_NIVE, // No immediate visual effects mandated
    PMODE_UWOR, // Update without regeneration
    PMODE_UQUM // Use quick update methods
} Pmod_mode;

FORTRAN Input Parameters

WTYPE Type of workstation.

FORTRAN Output Parameters

ERRIND The error number of any error this function detects.

DEFMOD The default deferral mode for the specified workstation type. Deferral mode values are enumerated values defined in phigs77.h as:

0 PASAP, // As soon as possible
1 PBNIG, // Before the next interaction globally
2 PBNIL, // Before the next interaction locally
3 PASTI, // At some time
4 PWAITD // When the application requests it

MODMOD The default modification mode for the specified workstation type. Modification mode values are enumerated values defined in phigs77.h as:

0 PNIVE, // No immediate visual effects mandated
1 PUWOR, // Update without regeneration
2 PUQUM // Use quick update methods

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not recognized by the implementation. This information is not available for this workstation type; open a workstation of this type and use the specific workstation type
062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET DISPLAY UPDATE STATE (3P)
INQUIRE DISPLAY UPDATE STATE (3P)
INQUIRE DEFAULT LOCATOR DEVICE DATA – inquire the predefined locator data

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pqdlc ( WTYPE, DEVNO, N, MLDR, ERRIND, DPX, DPY, OL, PET, EAREA, LDR, DATREC )
INTEGER WTYPE  // workstation type
INTEGER DEVNO  // logical input device number
INTEGER N  // list element requested
INTEGER MLDR  // dimension of data record array
INTEGER ERRIND  // OUT error indicator
REAL DPX, DPY  // OUT default initial locator position
INTEGER OL  // OUT number of available prompt/echo types
INTEGER PET  // OUT Nth element of list of available prompt/echo types
REAL EAREA(4)  // OUT default echo area in device coordinates
INTEGER LDR  // OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR)  // OUT data record
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

**Purpose**

Use INQUIRE DEFAULT LOCATOR DEVICE DATA to determine the following information for a LOCATOR input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo area.
- Default input data record.
- Default initial locator position.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.
C Input Parameters

Applications using the C binding must create a buffer for this function to use as memory space for storing portions of the device data. This buffer is passed as the `store` argument. The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The locator device data record within the store buffer is accessed via the pointer returned in `loc_data`. The prompt/echo type list is accessed in the store buffer via the pointer returned in `pets`.

- **type**: A Pint value specifying the workstation type with which the device is associated.
- **device**: The device number of the locator device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.
- **store**: The memory buffer PHIGS is to use for storing the information returned for the Ploc_data structure and the prompt/echo type. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

- **error_ind**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.
- **loc_pos**: The default initial locator position.
- **pets**: The list of prompt and echo types. Pint_list is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

- **echo_area**: A pointer to an object of type Plimit that contains the echo area of the device. Plimit is defined in phigs.h as follows:

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

- **loc_data**: A pointer to a Ploc_data pointer. PHIGS assigns this pointer to the location in the Pstore structure that contains the device’s data record contents. Ploc_data is defined in phigs.h as follows:

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

typedef Ploc_data Ploc_data3;

FORTRAN Input

Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the contents of the default input data record are placed. The application subsequently extracts the contents of the data record by using the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.
Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo area, and the default initial locator position. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The calling program indicates the element to return via the argument N.

**TYPE**
The workstation type with which the device is associated.

**DEVNO**
The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.

**N**
The index of the prompt/echo type list entry to return.

**MLDR**
The dimension of the data record array, DATREC.

**ERRIND**
The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**DPX, DPY**
The default initial locator position.

**OL**
The number of available prompt/echo types.

**PET**
The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EAREA**
An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**
The required dimension of the data record array, DATREC.

**DATREC**
The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

---

**Execution**

INQUIRE DEFAULT LOCATOR DEVICE DATA returns the default data of the specified locator device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE LOCATOR for a description of the prompt/echo types, echo area and data record contents and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN Parameters sections above, then the error indicator indicates the error number of the detected error, and no other output data is returned. If no error is detected, then the error indicator is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.
### ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation category is not (INPUT) or (OUTIN)</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td>FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

### SEE ALSO

- INITIALIZE LOCATOR (3P)
- INQUIRE DEFAULT LOCATOR DEVICE DATA 3 (3P)
NAME
INQUIRE DEFAULT LOCATOR DEVICE DATA 3 – inquire the predefined locator data

SYNOPSIS
C Syntax

```c
void
pinq_def_loc_data3 ( type, device, store, error_ind, loc_pos, pets, echo_vol, loc_data )
```

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

FORTRAN Syntax
SUBROUTINE pqdlc3 ( WTYPE, DEVNO, N, MLDR, ERRIND, DPX, DPY, DPZ, OL, PET, EVOL, LDR, DATREC )

| INTEGER | WTYPE | workstation type |
| INTEGER | DEVNO | logical input device number |
| INTEGER | N | list element requested |
| INTEGER | MLDR | dimension of data record array |
| INTEGER | ERRIND | OUT error indicator |
| REAL | DPX, DPY, DPZ | OUT default initial locator position |
| INTEGER | OL | OUT number of available prompt/echo types |
| INTEGER | PET | OUT Nth element of list of available prompt/echo types |
| REAL | EVOL(6) | OUT default echo volume in device coordinates |
| INTEGER | LDR | OUT number of array elements used in data record |
| CHARACTER=80 | DATREC(MLDR) | OUT data record |

Required PHIGS Operating States
( PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE DEFAULT LOCATOR DEVICE DATA 3 to determine the following information for a LOCATOR input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.
- Default initial locator position.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.
Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The locator device data record within the store buffer is accessed via the pointer returned in `loc_data`. The prompt/echo type list is accessed in the store buffer via the pointer returned in `pets`.

- **type**: The workstation type with which the device is associated.
- **device**: The device number of the locator device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.
- **store**: The memory buffer PHIGS is to use for storing the information returned for the Ploc_data3 structure and the prompt/echo type. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

- **error_ind**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.
- **loc_pos**: Default initial locator position.
- **pets**: List of available prompt/echo types. Pint_list is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

- **echo_vol**: A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
```

- **loc_data**: A pointer to a Ploc_data3 pointer. PHIGS assigns this pointer to the location in the Pstore structure that contains the device’s data record contents. Ploc_data3 is

modified 2 April 1993
defined in phigs.h as follows:

typedef struct {
  union Ploc_pets {
    struct Ploc_pet_r1 {
      Pint unused;
    } pet_r1;
    struct Ploc_pet_r2 {
      Pint unused;
    } pet_r2;
    struct Ploc_pet_r3 {
      Pint unused;
    } pet_r3;
    struct Ploc_pet_r4 {
      Pline_attrs line_attrs; /* polyline attributes */
    } pet_r4;
    struct Ploc_pet_r5 {
      Pline_fill_ctrl_flag line_fill_ctrl_flag;
    } pet_r5;
    struct Ploc_pet_u2 {
      Pint crosshair_colr;
    } pet_u2;
    struct Ploc_pet_u4 {
      Pline_bundle line_bundle;
    } pet_u4;
    struct Ploc_pet_u5 {
      Pline_bundle line_bundle;
    } pet_u5;
  } pets;
} Ploc_data;

typedef Ploc_data Ploc_data3;

FORTRAN Parameters

An application using the FORTRAN binding must supply a CHARACTER array to this function, into which the contents of the default input data record are put. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can
determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo volume, and the default initial locator position. Error 2001 is returned if MLDR is too small, but not if it’s zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The calling program indicates the element to return via the argument N.

**TYPE** The workstation type with which the device is associated.

**DEVNO**

The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

**N** The index of the prompt/echo type list entry to return.

**MLDR** The dimension of the data record array, DATREC.

**ERRIND**

The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**DPX, DPY, DPZ**

The default initial locator position.

**OL** The number of available prompt/echo types.

**PET** The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EVOL** An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR** The required dimension of the data record array, DATREC.

**DATREC**

The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**

INQUIRE DEFAULT LOCATOR DEVICE DATA 3 returns the default data of the specified locator device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE LOCATOR 3 for a description of the prompt/echo types, echo volume and data record contents and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN Parameters sections above, then the error indicator indicates the error number of the detected error and no other output data is returned. If it does not detect an error, then the error indicator is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation category is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td>FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INITIALIZE LOCATOR 3 (3P)
- INQUIRE DEFAULT LOCATOR DEVICE DATA (3P)
NAME
INQUIRE DEFAULT PICK DEVICE DATA – inquire the predefined pick data

SYNOPSIS

C Syntax

```c
void inquire_default_pick_data ( type, device, store, error_ind, pets, echo_area, pick_data )

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

FORTRAN Syntax

```fortran
SUBROUTINE pqdpk ( WTYPE, DEVNO, N, MLDR, ERRIND, OL, PET, EAREA, LDR, DATREC )

INTEGER WTYPE          /* workstation type */
INTEGER DEVNO          /* logical input device number */
INTEGER N              /* list element requested */
INTEGER MLDR           /* dimension of data record array */
INTEGER ERRIND         /* OUT error indicator */
INTEGER OL             /* OUT number of available prompt/echo types */
INTEGER PET            /* OUT Nth element of list of available prompt/echo types */
REAL EAREA(4)          /* OUT default echo area in device coordinates */
INTEGER LDR            /* OUT number of array elements used in data record */
CHARACTER*80 DATREC(MLDR) /* OUT data record */
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose
Use INQUIRE DEFAULT PICK DEVICE DATA to determine the following information for a PICK input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo area.
- Default input data record.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

modified 2 April 1993
The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The pick device data record within the store buffer is accessed via the pointer returned in \textit{pick\_data}. The prompt/echo type list is accessed in the store buffer via the pointer returned in \textit{pets}.

- \textit{type}: The workstation type with which the device is associated.
- \textit{device}: The device number of the pick device. See the \textit{AVAILABLE DEVICES} section of INITIALIZE PICK for a description of the available devices.
- \textit{store}: The memory buffer PHIGS is to use for storing the information returned for the \texttt{Ppick\_data} structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

- \textit{error\_ind}: The error indicator. See the \textit{Execution} section below for a description of its use. See the \textit{ERRORS} section below for the possible values it can return.
- \textit{pets}: Available prompt/echo types. \texttt{Pint\_list} is defined in phigs.h as follows:

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

  The \texttt{num\_ints} component specifies the number of elements in the list. The \texttt{ints} component is a pointer to a list \texttt{num\_ints} long.

- \textit{echo\_area}: A pointer to an object of type \texttt{Plimit} that will represent the echo area of the device. \texttt{Plimit} is defined in phigs.h as follows:

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

- \textit{pick\_data}: A pointer to a pointer to a \texttt{Ppick\_data} pointer. PHIGS assigns this pointer to the location in the \texttt{Pstore} structure that contains the device’s data record contents. \texttt{Ppick\_data} is defined in phigs.h as follows:

  /* \texttt{Ppick\_data} -- pick data record */
  typedef struct {
    union Ppick_pets {
      struct Pick_pet_r1 {
      }
    }
  } Ppick_data;
struct Pick_pet_r2 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_r2;

struct Pick_pet_r3 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_r3;

struct Pick_pet_u1 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_u1;

struct Pick_pet_u2 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_u2;

struct Pick_pet_u3 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_u3;

struct Pick_pet_u4 {
Pint  highl_colr;
Pint  highl_count;
Pfloat highl_duration;
Pfloat ap_size;  /* aperture size, half-width in DC units */
} pet_u4;

} pets;

} Ppick_data;

FORTRAN Parameters
Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which is placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD.
function. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR too small, including the case of its being zero, some values will be returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, and the echo area. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  The workstation type with which the device is associated.

**DEVNO**  The device number of the PICK device. See the AVAILABLE DEVICES section of INITIALIZE PICK for a description of the available devices.

**N**  The index of the prompt/echo type list entry to return.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**OL**  The number of available prompt/echo types.

**PET**  The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EAREA**  An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**  INQUIRE DEFAULT PICK DEVICE DATA returns the default data of the specified pick device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE PICK for a description of the prompt/echo types, echo area and data record contents and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN Parameters sections above, the error indicator indicates the error number of the error detected and no other output data is returned. If no error is detected, the error indicator is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.
**ERRORS**

002 Ignoring function, function requires state \((\text{PHOP}, *, *, *)\)

052 Ignoring function, workstation type not recognized by the implementation

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

061 Ignoring function, category of the specified workstation not \(\text{INPUT}\) or \(\text{OUTIN}\)

250 Ignoring function, the specified device is not available on the specified workstation

2200 \(C\): Buffer overflow in input or inquiry function

2001 *FORTRAN*: Ignoring function, output parameter size insufficient — a *FORTRAN* array or string being passed as an output parameter is too small to contain the returned value

**SEE ALSO**

INITIALIZE PICK (3P)

INQUIRE DEFAULT PICK DEVICE DATA 3 (3P)
NAME

INQUIRE DEFAULT PICK DEVICE DATA 3 – inquire the predefined pick data

SYNOPSIS

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pqdpk3 ( WTYPE, DEVNO, N, MLD, ERRIND, OL, PET, EVOL, LDR, DATREC )
INTEGER WTYPE /* workstation type */
INTEGER DEVNO /* logical input device number */
INTEGER N /* list element requested */
INTEGER MLD /* dimension of data record array */
INTEGER ERRIND /* OUT error indicator */
INTEGER OL /* OUT number of available prompt/echo types */
INTEGER PET /* OUT Nth element of list of available prompt/echo types */
REAL EVOL(6) /* OUT default echo volume in device coordinates */
INTEGER LDR /* OUT number of array elements used in data record */
CHARACTER*80 DATREC(MLD) /* OUT data record */
```

DESCRIPTION

Purpose

Use INQUIRE DEFAULT PICK DEVICE DATA 3 to determine the following information for a PICK input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

modified 2 April 1993 455
The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The pick device data record within the store buffer is accessed via the pointer returned in `pick_data`. The prompt/echo type list is accessed in the store buffer via the pointer returned in `pets`.

- **type**: The workstation type with which the device is associated.
- **device**: The device number of the pick device. See the AVAILABLE DEVICES section of INITIALIZE PICK 3 for a description of the available devices.
- **store**: The memory buffer PHIGS is to use for storing the information returned for the `Ppick_data3` structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

- **error_ind**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

- **pets**: Available prompt/echo types. `Pint_list` is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

- **echo_vol**: A pointer to a `Plimit3` structure defining the x, y, and z components of the echo volume, in Device Coordinates. `Plimit3` is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
```

- **pick_data**: A pointer to a `Ppick_data3` pointer. PHIGS assigns this pointer to the location in the `Pstore` structure that contains the device’s data record contents. `Ppick_data3` is defined in phigs.h as follows:

```c
typedef struct {
    union Ppick_pets {
        struct Pick_pet_r1 {
            Pint unused;
        }
    }
} Ppick_data3;
```

456 modified 2 April 1993
} pet_r1;
struct Pick_pet_r2 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_r2;
struct Pick_pet_r3 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_r3;
struct Pick_pet_u1 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u1;
struct Pick_pet_u2 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u2;
struct Pick_pet_u3 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u3;
struct Pick_pet_u4 {
  Pint highl_colr;
  Pint highl_count;
  Pfloat highl_duration;
  Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u4;
} pets;
} Ppick_data;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which is placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument.
The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, and the echo volume. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  The workstation type with which the device is associated.

**DEVNO**  The device number of the PICK device. See the AVAILABLE DEVICES section of INITIALIZE PICK 3 for a description of the available devices.

**N**  The index of the prompt/echo type list entry to return.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**OL**  The number of available prompt/echo types.

**PET**  The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**Evol**  An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**

INQUIRE DEFAULT PICK DEVICE DATA 3 returns the default data of the specified pick device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE PICK 3 for a description of the prompt/echo types, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator indicates the error number of the detected error, and no other output data is returned. If no error is detected, then the error indicator is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.
ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type;
open a workstation of this type and use the specific workstation type
061 Ignoring function, specified workstation category is not INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified
workstation
2200 C: Buffer overflow in input or inquiry function
2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN
array or string being passed as an output parameter is too small to contain the
returned value.

SEE ALSO

INITIALIZE PICK 3 (3P)
INQUIRE DEFAULT PICK DEVICE DATA (3P)

modified 2 April 1993
NAME
INQUIRE DEFAULT STRING DEVICE DATA – inquire the predefined string data

SYNOPSIS
C Syntax

void
pinq_def_string_data ( type, device, store, error_ind, max_buf_size, pets, echo_area, 
string_data )

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

FORTRAN Syntax

SUBROUTINE pqdst ( WTYPE, DEVNO, N, MLDR, ERRIND, MBUFF, OL, 
PET, EAREA, LDR, DATREC )

INTEGER WTYPE workstation type
INTEGER DEVNO logical input device number
INTEGER N list element requested
INTEGER MLDR dimension of data record array
INTEGER ERRIND OUT error indicator
INTEGER MBUFF OUT maximum string buffer size
INTEGER OL OUT number of available prompt/echo types
INTEGER PET OUT Nth element of list of available prompt/echo types
REAL EAREA(4) OUT default echo area in device coordinates
INTEGER LDR OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR) OUT data record

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE DEFAULT STRING DEVICE DATA to determine the following information for a STRING input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo area.
- Default input data record.
- Maximum string buffer size.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.
**C Input Parameters**

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, PHIGS manages this area to provide sufficient memory for the specific inquiry. The string device data record within the store buffer is accessed via the pointer returned in `string_data`. The prompt/echo type list is accessed in the store buffer via the pointer returned in `pets`.

- **type**: The workstation type with which the device is associated.
- **device**: The device number of the string device. See the `AVAILABLE DEVICES` section of `INITIALIZE STRING` for a description of the available devices.
- **store**: The memory buffer PHIGS is to use for storing the information returned for the `Pstring_data` structure and the prompt/echo type list. This buffer must exist prior to calling this function (see `CREATE STORE (3P)`).

**C Output Parameters**

- **error_ind**: The error indicator. See the `Execution` section below for a description of its use. See the `ERRORS` section below for the possible values it can return.
- **max_buf_size**: Maximum input buffer size.
- **pets**: Available prompt/echo types. `Pint_list` is defined in `phigs.h` as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

- **echo_area**: A pointer to an object of type `Plimit` that represents the echo area of the device. `Plimit` is defined in `phigs.h` as follows:

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

- **string_data**: A pointer to a `Pstring_data` pointer. PHIGS assigns this pointer to the location in the `Pstore` structure that contains the device’s data record contents. `Pstring_data` modified 2 April 1993
is defined in phigs.h as follows:

typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos;   /* initial editing position */
} Pstring_data;

FORTRAN Input Parameters

An application using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the default echo area, the default edit position, the default buffer size, and the maximum string buffer size. Error 2001 is returned if MLDR is too small, but not if it's zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  The workstation type with which the device is associated.

**DEVNO**  The device number of the STRING device. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available devices.

**N**  The index of the prompt/echo type list entry to return.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**MBUFF**  The maximum string buffer size.

**OL**  The number of available prompt/echo types.

**PET**  The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.
INQUIRE DEFAULT STRING DEVICE DATA

**EAREA**  An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**

The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**

INQUIRE DEFAULT STRING DEVICE DATA returns the default data of the specified string device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE STRING for a description of the prompt/echo types, echo area and data record contents and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN Parameters sections above, then the error number of the detected error, and no other output data, is returned. If no error is detected, then the error indicator is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, category of the specified workstation is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td>FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

INITIALIZE STRING (3P)

INQUIRE DEFAULT STRING DEVICE DATA 3 (3P)
**NAME**
INQUIRE DEFAULT STRING DEVICE DATA 3 – inquire the predefined string data

**SYNOPSIS**

C Syntax

```c
void pinq_def_string_data3 ( type, device, store, error_ind, max_buf_size, pets, echo_vol,
string_data );
```

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

FORTRAN Syntax

```fortran
SUBROUTINE pqdst3 ( WTYPE, DEVNO, N, MLDR, ERRIND, MBUFF, OL, PET, EVOL, LDR, DATREC )
```

- INTEGER `WTYPE` workstation type
- INTEGER `DEVNO` logical input device number
- INTEGER `N` list element requested
- INTEGER `MLDR` dimension of data record array
- INTEGER `ERRIND` OUT error indicator
- INTEGER `MBUFF` OUT maximum string buffer size
- INTEGER `OL` OUT number of available prompt/echo types
- INTEGER `PET` OUT Nth element of list of available prompt/echo types
- REAL `EVOL(6)` OUT default echo volume in device coordinates
- INTEGER `LDR` OUT number of array elements used in data record
- CHARACTER*80 `DATREC(MLDR)` OUT data record

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE DEFAULT STRING DEVICE DATA 3 to determine the following information for a STRING input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.
- Maximum string buffer size.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

464 modified 2 April 1993
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The string device data record within the store buffer is accessed via the pointer returned in string_data. The prompt/echo type list is accessed in the store buffer via the pointer returned in pets.

type  The workstation type with which the device is associated.
device The device number of the string device. See the AVAILABLE DEVICES section of INITIALIZE STRING 3 for a description of the available devices.
store  The memory buffer PHIGS is to use for storing the information returned for the Pstring_data3 structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind

The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

max_buf_size

Maximum input buffer size.
pets

Available prompt/echo types. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

echo_vol

A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min;  /* minimum x coordinate value */
    Pfloat x_max;  /* maximum x coordinate value */
    Pfloat y_min;  /* minimum y coordinate value */
    Pfloat y_max;  /* maximum y coordinate value */
    Pfloat z_min;  /* minimum z coordinate value */
    Pfloat z_max;  /* maximum z coordinate value */
} Plimit3;

string_data

A pointer to a Pstring_data3 pointer. PHIGS assigns this pointer to the location in...
the Pstore structure that contains the device’s data record contents. 

Pstring_data3 is defined in phigs.h as follows:

typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */

    union {
        struct Pstring_pet_r1 {
            Pint unused;
        } pet_r1;
    } pets;
} Pstring_data3;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the default echo volume, the default edit position, the default buffer size, and the maximum string buffer size. Error 2001 is returned if MLDR is too small, but not if it’s zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

TYPE The workstation type with which the device is associated.

DEVNO The device number of the STRING device. See the AVAILABLE DEVICES section of INITIALIZE STRING 3 for a description of the available devices.

N The index of the prompt/echo type list entry to return.

MLDR The dimension of the data record array, DATREC.

ERRIND The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

MBUFF The maximum string buffer size.

OL The number of available prompt/echo types.

PET The prompt/echo type corresponding to the Nth position in the list of
prompt/echo types.

**EVOL**  An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

**Execution**  INQUIRE DEFAULT STRING DEVICE DATA 3 returns the default data of the specified string device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE STRING 3 for a description of the prompt/echo types, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator indicates the error number of the detected error, and no other output data is returned. If no error is detected, then the error indicator is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

**ERRORS**

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

052  Ignoring function, workstation type not recognized by the implementation

051  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

061  Ignoring function, category of the specified workstation is not INPUT or OUTIN

250  Ignoring function, the specified device is not available on the specified workstation

2200  C: Buffer overflow in input or inquiry function

2001  FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

**SEE ALSO**

INITIALIZE STRING 3 (3P)

INQUIRE DEFAULT STRING DEVICE DATA (3P)
NAME
INQUIRE DEFAULT STROKE DEVICE DATA – inquire the predefined stroke data

SYNOPSIS
C Syntax
void
pinq_def_stroke_data ( type, device, store, error_ind, max_buf_size, pets, echo_area,
stroke_data )

FORTRAN Syntax
SUBROUTINE pqdsk ( WTYPE, DEVNO, N, MLDR, ERRIND, MBUFF, OL,
PET, EAREA, LDR, DATREC )

DESCRIPTION
Purpose
Use INQUIRE DEFAULT STROKE DEVICE DATA to determine the following information for a
STROKE input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo area.
- Default input data record.
- Maximum stroke buffer size.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The stroke device data record within the store buffer is accessed via the pointer returned in stroke_data. The prompt/echo type list is accessed in the store buffer via the pointer returned in pets.

type  The workstation type with which the device is associated.

device  The device number of the stroke device. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.

store  The memory buffer PHIGS is to use for storing the information returned for the Pstroke_data structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

max_buf_size  Maximum buffer size.

pets  A list of the available prompt/echo type. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

echo_area  A pointer to an object of type Plimit that contains the echo area of the device. Plimit is defined in phigs.h as follows:

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

stroke_data  A pointer to a Pstroke_data pointer. PHIGS will assign this pointer to the location

modified 2 April 1993 469
in the Pstore structure that contains the device's data record contents.
Pstroke_data is defined in phigs.h as follows:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos;    /* initial editing position */
    Pfloat x_interval; /* x interval */
    Pfloat y_interval; /* y interval */
    Pfloat time_interval; /* time interval */
    union {
        struct Pstroke_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pstroke_pet_r2 {
            Pint unused;
        } pet_r2;
        struct Pstroke_pet_r3 {
            Pmarker_attrs marker_attrs; /* marker attributes */
        } pet_r3;
        struct Pstroke_pet_r4 {
            Pline_attrs line_attrs; /* line attributes */
        } pet_r4;
        struct Pstroke_pet_u3 {
            Pmarker_bundle marker_bundle;
        } pet_u3;
        struct Pstroke_pet_u4 {
            Pline_bundle line_bundle;
        } pet_u4;
    } pets;
} Pstroke_data;
```

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the `UNPACK DATA RECORD` function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo area, the default stroke buffer size, and the maximum stroke buffer size. Error 2001 is returned if MLDR is too small, but not if it's zero.
The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  The workstation type with which the device is associated.

**DEVNO**  The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.

**N**  The index of the prompt/echo type list entry to return.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**MBUFF**  The maximum stroke buffer size.

**OL**  The number of available prompt/echo types.

**PET**  The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EAREA**  An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

### Execution

INQUIRE DEFAULT STROKE DEVICE DATA returns the default data of the specified stroke device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE STROKE for a description of the prompt/echo types, echo area and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator indicates the error number of the detected error, and no other output data is returned. If no error is detected, the error indicator is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

### ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, category of the specified workstation is not INPUT or OUTIN</td>
</tr>
</tbody>
</table>

modified 2 April 1993
250 Ignoring function, the specified device is not available on the specified workstation
2200 C: Buffer overflow in input or inquiry function
2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

INITIALIZE STROKE (3P)
INQUIRE DEFAULT STROKE DEVICE DATA 3 (3P)
NAME

INQUIRE DEFAULT STROKE DEVICE DATA 3 – inquire the predefined stroke data

SYNOPSIS

C Syntax

```c
void
pinq_def_stroke_data3( type, device, store, error_ind, max_buf_size, pets,
    echo_volume, stroke_data )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pint</td>
<td>workstation type</td>
</tr>
<tr>
<td>Pint</td>
<td>logical input device number</td>
</tr>
<tr>
<td>Pstore</td>
<td>handle to Store object</td>
</tr>
<tr>
<td>Pint</td>
<td>OUT error indicator</td>
</tr>
<tr>
<td>Pint</td>
<td>OUT max. input buffer size</td>
</tr>
<tr>
<td>Pint_list</td>
<td>OUT list of prompt and echo types</td>
</tr>
<tr>
<td>Plimit3</td>
<td>OUT default echo volume</td>
</tr>
<tr>
<td>Pstroke_data3</td>
<td>OUT default data record</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

```fortran
SUBROUTINE pqdsk3 ( WTYPE, DEVNO, N, MLDR, ERRIND, MBUFF, OL,
    PET, EVOL, LDR, DATREC )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>workstation type</td>
</tr>
<tr>
<td>INTEGER</td>
<td>logical input device number</td>
</tr>
<tr>
<td>INTEGER</td>
<td>list element requested</td>
</tr>
<tr>
<td>INTEGER</td>
<td>dimension of data record array</td>
</tr>
<tr>
<td>INTEGER</td>
<td>OUT error indicator</td>
</tr>
<tr>
<td>INTEGER</td>
<td>OUT maximum input buffer size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>OUT number of available prompt/echo types</td>
</tr>
<tr>
<td>INTEGER</td>
<td>OUT Nth element of list of available prompt/echo types</td>
</tr>
<tr>
<td>REAL</td>
<td>OUT default echo volume in device coordinates</td>
</tr>
<tr>
<td>INTEGER</td>
<td>OUT number of array elements used in data record</td>
</tr>
<tr>
<td>CHARACTER*80</td>
<td>OUT data record</td>
</tr>
</tbody>
</table>

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose

Use INQUIRE DEFAULT STROKE DEVICE DATA 3 to determine the following information for a STROKE input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.
- Maximum stroke buffer size.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

modified 2 April 1993
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE_STORE, the implementation manages this area to provide sufficient memory for the specific inquiry. The stroke device data record within the store buffer is accessed via the pointer pointed to by stroke_data. The prompt/echo type list is accessed in the store buffer via the pointer returned in pets.

type
The workstation type with which the device is associated.
device
The device number of the stroke device. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available devices.
store
The memory buffer PHIGS is to use for storing the information returned for the Pstroke_data3 structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE_STORE (3P)).

C Output Parameters

error_ind
The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

max_buf_size
Maximum buffer size.
pets
Default prompt/echo types. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of elements in the list. The ints component is a pointer to a list num_ints long.

echo_volume
A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat    x_min;     /* minimum x coordinate value */
    Pfloat    x_max;     /* maximum x coordinate value */
    Pfloat    y_min;     /* minimum y coordinate value */
    Pfloat    y_max;     /* maximum y coordinate value */
    Pfloat    z_min;     /* minimum z coordinate value */
    Pfloat    z_max;     /* maximum z coordinate value */
} Plimit3;
```

stroke_data
A pointer to a pointer to a Pstroke_data3 structure. Pstroke_data3 is defined in

474 modified 2 April 1993
phigs.h as follows:

typedef struct {
Pint buffer_size; /* input buffer size */
Pint init_pos; /* initial editing position */
Pfloat x_interval; /* x interval */
Pfloat y_interval; /* y interval */
Pfloat z_interval; /* z interval */
Pfloat time_interval; /* time interval */
union {
struct Pstroke3_pet_r1 {
Pint unused;
} pet_r1;
} pet_r1;
struct Pstroke3_pet_r2 {
Pint unused;
} pet_r2;
} pet_r2;
struct Pstroke3_pet_r3 {
Pmarker_attrs marker_attrs; /* marker attributes */
} pet_r3;
} pet_r3;
struct Pstroke3_pet_r4 {
pline_attrs line_attrs; /* line attributes */
} pet_r4;
} pet_r4;
struct Pstroke3_pet_u3 {
Pmarker_bundle marker_bundle;
} pet_u3;
} pet_u3;
struct Pstroke3_pet_u4 {
pline_bundle line_bundle;
} pet_u4;
} pets;
Pstroke_data3;

FORTRAN Input

Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument; the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo volume, the default stroke buffer size, and the maximum stroke buffer size. Error 2001 is returned if MLDR is too small, but not if it is zero.

modified 2 April 1993
The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

**TYPE**  
The Workstation type with which the device is associated.

**DEVNO**  
The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available devices.

**N**  
The index of the prompt/echo type list entry to return.

**MLDR**  
The dimension of the data record array, DATREC.

**ERRIND**  
The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**MBUFF**  
The maximum stroke buffer size.

**OL**  
The number of available prompt/echo types.

**PET**  
The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EVOl**  
An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  
The required dimension of the data record array, DATREC.

**DATREC**  
The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

### Execution

INQUIRE DEFAULT STROKE DEVICE DATA 3 returns the default data of the specified stroke device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE STROKE 3 for a description of the prompt/echo types, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator indicates the error number of the detected error detected, and no other output data is returned. If no error is detected, then the error indicator is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

### ERRORS

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

052  Ignoring function, workstation type not recognized by the implementation

051  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

061  Ignoring function, category of specified workstation not INPUT or OUTIN

476  modified 2 April 1993
Ignoring function, the specified device is not available on the specified workstation.

C: Buffer overflow in input or inquiry function.

FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

INITIALIZE STROKE 3 (3P)

INQUIRE DEFAULT STROKE DEVICE DATA (3P)

modified 2 April 1993
NAME
INQUIRE DEFAULT VALUATOR DEVICE DATA – inquire the predefined valuator data

SYNOPSIS
C Syntax

void
pinq_def_val_data ( type, device, store, error_ind, def_value, pets, echo_area,
val_data )

Pint type; /* workstation type */
Pint device; /* logical input device number */
Pstore store; /* handle to Store object */
Pint *error_ind; /* OUT error indicator */
Pfloat *def_value; /* OUT default initial value */
Pint_list **pets; /* OUT list of prompt and echo types */
Plimit *echo_area; /* OUT default echo volume */
Pval_data **val_data; /* OUT pointer to default data record */

FORTRAN Syntax

SUBROUTINE pqdvl ( WTYPE, DEVNO, N, MLDR, ERRIND, DVAL, OL,
PET, EAREA, LDR, DATREC )

INTEGER WTYPE /* workstation type */
INTEGER DEVNO /* logical input device number */
INTEGER N /* list element requested */
INTEGER MLDR /* dimension of data record array */
INTEGER ERRIND /* OUT error indicator */
REAL DVAL /* OUT default initial value */
INTEGER OL /* OUT number of available prompt/echo types */
INTEGER PET /* OUT Nth element of list of available prompt/echo types */
REAL EAREA(4) /* OUT default echo area in device coordinates */
INTEGER LDR /* OUT number of array elements used in data record */
CHARACTER*80 DATREC(MLDR) /* OUT data record */

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE DEFAULT VALUATOR DEVICE DATA to determine the following information for a VALUATOR input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo area.
- Default input data record.
- Default initial valuator value.

478 modified 2 April 1993
Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

**C Input Parameters**

Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, PHIGS manages this area to provide sufficient memory for the specific inquiry. The valuator device data record within the store buffer is accessed via the pointer returned in `val_data`.

- **type**  
  The workstation type with which the device is associated.

- **device**  
  The device number of the valuator device. See the `AVAILABLE DEVICES` section of `INITIALIZE VALUATOR` for a description of the available devices.

- **len**  
  The number of `ints` in the `pets` output parameter for which the application has allocated memory. `len` is the number of list elements that the system can return in `pets→ints`. If a value of 0 is used here, no data is returned in the `pets→ints` list, but the total number of elements will be returned in `tlen`.

- **st**  
  Starting position of inquiry. The elements in the list, beginning with the item number specified by `st`, are copied sequentially into `pets→ints` until `pets→ints` is full or all the elements have been copied.

- **store**  
  The memory buffer PHIGS is to use for storing the information returned for the `Pval_data` structure. This buffer must exist prior to calling this function (see `CREATE STORE (3P)`).

**C Output Parameters**

- **error_ind**  
  The error indicator. See the `Execution` section below for a description of its use. See the `ERRORS` section below for the possible values it can return.

- **def_value**  
  Default initial valuator value.

- **pets**  
  Available prompt/echo types. `Pint_list` is defined in `phigs.h` as follows:

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

  The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

  The application must allocate memory for `len` elements in the list of `ints`.

- **tlen**  
  A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for `len` if all elements in the list are to be
echo_area
A pointer to an object of type Plimit that contains the echo area of the device.
Plimit is defined in phigs.h as follows:

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

val_data
A pointer to a Pval_data pointer. PHIGS assigns this pointer to the location in the
Pstore structure that contains the device’s data record contents. Pval_data is
defined in phigs.h as follows:

typedef struct {
  Pfloat low; /* low range limit */
  Pfloat high; /* high range limit */
  union {
    struct Pval_pet_r1 {
      Pint unused;
    } pet_r1;
    struct Pval_pet_u1 {
      char *label;
      char *format;
      char *low_label;
      char *high_label;
    } pet_u1;
    struct Pval_pet_u4 {
      Phigs_dial_limits dial_limit; /* wraparound or stick at limits */
      Pint threshold; /* number of dial events to collapse
                      into 1 */
    } pet_u4;
  } pets;
} Pval_data;

FORTRAN Input
Parameters
Applications using the FORTRAN binding must supply a CHARACTER array to this function,
into which is placed the contents of the default input data record. The application
subsequently extracts the contents of the data record by using the UNPACK DATA RECORD
function. The allocated dimension of the character array is passed in the MLDR argument;

modified 2 April 1993
the dimension needed is returned in the LDR argument. The caller can determine the
required dimension by calling this function with MLDR set to zero, in which case PHIGS
returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero,
some values will be returned. These are LDR, the number of available prompt/echo
types, the prompt/echo type list entry requested, the echo area, the default valuator
range limits, and the default initial value. Error 2001 is returned if MLDR is too small, but
not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it
returns only one element of this list. The element to return is indicated by the calling
program via the argument N.

**TYPE**  The workstation type with which the device is associated.

**DEVNO**  The device number of the VALUATOR device. See the AVAILABLE DEVICES
section of INITIALIZE VALUATOR for a description of the available devices.

**N**  The index of the prompt/echo type list entry to return.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use.
See the ERRORS section below for the possible values it can return.

**DVAL**  The default initial value.

**OL**  The number of available prompt/echo types.

**PET**  The prompt/echo type corresponding to the Nth position in the list of
prompt/echo types.

**EAREA**  An array in which to place the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in
Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array. This must subsequently be passed to UNPACK DATA
RECORD to access its contents.

**Execution**  INQUIRE DEFAULT VALUATOR DEVICE DATA returns the default data of the specified valuator
device. This data is stored in the workstation description table associated with the
workstation type. See INITIALIZE VALUATOR for a description of the prompt/echo types,
echo area and data record contents and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN
Parameters sections above, then the error indicator indicates the error number of the
detected error, and no other output data is returned. If no error is detected, the error
indicator is set to zero, and the inquired information is available in the output parameters.
Since this is an inquiry function, ERROR HANDLING is not invoked when this function
detects an error.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state ((PHOP, *, *, *))</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, category of the specified workstation is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td>FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

INITIALIZE VALUATOR (3P)

INQUIRE DEFAULT VALUATOR DEVICE DATA 3 (3P)
NAME

INQUIRE DEFAULT VALUATOR DEVICE DATA 3—inquire the predefined valuator data

SYNOPSIS

C Syntax

void

pinq_def_val_data3 ( type, device, store, error_ind, def_value, pets, echo_vol, val_data )

Pint type; workstation type
Pint device; logical input device number
Pstore store; handle to Store object
Pint *error_ind; OUT error indicator
Pfloat *def_value; OUT default initial value
Pint_list **pets; OUT list of prompt and echo types
Plimit3 *echo_vol; OUT default echo volume
Pval_data3 **val_data; OUT default data record

FORTRAN Syntax

SUBROUTINE pqdvl3 ( WTYPE, DEVNO, N, MLDR, ERRIND, DVAL, OL, PET, EVOL, LDR, DATREC )

INTEGER WTYPE workstation type
INTEGER DEVNO logical input device number
INTEGER N list element requested
INTEGER MLDR dimension of data record array
INTEGER ERRIND OUT error indicator
REAL DVAL OUT default initial value
INTEGER OL OUT number of available prompt/echo types
INTEGER PET OUT Nth element of list of available prompt/echo types
REAL EVOL(6) OUT default echo volume in device coordinates
INTEGER LDR OUT number of array elements used in data record
CHARACTER∗80 DATREC(MLDR) OUT data record

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use INQUIRE DEFAULT VALUATOR DEVICE DATA 3 to determine the following information for a VALUATOR input device associated with a given workstation type:

- Number and list of available prompt/echo types.
- Default echo volume.
- Default input data record.
- Default initial valuator value.

Since the default prompt/echo type for all input devices is 1, the default input data record is for that prompt/echo type. There are no default input data records for prompt/echo types other than 1.

modified 2 April 1993 483
Applications using the C binding must create a buffer to be used by this function as memory space for storing portions of the device data. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The valuator device data record within the store buffer is accessed via the pointer returned in `val_data`. The prompt/echo type list is accessed in the store buffer via the pointer returned in `pets`.

**type**  
The workstation type with which the device is associated.

**device**  
The device number of the valuator device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available devices.

**store**  
The memory buffer PHIGS is to use for storing the information returned for the Pval_data structure and the prompt/echo type list. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

**Output Parameters**

**error_ind**  
The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

**def_value**  
Default initial valuator value.

**pets**  
Available prompt/echo types. Pint_list is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

**echo_vol**  
A pointer to an object of type Plimit that contains the echo volume of the device. Plimit is defined in phigs.h as follows:

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

**val_data**  
moved 2 April 1993
A pointer to a Pval_data3 pointer. PHIGS will assign this pointer to the location in the Pstore structure that contains the device's data record contents. Pval_data3 is defined in phigs.h as follows:

typedef struct {
    Pfloat low; /* low range limit */
    Pfloat high; /* high range limit */
    union {
        struct Pval_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pval_pet_u1 {
            char *label;
            char *format;
            char *low_label;
            char *high_label;
        } pet_u1;
        struct Pval_pet_u4 {
            Phigs_dial_limits dial_limit; /* wraparound or stick at limits */
            Pint threshold; /* number of dial events to collapse into 1 */
        } pet_u4;
    } pets;
} Pval_data3;

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into is placed the contents of the default input data record. The application subsequently extracts the contents of the data record by using the UNPACK DATA RECORD function. The allocated dimension of the character array is passed in the MLDR argument, the dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, the number of available prompt/echo types, the prompt/echo type list entry requested, the echo volume, the default valuator range limits, and the default initial value. Error 2001 is returned if MLDR is too small, but not if it is zero.

The FORTRAN function does not return the complete list of prompt/echo types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument N.

*TYPE*  The workstation type with which the device is associated.
**DEVNO**

The device number of the VALUATOR device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available devices.

**N**

The index of the prompt/echo type list entry to return.

**MLDR**

The dimension of the data record array, DATREC.

### FORTRAN Output Parameters

**ERRIND**

The *error indicator*. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**DVAL**

The default initial value.

**OL**

The number of available prompt/echo types.

**PET**

The prompt/echo type corresponding to the Nth position in the list of prompt/echo types.

**EVOl**

An array in which to place the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, and ZMAX in Device Coordinates.

**LDR**

The required dimension of the data record array, DATREC.

**DATREC**

The data record array. This must subsequently be passed to UNPACK DATA RECORD to access its contents.

### Execution

INQUIRE DEFAULT VALUATOR DEVICE DATA returns the default data of the specified valuator device. This data is stored in the workstation description table associated with the workstation type. See INITIALIZE VALUATOR for a description of the prompt/echo types, echo volume and data record contents, and how to set these values.

If this function detects an error, except in the cases mentioned in the C and FORTRAN Parameters sections above, then the *error indicator* indicates the error number of the detected error, and no other output data is returned. If no error is detected, the *error indicator* is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

### ERRORS

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

052 Ignoring function, workstation type not recognized by the implementation

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

061 Ignoring function, category of the specified workstation not INPUT or OUTIN

250 Ignoring function, the specified device is not available on the specified workstation

2200 C: Buffer overflow in input or inquiry function

2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN
SEE ALSO

array or string being passed as an output parameter is too small to contain the returned value.

INITIALIZE VALUATOR (3P)
INQUIRE DEFAULT VALUATOR DEVICE DATA 3 (3P)
**NAME**  
INQUIRE DISPLAY SPACE SIZE—obtain display space size of workstation type

**SYNOPSIS**

**C Syntax**

```c
void qinq_disp_space_size ( type, error_ind, size )
Pint type;  /* workstation type */
Pint *error_ind;  /* OUT error indicator */
Pdisp_space_size *size;  /* OUT display size */
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pqdsp ( WTYPE, ERRIND, DCUNIT, DX, DY, RX, RY )
INTEGER WTYPE  /* workstation type */
INTEGER ERRIND  /* OUT error indicator */
INTEGER DCUNIT  /* OUT device coordinate units (PMETRE, POTHU) */
REAL DX, DY  /* OUT maximum display surface size (DC) */
INTEGER RX, RY  /* OUT maximum display surface size (raster units) */
```

**Required PHIGS Operating States**

( PHOP, *, *, * )

**DESCRIPTION**

**Purpose**

INQUIRE DISPLAY SPACE SIZE returns the 2D display space size associated with the workstation type specified. The workstation type passed as an argument to this function can be either a generic type or a specific type, a specific type being one associated with an open workstation and obtained by a call to INQUIRE WORKSTATION CONNECTION AND TYPE. The information is retrieved from the workstation description table associated with the workstation type.

INQUIRE DISPLAY SPACE SIZE returns only the x and y components of the display space limits. Use INQUIRE DISPLAY SPACE SIZE 3 to retrieve all three dimensions.

**C Input Parameters**

- `type`  
The workstation type, either generic or specific.

**C Output Parameters**

- `error_ind`  
The error indicator. See the `Execution` section below for a description of its use. See the `ERRORS` section below for the possible values it can return.

- `size`  
A pointer to a `Pdisp_space_size` structure in which to store the requested information. `Pdisp_space_size` is defined in `phigs.h` as:

```c
typedef struct {
    Pdc_units dc_units;  /* device coordinate units */
    Pfloat_size size_dc;  /* device volume in coordinate units */
    Pint_size size_raster;  /* addressable units */
} Pdisp_space_size;
```

`dc_units` indicates the unit type, if any, of the device coordinate units. `Pdc_units` is defined in `phigs.h` as:

488 modified 2 April 1993
typedef enum {
  PDC_METRES,
  PDC_OTHER
} Pdc_units;

size(dc) contains the upper limits of the device coordinate units. The lower limits
are zero in all dimensions. See the Execution section below for a description of
these values. Pfloat_size is defined in phigs.h as:

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

size(raster) contains the number of addressable units corresponding to the device
coordinate units. See the Execution section below for a description of these values
for each supported workstation type. Pint_size is defined in phigs.h as:

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

**FORTRAN Input Parameters**

- **WTYPE**  The workstation type, either generic or specific.

**FORTRAN Output Parameters**

- **ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.
- **DCUNIT**  The unit type, if any, of the device coordinate units. Valid values as defined in phigs77.h are:
  - PMETRE  Meters
  - POTHU  Other
- **DX, DY**  The upper limits of the device coordinate units. The lower limits are zero in all dimensions. See the Execution section below for a description of these values.
- **RX, RY**  The number of addressable units corresponding to the device coordinate units. See the Execution section below for a description of the meaning of these values for each workstation type.

**Execution**

INQUIRE DISPLAY SPACE SIZE retrieves the device coordinate space limits and the number of
addressable units of a workstation type. The device coordinate space limits correspond
to the maximum device coordinate (DC) values of a workstation type. DC is the conceptual
coordinate system of the display device and is the coordinate system used by the
application to specify the workstation viewport and input device echo areas. The lower
limits of DC are always zero in all dimensions.

modified 2 April 1993
The DC limits and addressable units are specified in the workstation description table associated with a workstation type. They can be different in all dimensions.

The meaning of the DC limits and addressable units depends on both the workstation type and its status:

**X Tool and X Drawable types:**

The DC limits are set to the size of the workstation’s X window when the workstation is opened. When the window is resized, the DC limits change to the new size of the window. Subsequent calls to `INQUIRE DISPLAY SPACE SIZE` reflect this change by returning the limits in effect at the time of the function call. See the `PHIGS_DC_MODEL` attribute description in `WORKSTATION TYPE SET` for a description of how to change this behavior so that DC limits do not change with the window size.

If the type specified is a generic X Tool workstation type (that is, one not associated with an open workstation), then the display space size corresponds to the size of the window that would be used if a workstation were opened with that type.

If the type specified is a generic X Drawable workstation type, then error 51 will be returned in the error indicator, indicating that the information is unavailable for such a workstation type and a specific workstation type is required.

**CGM output types:**

The addressable limits are fixed at 32767 in the x and y dimensions.

If this function detects an error, then the `error indicator` indicates the error number of the detected error, and no other output data is returned. If no error is detected, then the `error indicator` is set to zero and the inquired information is available in the output parameters.

Since this is an inquiry function, `ERROR HANDLING` is not invoked when this function detects an error.

### ERRORS

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **052** Ignoring function, workstation type not recognized by the implementation
- **051** Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- **057** Ignoring function, specified workstation is of category MI
- **062** Ignoring function, this information is not available for this MO workstation type

### SEE ALSO

- `INQUIRE WORKSTATION CONNECTION AND TYPE (3P)`
- `WORKSTATION TYPE SET (3P)`
- `INQUIRE DISPLAY SPACE SIZE 3 (3P)`

modified 2 April 1993
NAME
INAQUIRE DISPLAY SPACE SIZE 3—obtain display space size of workstation type

SYNOPSIS
C Syntax

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

FORTRAN Syntax

    SUBROUTINE pqdsp3 ( WTYPE, ERRIND, DCUNIT, DX, DY, DZ, RX, RY, RZ )
       INTEGER WTYPE                  /* workstation type */
       INTEGER ERRIND                 /* OUT error indicator */
       INTEGER DCUNIT                 /* OUT device coordinate units (PMETRE, POTHU) */
       REAL DX, DY, DZ                /* OUT maximum display surface size (DC) */
       INTEGER RX, RY, RZ             /* OUT maximum display surface size (raster units) */

Description

Purpose
INQUIRE DISPLAY SPACE SIZE 3 returns the display space size associated with the
workstation type specified. The workstation type passed as an argument to this function
can be either a generic type or a specific type, a specific type being one associated with an
open workstation and obtained by a call to INQUIRE WORKSTATION CONNECTION AND TYPE.
The information is retrieved from the workstation description table associated with the
workstation type.

C Input Parameters

    type             The workstation type, either generic or specific.

C Output Parameters

    error_ind
       The error indicator. See the Execution section below for a description of its use.
       See the ERRORS section below for the possible values it can return.

    size
       A pointer to a Pdisp_space_size3 structure in which to store the requested
       information. Pdisp_space_size3 is defined in phigs.h as:

       typedef struct {
         Pdc_units       dc_units;        /* device coordinate units */
         Pfloat_size3    size_dc;        /* device volume in coordinate
                                         units */
         Pint_size3      size_raster;    /* addressable units */
       } Pdisp_space_size3;

       dc_units indicates the unit type, if any, of the device coordinate units. Pdc_units
       is defined in phigs.h as:

       typedef enum {
         /* enumeration values */
       } Pdc_units;

modified 2 April 1993

R}
INQUIRE DISPLAY SPACE SIZE 3(3P) SunPHIGS Release 3.0

PDC_METRES,
PDC_OTHER
}
Pdc_units;

size_dc contains the upper limits of the device coordinate units. The lower limits are zero in all dimensions. See the Execution section below for a description of these values. Pfloat_size3 is defined in phigs.h as:

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

size_raster contains the number of addressable units corresponding to the device coordinate units. See the Execution section below for a description of these values for each supported workstation type. Pint_size3 is defined in phigs.h as:

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

FORTRAN Input Parameters

WTYPE The workstation type, either generic or specific.

FORTRAN Output Parameters

ERRIND The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

DCUNIT The unit type, if any, of the device coordinate units. Valid values as defined in phigs77.h are:

    PMETRE  Meters
    POTHU   Other

DX, DY, DZ The upper limits of the device coordinate units. The lower limits are zero in all dimensions. See the Execution section below for a description of these values.

RX, RY, RZ The number of addressable units corresponding to the device coordinate units. See the Execution section below for a description of these values for each workstation type.

Execution INQUIRE DISPLAY SPACE SIZE 3 retrieves the device coordinate space limits and the number of addressable units of a workstation type. The device coordinate space limits correspond to the maximum device coordinate (DC) values of a workstation type. DC is the conceptual coordinate system of the display device and is the coordinate system used by the application to specify the workstation viewport and input device echo areas. The

492 modified 2 April 1993
lower limits of DC are always zero in all three dimensions.
The DC limits and addressable units are specified in the workstation description table associated with a workstation type. They can be different in all three dimensions.
The description of the addressable units depends both on the workstation type and its status:

X Tool and X Drawable types:
The DC limits are set to the size of the workstation’s X window when the workstation is opened. When the window is resized, the DC limits change to the new size of the window. Subsequent calls to INQUIRE DISPLAY SPACE SIZE 3 reflect this change by returning the limits in effect at the time of the function call. See the PHIGS_DC_MODEL attribute description in WORKSTATION TYPE SET for a description of how to change this behavior so that DC limits do not change with the window size.

If the type specified is a generic X Tool workstation type (that is, one not associated with an open workstation), then the display space size corresponds to the size of the window that would be used if a workstation were opened with that type.

If the type specified is a generic X Drawable workstation type, then error 51 will be returned in the error indicator, indicating that the information is unavailable for that workstation type, and a specific workstation type is required.

The z dimension of the display space size is always 1.

CGM output types:
The addressable limits are fixed at 32767 in the x and y dimensions and 1 in the z dimension.

If this function detects an error, then the error indicator indicates the error number of the detected error, and no other output data is returned. If no error is detected, then the error indicator is set to zero, and the inquired information is available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
057 Ignoring function, specified workstation is of category MI
062 Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

INQUIRE WORKSTATION CONNECTION AND TYPE (3P)
WORKSTATION TYPE SET (3P)
INQUIRE DISPLAY SPACE SIZE (3P)

modified 2 April 1993
NAME

INQUIRE DISPLAY UPDATE STATE – obtain current display update state

SYNOPSIS

C Syntax

void

pinq_disp_upd_st ( ws, error_ind, def_mode, mod_mode, display_empty, state )

Pint ws;  /* workstation identifier */
Pint *error_ind;  /* OUT error indicator */
Pdefer_mode *def_mode;  /* OUT deferral mode */
Pmod_mode *mod_mode;  /* OUT modification mode */
Pdisp_surf_empty *disp_empty;  /* OUT display surface empty */
Pvisual_st *state; /* OUT state of visual representation */

FORTRAN Syntax

SUBROUTINE pqdus ( WKID, ERRIND, DEFMOD, MODMOD, DEMPTY, STOFVR )

INTEGER WKID  /* workstation identifier */
INTEGER ERRIND  /* OUT error indicator */
INTEGER DEFMOD  /* OUT deferral mode */
INTEGER MODMOD  /* OUT modification mode */
INTEGER DEMPTY  /* OUT display surface empty */
INTEGER STOFVR  /* OUT state of visual representation */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE DISPLAY UPDATE STATE to determine the current display update state from a specified workstation’s state list.

The deferral mode and modification mode were set by the application. The display surface empty and state of visual representation state list values are maintained by PHIGS.

C Input Parameters

ws  The workstation identifier of the workstation whose state list is queried.

C Output Parameters

error_ind  A pointer to the location to store the error number of any error that this function detects.

def_mode  A pointer to a location in which the system returns the current deferral mode. Possible values for Pdefer_mode are defined in phigs.h as follows:

typedef enum {

  PDEFER_ASAP  /* make the display visually correct As Soon As Possible */

  PDEFER_BNIG  /* make the display visually correct Before the Next Interaction Globally */

}
PDEFER_BNIL /* make the display visually correct
   Before the Next Interaction Locally */
PDEFER_ASTI /* make the display visually correct
   At Some Time */
PDEFER_WAIT /* make the display visually correct
   When the Application Requests It */

} Pdefer_mode;

mod_mode
A pointer to a location in which the system returns
the current modification mode.
Possible values for Pmod_mode are defined in phigs.h
as follows:

typedef enum {
    PMODE_NIVE /* no immediate visual effects */
    PMODE_UWOR /* update without regeneration */
    PMODE_UQUM /* use quick update methods */
} Pmod_mode;

display_empty
A pointer to a location in which the system returns the
current display surface empty status.
Possible values for Pdisp_surf_empty are:

typedef enum {
    PSURF_EMPTY /* empty */
    PSURF_NOT_EMPTY /* not empty */
} Pdisp_surf_empty;

state
A pointer to a location in which the system returns the current state of visual
representation. Possible values for Pvisual_st are:

typedef enum {
    PVISUAL_ST_CORRECT /* correct */
    PVISUAL_ST_DEFER /* deferred */
    PVISUAL_ST_SIMULATED /* simulated */
} Pvisual_st;

FORTRAN Input
Parameters

WKID The workstation identifier of the workstation whose state list is queried.

FORTRAN Output
Parameters

ERRIND The error number of any error detected by this function.

DEFMOD The current deferral mode. Possible values are:

0 PASAP Make the display visually correct — As Soon As Possible

modified 2 April 1993
<table>
<thead>
<tr>
<th></th>
<th>MODMOD</th>
<th>DEMPTY</th>
<th>STOFVR</th>
<th>ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PNIVE</td>
<td>PNEMPTY</td>
<td>PVROK</td>
<td>003</td>
</tr>
<tr>
<td>1</td>
<td>PUWOR</td>
<td>PEMPTY</td>
<td>PVROK</td>
<td>054</td>
</tr>
<tr>
<td>2</td>
<td>PUQUM</td>
<td></td>
<td>PVDRDF</td>
<td>059</td>
</tr>
</tbody>
</table>

**MODMOD** The current modification mode. Possible values are:

- **PNIVE** No Immediate Visual Effects
- **PUWOR** Update Without Regeneration
- **PUQUM** Use Quick Update Methods

**DEMPHY** The current display surface empty status. Possible values are:

- **PNEMPTY** Not Empty
- **PEMPTY** Empty

**STOFVR** The current state of visual representation. Possible values are:

- **PVROK** Correct
- **PVDRDF** Deferred
- **PVRSIM** Simulated

**ERRORS**

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

- SET DISPLAY UPDATE STATE (3P)
- INQUIRE DEFAULT DISPLAY UPDATE STATE (3P)
NAME

INQUIRE DYNAMICS OF STRUCTURES – obtain dynamics of structures for workstation type

SYNOPSIS

C Syntax

void

pinq_dyns_structs ( type, error_ind, dynamics )

Pint type; /* workstation type */
Pint *error_ind; /* OUT error indicator */
Pdyns_structs *dynamics; /* OUT structure dynamics */

FORTRAN Syntax

SUBROUTINE pqdstr ( WTYPE, ERRIND, STRCON, POST, UNPOST, DELETE, REFMOD )

INTEGER WTYPE /* workstation type */
INTEGER ERRIND /* OUT error indicator */
INTEGER STRCON /* OUT structure content */
INTEGER POST /* OUT posting a structure */
INTEGER UNPOST /* OUT unposting a structure */
INTEGER DELETE /* OUT deleting a structure */
INTEGER REFMOD /* OUT reference modification */

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose

Use INQUIRE DYNAMICS OF STRUCTURES to determine the dynamics of changes to structure contents and posting for a specified workstation type.

The possible dynamic modification accepted values are the same in C or FORTRAN:

- **PIRG** Modification leads to an Implicit Regeneration.
- **PIMM** Modification is performed Immediately, without an implicit regeneration.
- **PCBS** Modification Can Be Simulated before an implicit regeneration is permitted.

C Input Parameters

type The workstation type whose capabilities are inquired.

C Output Parameters

- **error_ind**
  
  A pointer to the location to store the error number of any error that this function detects.

- **dynamics**
  
  A pointer to a structure in which the system returns the dynamic modification accepted values for various structure manipulations. Pdyns_structs is defined in phigs.h as follows:

  ```
  typedef struct {
      /* structure details */
  } Pdyns_structs;
  ```

modified 2 April 1993
Pdyn_mod content; /* structure content */
Pdyn_mod post; /* post structure */
Pdyn_mod unpost; /* unpost structure */
Pdyn_mod del; /* delete structure */
Pdyn_mod ref; /* structure reference */
}

Pdyns_structs;
The Pdyn_mod enumeration holds the dynamic modification accepted value, one of the values PDYN_IRG, PDYN_IMM, or PDYN_CBC.

**FORTRAN Input Parameters**

- **WTYPE** The workstation type of the workstation whose capabilities are inquired.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **STRCON** The dynamic modification accepted value for structure content changes or structure editing.
- **POST** The dynamic modification accepted value for structure posting.
- **UNPOST** The dynamic modification accepted value for structure unposting.
- **DELETE** The dynamic modification accepted value for structure deletion.
- **REFMOD** The dynamic modification accepted value for structure reference modifications. This value applies to calls to CHANGE STRUCTURE IDENTIFIER, CHANGE STRUCTURE REFERENCES, and CHANGE STRUCTURE IDENTIFIER AND REFERENCES.

**ERRORS**

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- 052 Ignoring function, workstation type not recognized by the implementation
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- 062 Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

PHIGS WORKSTATION DESCRIPTION TABLE (7P)

498 modified 2 April 1993
NAME
INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES – obtain dynamics of workstation attributes for specified workstation type

SYNOPSIS

C Syntax
void
pinq_dyns_ws_attrs ( type, error_ind, attr )
Pint ws_type; /* workstation type */
Pint *err_ind; /* OUT error indicator */
Pdyns_ws_attr *attr; /* OUT attributes dynamics */

FORTRAN Syntax
SUBROUTINE pqdswa ( WTYPE, ERRIND, PLBUN, PMBUN, TXBUN, INBUN,
                     EDBUN, PAREP, COLREP, VWREP, WKTR, HIFLTR, INFLTR, HLHSR )
INTEGER WTYPE /* workstation type */
INTEGER ERRIND /* error indicator */
INTEGER PLBUN /* OUT polyline representation changeable */
INTEGER PMBUN /* OUT polymarker representation changeable */
INTEGER TXBUN /* OUT text representation changeable */
INTEGER INBUN /* OUT interior representation changeable */
INTEGER EDBUN /* OUT edge representation changeable */
INTEGER PAREP /* OUT pattern representation changeable */
INTEGER COLREP /* OUT colour representation changeable */
INTEGER VWREP /* OUT view representation changeable */
INTEGER WKTR /* OUT workstation transformation changeable */
INTEGER HIFLTR /* OUT highlighting filter changeable */
INTEGER INFLTR /* OUT invisibility filter changeable */
INTEGER HLHSR /* OUT HLHSR mode changeable */

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES to determine dynamics of workstation attributes for a specified workstation type.

The possible dynamic modification accepted values are the same in C or FORTRAN:

PIRG Modification leads to an Implicit Regeneration.

PIMM Modification is performed Immediately, without an implicit regeneration.

PCBS Modification Can Be Simulated before an implicit regeneration is permitted.

C Input Parameters

type The workstation type whose capabilities are inquired.
### C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error that this function detects.

- **attr**
  A pointer to a structure in which the system returns the dynamic modification accepted values for various workstation attributes. `Pdyns_ws_attrs` is defined in `phigs.h` as follows:

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

The `Pdyn_mod` enumeration holds the dynamic modification accepted value, one of the values PDYN_IRG, PDYN_IMM, or PDYN_CBS.

### FORTRAN Input Parameters

- **WTYPE**
  The workstation type of the workstation whose capabilities are queried for.

### FORTRAN Output Parameters

- **ERRIND**
  The error number of any error detected by this function.

- **PLBUN**
  `PLBUN` is the dynamic modification accepted value for polyline bundle changes.

- **PMBUN**
  `PMBUN` is the dynamic modification accepted value for polymarker bundle changes.

- **TXBUN**
  `TXBUN` is the dynamic modification accepted value for text bundle changes.

- **INBUN**
  `INBUN` is the dynamic modification accepted value for interior bundle changes.

- **EDBUN**
  `EDBUN` is the dynamic modification accepted value for edge bundle changes.

- **PABUN**
  `PABUN` is the dynamic modification accepted value for pattern bundle changes.

- **COLBUN**
  `COLBUN` is the dynamic modification accepted value for colour bundle changes.

- **VWREP**
  `VWREP` is the dynamic modification accepted value for view representation changes.

- **WKTR**
  `WKTR` is the dynamic modification accepted value for workstation transformation changes.
**HIFLTR**  
_HIFLTR_ is the dynamic modification accepted value for highlighting filter changes.

**INFLTR**  
_INFLTR_ is the dynamic modification accepted value for invisibility filter changes.

**HLHSR**  
_HLHSR_ is the dynamic modification accepted value for HLHSR mode changes.

**ERRORS**

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

051  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052  Ignoring function, workstation type not recognized by the implementation

059  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062  Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**  
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NAME
INQUIRE EDGE FACILITIES – obtain list of workstation edge facilities

SYNOPSIS
C Syntax

```c
void
pinq_edge_facs ( type, length, start, error_ind, facilities, total_length )
```

- `type`: workstation type
- `length`: length of application list
- `start`: starting position
- `error_ind`: OUT error indicator
- `facilities`: OUT edge facilities
- `total_length`: OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqedf ( WTYPE, N, ERRIND, NEDT, EDT, NEDW, NOMEDW,
REDWMN, REDWMX, NPEDI )
```

- `WTYPE`: workstation type
- `N`: list element requested
- `ERRIND`: OUT error indicator
- `NEDT`: OUT number of available edgetypes
- `EDT`: OUT Nth element of list of available edgetypes
- `NEDW`: OUT number of available edgewidths
- `NOMEDW`: OUT nominal edgewidth
- `REDWMN`: OUT range of edgewidths
- `REDWMX`: OUT range of edgewidths
- `NPEDI`: OUT number of predefined edge indices

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose
Use INQUIRE EDGE FACILITIES to obtain a list of the edge facilities available for the specified type of workstation.

C Input Parameters

- `type`: Workstation type.
- `length`: The number of elements for which the application has allocated memory in the output parameter list of integers. If a value of zero is used, no data will be returned in this list, but the total number of elements in the workstation list of available edge types will be returned in `total_length`.
- `start`: Starting position in the list of available edge types in the workstation description table at which to begin the inquiry. The elements of the workstation list of available edge types, beginning with the item number specified by `start`, are copied sequentially into the list of integers until it is full or all the items on the workstation list have been copied.

502 modified 2 April 1993
C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

facilities
A pointer to a Pedge_facs structure in which the system returns a list of the edge facilities available for the specified workstation type. Pedgefac is defined in phigs.h as follows:

typedef struct {
    Pint_list types; /* list of edge types */
    Pint num_widths; /* number of available edge widths */
    Pfloat nom_width; /* nominal edge width */
    Pfloat min_width; /* minimum edge width */
    Pfloat max_width; /* maximum edge width */
    Pint num_pred_inds; /* number of predefined bundles */
} Pedge_facs;

And Pint_list is defined as follows:

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

The application must allocate memory for length elements in the list of integers.

total_length
A pointer to an integer in which the system returns the total number of items in the list of available edge types. This is the value required for length if all the items on the workstation list are to be returned.

FORTRAN Input Parameters

WTYPE Workstation type.

N Element of the specified workstation type list of available edge types to return. Only one edge type may be inquired per subroutine call. If a value of zero is used here, no edge type data will be returned, but the total number of elements in the workstation list of available edge types will be returned in NEDT.

FORTRAN Output Parameters

ERRIND The error number of any error detected by this function.

NEDT The total number of edge types available for the given workstation type.

EDT The Nth edge type from the list of available edge types for the given workstation type.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEDW</td>
<td>The number of available edge widths for the given workstation type.</td>
</tr>
<tr>
<td>NOMEDW</td>
<td>The nominal edge width used by the given workstation type as a base for</td>
</tr>
<tr>
<td></td>
<td>edge width scaling.</td>
</tr>
<tr>
<td>REDWMN</td>
<td>The minimum available edge width for the given workstation type.</td>
</tr>
<tr>
<td>REDWMX</td>
<td>The maximum available edge width for the given workstation type.</td>
</tr>
<tr>
<td>NPEDI</td>
<td>The number of predefined edge bundle indices for the given workstation</td>
</tr>
<tr>
<td></td>
<td>type.</td>
</tr>
</tbody>
</table>

**ERRORS**

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **051** Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- **052** Ignoring function, workstation type not recognized by the implementation
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- **062** Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

- PHIGS WORKSTATION DESCRIPTION TABLE (7P)
- INQUIRE PREDEFINED EDGE REPRESENTATION (3P)
- INQUIRE PREDEFINED EDGE REPRESENTATION PLUS (3PP)
NAME
INQUIRE EDGE REPRESENTATION – obtain edge representation on specified workstation

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

FORTRAN Syntax
SUBROUTINE pqedr ( WKID, EDI, TYPE, ERRIND, EDFLAG, EDTYPE, EWIDTH, COLI )
INTEGER  WKID  /* workstation identifier */
INTEGER  EDI  /* edge index */
INTEGER  TYPE  /* type of returned values (PSET, PREALI) */
INTEGER  ERRIND  /* OUT error indicator */
INTEGER  EDFLAG  /* OUT edge flag (POFF, PON) */
INTEGER  EDTYPE  /* OUT edge type */
REAL  EWIDTH  /* OUT edgewidth scale factor */
INTEGER  COLI  /* OUT edge colour index */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE EDGE REPRESENTATION to determine the current attribute values for a
specified entry in a specified workstation’s table of defined edge representations.
See the description of the subroutine SET EDGE REPRESENTATION for information about the
meaning of these attribute values.

C Input Parameters
ws  Workstation identifier.
index  Entry to be returned from the workstation’s table of edge representations; if this
entry is not present in the table and the type of returned value parameter is
REALIZED, the representation for edge index 1 is returned.
type  An enumerated value specifying whether the inquired values are to be returned
as the values originally specified by the application (SET), or as the values actually
being used by the workstation if any of the application-specified values had to be
mapped to ones available on the workstation (REALIZED). Valid values are
defined in phigs.h as:
typedef enum {
   PINQ_SET  /* Return application-specified value */
   }
INQUIRE EDGE REPRESENTATION (3P) SunPHIGS Release 3.0

**C Output Parameters**

- **error_ind**: A pointer to the location to store the error number of any error that this function detects.
- **rep**: A pointer to a Pedge_bundle structure in which the system returns the edge representation at index in the workstation’s table of edge representations. Pedge_bundle is defined in phigs.h as follows:

```c
typedef struct {
    Pedge_flag flag; /* edge_flag */
    Pint type; /* edgetype */
    Pfloat width; /* edgewidth_scale_factor */
    Pint colr_ind; /* edge_colour_index */
} Pedge_bundle;
```

Valid values for the Pedge_flag enumerated type are defined in phigs.h as:

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

**FORTRAN Input Parameters**

- **WKID**: Workstation identifier.
- **EDI**: Entry to be returned from the workstation’s table of edge representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for edge index 1 is returned.
- **TYPE**: An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs77.h as:

```c
PSET   Return application-specified value
PREALI Return value available on the workstation
```

**FORTRAN Output Parameters**

- **ERRIND**: The error number of any error that this function detects.
- **EDFLAG**: The edge flag value at index EDI in the workstation’s table of edge representations.
- **EDTYPE**: The edge type at index EDI in the workstation’s table of edge representations.
- **EWIDTH**: The edge width scale factor at index EDI in the workstation’s table of edge representations.
- **COLI**: The edge colour index at index EDI in the workstation’s table of edge representations.
ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
059  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO.
100  Ignoring function, the bundle index value is less than one
101  Ignoring function, the specified representation has not been defined
134  Ignoring function, the requested entry contains a general colour specification with colour type other than INDIRECT.

SEE ALSO

INQUIRE EDGE FACILITIES (3P)
SET EDGE REPRESENTATION (3P)
INQUIRE PREDEFINED EDGE REPRESENTATION (3P)
INQUIRE EDGE REPRESENTATION PLUS (3PP)
INQUIRE EDIT MODE – obtain current edit mode for structure

**SYNOPSIS**

C Syntax

```c
void pqnq_edit_mode ( error_ind, edit_mode )
Pint *error_ind; OUT error indicator
Pedit_mode *edit_mode; OUT edit mode
```

FORTRAN Syntax

```fortran
SUBROUTINE pqedm ( ERRIND, EDITMO )
INTEGER ERRIND OUT error indicator
INTEGER EDITMO OUT edit mode (PINSRT, PREPLC)
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE EDIT MODE to determine the current edit mode for structures. The edit mode may be INSERT or REPLACE.

See the description of the subroutine SET EDIT MODE for information on the meaning of these modes.

**C Output Parameters**

- `error_ind`
  
  A pointer to the location to store the error number of any error that this function detects.

- `edit_mode`
  
  A pointer to a Pedit_mode enumerated type in which the system returns the current structure edit mode. Values for Pedit_mode are defined in phigs.h as follows:

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

**FORTRAN Output Parameters**

- `ERRIND`
  
  The error number of any error that this function detects.

- `EDITMO`
  
  The current structure edit mode, defined in phigs77.h as follows:

  ```c
  PINSRT Insert
  PREPLC Replace
  ```

**ERRORS**

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

**SEE ALSO**

SET EDIT MODE (3P)

modified 2 April 1993
**NAME**  
INQUIRE ELEMENT CONTENT – inquire the contents of the specified element

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqeco ( STRID, ELENUM, IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )
INTEGER STRID  // structure identifier
INTEGER ELENUM // element number
INTEGER IIL    // dimension of integer array
INTEGER IRL    // dimension of real array
INTEGER ISL    // dimension of character array
INTEGER ERRIND // OUT error indicator
INTEGER IL     // OUT number of integer entries
INTEGER IA(IIL) // OUT array containing integer entries
INTEGER RL     // OUT number of real entries
REAL RA(IRL)   // OUT array containing real entries
INTEGER SL     // OUT number of character string entries
INTEGER LSTR(ISL) // OUT length of each character string entry
CHARACTER*80 STR(ISL) // OUT character string entries
```

**FORTRAN Subset Syntax**

```fortran
SUBROUTINE pqeco ( STRID, ELENUM, IIL, IRL, ISL, ERRIND, IL, IA, RL, RA, SL, LSTR, STR )
INTEGER STRID  // structure identifier
INTEGER ELENUM // element number
INTEGER IIL    // dimension of integer array
INTEGER IRL    // dimension of real array
INTEGER ISL    // dimension of character array
INTEGER ERRIND // OUT error indicator
INTEGER IL     // OUT number of integer entries
INTEGER IA(IIL) // OUT array containing integer entries
INTEGER RL     // OUT number of real entries
REAL RA(IRL)   // OUT array containing real entries
INTEGER SL     // OUT number of character string entries
INTEGER LSTR(ISL) // OUT length of each character string entry
CHARACTER*80 STR(ISL) // OUT character string entries
```

modified 2 April 1993
INQUIRE ELEMENT CONTENT (3P) SunPHIGS Release 3.0

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION Purpose

INQUIRE ELEMENT CONTENT determines the contents of the specified element in the specified structure.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by data.

struct_id
Identifier of the structure containing the element for which to return the contents.

element
Position in the specified structure of the element for which to return the contents.

store
The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

data
A pointer to a Pelem_data union, in which the contents of the element will be returned, as appropriate for the element type. Pelem_data is defined in phigs.h as follows:

typedef union {
Pint int_data; /* integer valued data */
Pfloat float_data; /* float valued data */
Ppoint_list3 point_list3; /* list of 3D points */
Ppoint_list point_list; /* list of 2D points */
Ppoint_list_list3 point_list_list3; /* list of 3D point lists */
Ppoint_list_list point_list_list; /* list of 2D point lists */
} Pelem_data;

struct Pelem_text3 {
Ppoint3 pos; /* text pt */
Pvec3 dir[2]; /* direction vectors */
char *char_string; /* text string */
} text3;

struct Pelem_text {
Ppoint pos; /* text pt */
char *char_string; /* text string */
} text;

struct Pelem_anno_text_rel3 {

modified 2 April 1993
Ppoint3 ref_point; /* reference pt */
Pvec3 offset; /* anno. pt/offset */
char *char_string; /* text string */
}
) anno_text_rel3;
struct Pelem_anno_text_rel {
Ppoint ref_point; /* reference pt */
Pvec offset; /* anno. pt/offset */
char *char_string; /* text string */
}
) anno_text_rel;
struct Pelem_cell_array3 {
Pparal paral; /* parallelogram */
Ppat_rep colr_array; /* colour array */
}
cell_array3;
struct Pelem_cell_array {
Prec rect; /* rectangle */
Ppat_rep colr_array; /* colour array */
}
cell_array;
struct Pelem_gdp3 {
Pint id; /* GDP3 id */
Ppoint_list3 point_list; /* pts */
Pgdp_data3 data; /* data record */
}
gdp3;
struct Pelem_gdp {
Pint id; /* GDP id */
Ppoint_list point_list; /* pts */
Pgdp_data data; /* data record */
}
gdp;
Ptext_prec text_prec; /* text precision */
Pvec char_up_vec; /* char up vector */
Ptext_path text_path; /* text path */
Ptext_align text_align; /* text alignment */
Pint_style int_style; /* interior style */
Pedge_flag edge_flag; /* edge flag */
Ppoint pat_ref_point; /* pat ref pt */
Pfloat_size pat_size; /* pattern size */
struct Pelem_pat_ref_point_vecs {
Ppoint3 ref_point; /* pattern ref. pt */
Pvec3 ref_vec[2]; /* vectors */
}
pat_ref_point_vecs;
Pint_list names; /* name sets */
struct Pelem_asf {
Paspect id; /* attribute id */
Pasf source; /* asf */
}
asf;

modified 2 April 1993
struct Pelem_local_tran3 {
    Pcompose_type compose_type;        /* composition type */
    Pmatrix3 matrix;                   /* transformation matrix */
}

struct Pelem_local_tran {
    Pcompose_type compose_type;        /* composition type */
    Pmatrix matrix;                    /* transformation matrix */
}

struct Pelem_model_clip3 {
    Pint op;                           /* operator */
    Phalf_space_list3 half_spaces;     /* half-space list */
}

struct Pelem_model_clip {
    Pint op;                           /* operator */
    Phalf_space_list half_spaces;      /* half-space list */
}

struct Pelem_nurb_curve {
    Pint order;
    Pfloat_list knots;
    Prational rationality;
    Ppoint_list34 cpts;
    Pfloat min;
    Pfloat max;
}

struct Pelem_curv_approx {
    Pint type;
    Pfloat value;
}

struct Pelem_nurb_surf {
    Pint u_order;
    Pint v_order;
    Prational rationality;
}
Pfloat_list uknots;
Pfloat_list vknots;
Ppoint_grid34 grid;
Pint num_trim_loops;
Ptrimcurve_list *trim_loops;
} nurb_surf;
struct Pelem_surf_approx {
Pint type;
Pfloat u_val;
Pfloat v_val;
} surf_approx;
struct Pelem_lsd3 {
Pint vflag;
Pint colr_model;
Pint npl;
Pline_vdata_list3 *vdata;
} plsd3;
struct Pelem_fasd3 {
Pint fflag;
Pint eflag;
Pint vflag;
Pint colr_model;
Pfacet_data3 fdata;
Pint nfa;
Pedge_data_list *edata;
Pfacet_vdata_list3 *vdata;
} fasd3;
struct Pelem_tsd3 {
Pint fflag;
Pint vflag;
Pint colr_model;
Pint nv; /* number of vertices */
Pfacet_data_arr3 fdata;
Pfacet_vdata_arr3 vdata;
} tsd3;
struct Pelem_qmd3 {
Pint fflag;
Pint vflag;
Pint colr_model;
Pint_size dim;
Pfacet_data_arr3 fdata;
Pfacet_vdata_arr3 vdata;
} qmd3;

modified 2 April 1993
INQUIRE ELEMENT CONTENT is the second function called in a two-function, or step, process. First, INQUIRE ELEMENT TYPE AND SIZE must be called to determine the type of element to return for the INQUIRE ELEMENT CONTENT request. Depending on the element type, the corresponding structure in the Pelem_data union is returned, as described below.

**Note:** this process works similarly for INQUIRE CURRENT ELEMENT CONTENT and INQUIRE CURRENT ELEMENT TYPE AND SIZE.

**ADD NAMES TO SET (PELEM_ADD_NAMES_SET)**

A Pint_list structure provided in Pelem_data as:

```
typedef struct {
    Pint_list names /* name sets */
} Pelem_data;
```

```c
INQUIRE ELEMENT CONTENT (3P)
```
Pint    num_ints;  /* number of Pints in list */  
Pint    *ints;    /* list of integers */

} Pint_list;

ALL (PELEM_ALL)
No data involved

ANNOTATION STYLE (PELEM_ANNO_STYLE)
An integer provided in Pelem_data as:

Pint    int_data;    /* integer valued data */

ANNOTATION TEXT ALIGNMENT (PELEM_ANNO_ALIGN)
A Ptext_align structure provided in Pelem_data as:

Ptext_align    text_align;    /* text alignment */

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

ANNOTATION TEXT CHARACTER HEIGHT (PELEM_ANNO_CHAR_HT)
A Pfloat floating point data provided in Pelem_data as:

Pfloat    float_data;    /* float valued data */

ANNOTATION TEXT CHARACTER UP VECTOR
(PELEM_ANNO_CHAR_UP_VEC)
A Pvec structure provided in Pelem_data as:

Pvec    char_up_vec;    /* char up vector */

where
typedef struct {
Pfloat    delta_x;        /* x magnitude */
Pfloat    delta_y;        /* y magnitude */
} Pvec;

ANNOTATION TEXT PATH (PELEM_ANNO_PATH)
A Ptext_path data structure provided in Pelem_data as:

Ptext_path    text_path;    /* text path */

where
typedef enum {
PPATH_RIGHT,
ANNOTATION TEXT RELATIVE (PELEM_ANNO_TEXT_REL)
An anno_text_rel structure defined in Pelem_data as:
struct {
    Ppoint ref_point;  /* reference pt */
    Pvec offset;       /* annotation offset */
    char *char_string; /* annotation text string */
} anno_text_rel;

ANNOTATION TEXT RELATIVE 3 (PELEM_ANNO_TEXT_REL3)
An anno_text_rel3 structure defined in Pelem_data as:
struct {
    Ppoint3 ref_point; /* reference pt */
    Pvec3 offset;    /* annotation offset */
    char *char_string; /* annotation text string */
} anno_text_rel3;

APPLICATION DATA (PELEM_APPL_DATA)
A Pdata structure provided in Pelem_data as:
Pdata appl_data; /* application data */
where
typedef struct {
    size_t size;    /* size of data */
    void *data;    /* pointer to data */
} Pdata;

BACK REFLECTANCE PROPERTIES (PELEM_BACK_REFL_PROPS)
A Prefl_props structure provided in Pelem_data as:
Prefl_props properties; /* area properties */
where
typedef struct {
    Pfloat ambient_coef; /* ambient reflectance coefficient */
    Pfloat diffuse_coef; /* diffuse reflectance coefficient */
    Pfloat specular_coef; /* specular reflectance coefficient */
    Pgcolr specular_colr; /* specular colour */
} Prefl_props;
BACK INTERIOR COLOUR (PELEM_BACK_INT_COLR)
A Pgcolr structure provided in Pelem_data as:

```c
typedef struct {
    Pint type; /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint ind; /* index in workstation colour bundle table */
        struct {
            Pfloat x; /* red, hue, and so on */
            Pfloat y; /* green, saturation, lightness, and so on */
            Pfloat z; /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

BACK INTERIOR REFLECTANCE EQUATION (PELEM_BACK_INT_REFL_EQN)
An integer provided in Pelem_data as:

```c
Pint int_data; /* integer valued data */
```

The predefined reflectance equation values are:

- 1 **PREFL_NONE** No Reflectance Calculation Performed
- 2 **PREFL_AMBIENT** Use Ambient Term
- 3 **PREFL_AMB_DIFF** Use Ambient and Diffuse Terms
- 4 **PREFL_AMB_DIFF_SPEC** Use Ambient, Diffuse, and Specular Terms

BACK INTERIOR SHADING METHOD (PELEM_BACK_INT_SHAD_METH)
An integer provided in Pelem_data as:

```c
Pint int_data;
```

The predefined shading method values are:

- 1 **PSD_NONE** No Shading
- 2 **PSD_COLOUR** Colour Interpolation Shading
- 3 **PSD_DOT_PRODUCT** Dot Product Interpolation Shading
- 4 **PSD_NORMAL** Normal Interpolation Shading

modified 2 April 1993
BACK INTERIOR STYLE (PELEM_BACK_INT_STYLE)
An Pint_style structure provided in Pelem_data as:

```c
Pint_style int_style; /* back interior style */
```

where

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

BACK INTERIOR STYLE INDEX (PELEM_BACK_INT_STYLE_IND)
An integer provided in Pelem_data as:

```c
Pint int_data; /* integer valued data */
```

CELL ARRAY (PELEM_CELL_ARRAY)
A cell_array structure defined in Pelem_data as:

```c
struct {
    Prect          rect;    /* rectangle */
    Ppat_rep       colr_array; /* colour array */
} cell_array;
```

CELL ARRAY 3 (PELEM_CELL_ARRAY3)
A cell_array3 structure defined in Pelem_data as:

```c
struct {
    Pparal         paral;  /* parallelogram */
    Ppat_rep       colr_array; /* colour array */
} cell_array3;
```

CELL ARRAY 3 PLUS (PELEM_CELL_ARRAY3_PLUS)
An cell_array_plus structure provided in Pelem_data as:

```c
struct {
    Pparal         paral;  /* parallelogram */
    Ppat_rep_plus  colr_array; /* colour array */
} cell_array_plus;
```

CHARACTER EXPANSION FACTOR (PELEM_CHAR_EXPAN)
A floating point provided in Pelem_data as:

```c
Pfloat float_data; /* float valued data */
```
CHARACTER HEIGHT (PELEM_CHAR_HT)
A floating point provided in Pelem_data as:

Pfloat float_data; /* float valued data */

CHARACTER SPACING (PELEM_CHAR_SPACE)
A float_data floating point data provided in Pelem_data as:

Pfloat float_data; /* float valued data */

CHARACTER UP VECTOR (PELEM_CHAR_UP_VEC)
A Pvec structure provided in Pelem_data as:

Pvec char_up_vec; /* char up vector */

where typedef struct {
    Pfloat delta_x; /* x magnitude */
    Pfloat delta_y; /* y magnitude */
} Pvec;

COLOUR MAPPING INDEX (PEX_COLR_MAPPING_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

CURVE APPROXIMATION CRITERIA (PELEM_CURVE_APPROX_CRIT)
A curv_approx structure defined in Pelem_data as:

struct {
    Pint type; /* approximation type */
    Pfloat value; /* approximation value */
} curv_approx;

DEPTH CUE INDEX (PELEM_DCUE_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

EDGE COLOUR (PELEM_EDGE_COLR)
A Pgcolr structure provided in Pelem_data as:

Pgcolr colr; /* extended colour model */

where
typedef struct {
    Pint type; /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint ind; /* index in workstation colour bundle table */
    }
    struct {
        Pfloat x; /* red, hue, and so on */
    }
} Pgcolr;

modified 2 April 1993
Pfloat y; /* green, saturation, lightness, and so on */
Pfloat z; /* blue, value, saturation, and so on */

} general;
} val;
} Pgcolr;

EDGE COLOUR INDEX (PELEM_EDGE_COLR_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

EDGE FLAG (PELEM_EDGE_FLAG)
A Pedge_flag data structure provided in Pelem_data as:

Pedge_flag edge_flag; /* edge flag */
where typedef enum {
    PEDGE_OFF,
    PEDGE_ON
} Pedge_flag;

EDGE INDEX (PELEM_EDGE_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

EDGETYPE (PELEM_EDGETYPE)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */
The predefined line type values are:
    1 PLINE_SOLID
    2 PLINE_DASH
    3 PLINE_DOT
    4 PLINE_DASH_DOT

EDGEWIDTH SCALE FACTOR (PELEM_EDGEWIDTH)
A floating point provided in Pelem_data as:

Pfloat float_data; /* float valued data */

EXECUTE STRUCTURE (PELEM_EXEC_STRUCT)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

FACE CULLING MODE (PELEM_FACE_CULL_MODE)
An Pcull_mode data structure provided in Pelem_data as:
cull_mode; /* culling mode */

typedef enum {
    PCULL_NONE,
    PCULL_BACKFACE,
    PCULL_FRONTFACE
} Pcull_mode;

FACE DISTINGUISHING MODE (PELEM_FACE_DISTING_MODE)
An Pdisting_mode data structure provided in Pelem_data as:

disting_mode; /* distinguishing mode */

typedef enum {
    PDISTING_NO,
    PDISTING_YES
} Pdisting_mode;

FILL AREA (PELEM.FILL_AREA)
A Ppoint_list structure provided in Pelem_data as:

point_list; /* list of 2D points */

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

FILL AREA 3 (PELEM.FILL_AREA3)
A Ppoint_list3 structure provided in Pelem_data as:

point_list3; /* list of 3D points */

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

FILL AREA SET (PELEM.FILL_AREA3)
A Ppoint_list_list structure provided in Pelem_data as:

typedef struct {
    Pint num_point_lists; /* number of point lists */
    Ppoint_list *point_lists; /* list of point lists */
} Ppoint_list_list;
FILL AREA SET 3 (PELEM_FILL_AREA3)
A Ppoint_list_list3 structure provided in Pelem_data as:

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

FILL AREA SET 3 WITH DATA (PELEM_FILL_AREA_SET3_DATA)
A fasd3 structure defined in Pelem_data as:

struct {
    Pint fflag;         /* data specified per facet */
    Pint eflag;         /* edge visibility status */
    Pint vflag;         /* data per vertex flag */
    Pint colr_model;    /* colour type */
    Pfacet_data3 fdata; /* facet data */
    Pint nfa;           /* number of fill areas in the set */
    Pedge_data_list *edata; /* edge data */
    Pfacet_vdata_list3 *vdata; /* vertex data */
} fasd3;

GDP (PELEM_GDP)
A gdp structure defined in Pelem_data as:

struct {
    Pint id;           /* GDP id */
    Ppoint_list point_list; /* pts */
    Pgdp_data data;    /* data record */
} gdp;

GDP 3 (PELEM_GDP3)
A gdp3 structure defined in Pelem_data as:

struct {
    Pint id;           /* GDP3 id */
    Ppoint_list3 point_list; /* pts */
    Pgdp_data3 data;    /* data record */
} gdp3;

GLOBAL MODELLING TRANSFORMATION (PELEM_GLOBAL_MODEL_TRAN)
A Pmatrix structure provided in Pelem_data as:

modified 2 April 1993
Pmatrix global_tran; /* global transform */

where
typedef float Pmatrix[3][3];

GLOBAL MODELLING TRANSFORMATION 3 (PELEM_GLOBAL_MODEL_TRAN3)
A Pmatrix3 structure provided in Pelem_data as:

Pmatrix3 global_tran3; /* global transform3 */

where
typedef Pfloat Pmatrix3[4][4];

GSE (PELEM_GSE)
A gse structure defined in Pelem_data as:

struct {
    Pint id; /* GSE id */
    Pgse_data data; /* GSE data record */
} gse;

HLHSR IDENTIFIER (PELEM_HLHSR_ID)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

INDIVIDUAL ASF (PELEM_INDIV_ASF)
An asf structure defined in Pelem_data as:

struct {
    Paspect id; /* attribute id */
    Pasf source; /* asf */
} asf;

where
typedef enum {
    PASPECT_LINETYPE,
    PASPECT_LINEWIDTH,
    PASPECT_LINE_COLR_IND,
    PASPECT_MARKER_TYPE,
    PASPECT_MARKER_SIZE,
    PASPECT_MARKER_COLR_IND,
    PASPECT_TEXT_FONT,
    PASPECT_TEXT_PREC,
    PASPECT_CHAR_EXPAN,
    PASPECT_CHAR_SPACE,
    PASPECT_TEXT_COLR_IND,
    PASPECT_INT_STYLE,
    PASPECT_INT_STYLE_IND,
}
PASPECT_INT_COLR_IND,
PASPECT_EDGE_FLAG,
PASPECT_EDGETYPE,
PASPECT_EDGEBLACK,
PASPECT_EDGE_COLR_IND,
PASPECT_CURVE_APPROX_CRIT,
PASPECT_SURF_APPROX_CRIT,
PASPECT_LINE_SHAD_METH,
PASPECT_REFL_PROPS,
PASPECT_INT_REFL_EQN,
PASPECT_INT_SHAD_METH,
PASPECT_BACK_INT_STYLE,
PASPECT_BACK_INT_STYLE_IND,
PASPECT_BACK_INT_COLR,
PASPECT_BACK_REFL_PROPS,
PASPECT_BACK_INT_REFL_EQN,
PASPECT_BACK_INT_SHAD_METH
}

} Paspect;

and

typedef enum {
    PASF_BUNDLED,
    PASF_INDIV
} Pasf;

INTERIOR COLOUR (PELEM_INT_COLR)

A Pgcolr structure provided in Pelem_data as:

```
Pgcolr colr; /* extended colour model */
```

where

typedef struct {
    Pint type; /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint ind; /* index in workstation colour bundle table */
        struct {
            Pfloat x; /* red, hue, and so on */
            Pfloat y; /* green, saturation, lightness, and so on */
            Pfloat z; /* blue, value, saturation, and so on */
        } general;
    } val;

} Pgcolr;

modified 2 April 1993
INTERIOR COLOUR INDEX (PELEM_INT_COLR_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

INTERIOR INDEX (PELEM_INT_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

INTERIOR SHADING METHOD (PELEM_INT_SHAD_METH)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

The predefined shading method values are:

1 PSD_NONE No Shading
2 PSD_COLOUR Colour Interpolation Shading
3 PSD_DOT_PRODUCT Dot Product Interpolation Shading
4 PSD_NORMAL Normal Interpolation Shading

INTERIOR STYLE (PELEM_INT_STYLE)
An Pint_style data structure provided in Pelem_data as:

Pint_style int_style; /* interior style */

where

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

INTERIOR STYLE INDEX (PELEM_INT_STYLE_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

LABEL (PELEM_LABEL)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

LIGHT SOURCE STATE (PELEM_LIGHT_SRC_STATE)
An lss structure defined in Pelem_data as:

struct {
    Pint_list activation; /* activation list */
    Pint_list deactivation; /* deactivation list */

modified 2 April 1993
INQUIRE ELEMENT CONTENT (3P)

SunPHIGS Release 3.0

LINETYPE (PELEM_LINETYPE)

An integer provided in Pelem_data as:

\[
\text{Pint } \text{int}_\text{data}; /* integer valued data */
\]

The predefined line type values are:

1. PLINE_SOLID
2. PLINE_DASH
3. PLINE_DOT
4. PLINE_DASH_DOT

LINEWIDTH SCALE FACTOR (PELEM_LINEWIDTH)

A floating point provided in Pelem_data as:

\[
\text{Pfloat } \text{float}_\text{data}; /* float valued data */
\]

LOCAL MODELLING TRANSFORMATION (PELEM_LOCAL_MODEL_TRAN)

A local_tran structure defined in Pelem_data as:

\[
\text{struct } \\
\text{Pcompose}_\text{type } \text{comp}; /* composition type */ \\
\text{Pmatrix } \text{tran}; /* matrix */
\]

} local_tran;

where

typedef enum {

\text{PTYPE\_PRECONCAT,} \\
\text{PTYPE\_POSTCONCAT,} \\
\text{PTYPE\_REPLACE}

} \text{Pcompose}_\text{type};

LOCAL MODELLING TRANSFORMATION 3 (PELEM_LOCAL_MODEL_TRAN3)

A local_tran3 structure defined in Pelem_data as:

\[
\text{struct } \\
\text{Pcompose}_\text{type } \text{comp}; /* composition type */ \\
\text{Pmatrix3 } \text{tran}; /* matrix */
\]

} local_tran3;

where

typedef enum {

\text{PTYPE\_PRECONCAT,} \\
\text{PTYPE\_POSTCONCAT,} \\
\text{PTYPE\_REPLACE}

} \text{Pcompose}_\text{type};
MARKER COLOUR INDEX (PELEM_MARKER_COLR_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

MARKER SIZE SCALE FACTOR (PELEM_MARKER_SIZE)
A floating point provided in Pelem_data as:

Pfloat float_data; /* float valued data */

MARKER TYPE (PELEM_MARKER_TYPE)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */
The predefined marker type values are:

1 PMARKER_DOT
2 PMARKER_PLUS
3 PMARKER_ASTERISK
4 PMARKER_CIRCLE
5 PMARKER_CROSS

MODELING CLIPPING INDICATOR (PELEM_MODEL_CLIP_IND)
A Pclip_ind structure provided in Pelem_data as:

Pclip_ind clip_ind; /* clipping indicator */

where
typedef enum {
    PIND_NO_CLIP,
PIND_CLIP
} Pclip_ind;

MODELING CLIPPING VOLUME (PELEM_MODEL_CLIP_VOL)
A model_clip structure defined in Pelem_data as:

struct {
    Pint op; /* operator */
    Phalf_space_list half_spaces; /* half-space list */
} model_clip;

MODELING CLIPPING VOLUME 3 (PELEM_MODEL_CLIP_VOL3)
A model_clip3 structure defined in Pelem_data as:

struct {
    Pint op; /* operator */
    Phalf_space_list3 half_spaces; /* half-space list */
} model_clip3;
NIL (PELEM_NIL)
No data involved

NON-UNIFORM B-SPLINE CURVE (PELEM_NUNI_BSP_CURVE)
A nurb_curve structure defined in Pelem_data as:

```c
struct {
    Pint order; /* curve order */
    Pfloat_list knots; /* knots */
    Prational rationality; /* rationality specifier */
    Ppoint_list34 cpts; /* control points */
    Pfloat min; /* curve min */
    Pfloat max; /* curve max */
} nurb_curve;
```

NON-UNIFORM B-SPLINE SURFACE (PELEM_NUNI_BSP_SURF)
A nurb_surf structure defined in Pelem_data as:

```c
struct {
    Pint u_order; /* order of u surface */
    Pint v_order; /* order of v surface */
    Prational rationality; /* rationality specifier */
    Pfloat_list uknots; /* knots in u dimension */
    Pfloat_list vknots; /* knots in v dimension */
    Ppoint_grid34 grid; /* grid of control points */
    Pint num_trim_loops; /* number of trim curve loops */
    Ptrimcurve_list *trim_loops; /* trim curve loops */
} nurb_surf;
```

NUM_EL_TYPES (PELEM_NUM_EL_TYPES)
No data involved

PARAMETRIC SURFACE CHARACTERISTICS (PELEM_PARA_SURF_CHARACS)
A para_surf_characs structure provided in Pelem_data as:

```c
struct {
    Pint type;
    Ppara Surf_characs data;
} para_surf_characs; /* parametric surface characteristics */
```

where

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

PATTERN REFERENCE POINT (PELEM_PAT_REF_POINT)
A Ppoint structure provided in Pelem_data as:
    Ppoint point;    /* pattern reference pt */
where
typedef struct {
    Pfloat x;    /* x coordinate */
    Pfloat y;    /* y coordinate */
} Ppoint;

PATTERN REFERENCE POINT AND VECTORS (PELEM_PAT_REF_POINT_VECS)
A pat_ref_point_vecs structure defined in Pelem_data as:
struct {
    Ppoint3 ref_point;    /* pattern ref. pt */
    Pvec3 ref_vec[2];    /* vectors */
} pat_ref_point_vecs;

PATTERN SIZE (PELEM_PAT_SIZE)
A Ppoint structure provided in Pelem_data as:
    Ppoint point;    /* pattern size */
where
typedef struct {
    Pfloat x;    /* x coordinate */
    Pfloat y;    /* y coordinate */
} Ppoint;
PICK ID (PELEM_PICK_ID)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

POLYLINE (PELEM_POLYLINE)
A Ppoint_list structure provided in Pelem_data as:

Ppoint_list point_list; /* list of 2D points */
where typedef struct {
  Pint num_points; /* number of Ppoints */
  Ppoint *points; /* array of points */
} Ppoint_list;

POLYLINE 3 (PELEM_POLYLINE3)
A Ppoint_list3 structure provided in Pelem_data as:

Ppoint_list3 pts3; /* list of 3D points */
where typedef struct {
  Pint num_points; /* number of Ppoint3s */
  Ppoint3 *points; /* array of points */
} Ppoint_list3;

POLYLINE COLOUR (PELEM_LINE_COLR)
A Pgcolr structure provided in Pelem_data as:

Pgcolr colr; /* extended colour model */
where typedef struct {
  Pint type; /* indirect, RGB, CIE, HSV, HLS */
  union {
    Pint ind; /* index in workstation colour bundle table */
    struct {
      Pfloat x; /* red, hue, and so on */
      Pfloat y; /* green, saturation, lightness, and so on */
      Pfloat z; /* blue, value, saturation, and so on */
    } general;
  } val;
} Pgcolr;
POLYLINE COLOUR INDEX (PELEM_LINE_COLR_IND)
An integer provided in Pelem_data as:
    Pint int_data; /* integer valued data */

POLYLINE INDEX (PELEM_LINE_IND)
An integer provided in Pelem_data as:
    Pint int_data; /* integer valued data */

POLYLINE SET 3 WITH DATA (PELEM_POLYLINE_SET3_DATA)
A plsd3 structure defined in Pelem_data as:
    struct {
        Pint vflag; /* data per vertex flag */
        Pint colr_model; /* colour type */
        Pint npl; /* number of polylines in set */
        Pline_vdata_list3 *vdata; /* per line vertex data list */
    } plsd3;

POLYLINE SHADING METHOD (PELEM_LINE_SHAD_METH)
An integer provided in Pelem_data as:
    Pint int_data; /* integer valued data */
The predefined shading method values are:
    1 PSD_NONE No Shading
    2 PSD_COLOUR Colour Interpolation Shading
    3 PSD_DOT_PRODUCT
    4 PSD_NORMAL

POLYMARKER (PELEM_POLYMARKER)
A Ppoint_list structure provided in Pelem_data as:
    Ppoint_list pts; /* list of 2D points */
where typedef struct {
    Pint num_points; /* number of Ppoints */
    Ppoint *points; /* array of points */
} Ppoint_list;

POLYMARKER 3 (PELEM_POLYMARKER3)
A Ppoint_list3 structure provided in Pelem_data as:
    Ppoint_list3 point_list3; /* list of 3D points */
where typedef struct {
    Pint num_points; /* number of Ppoint3s */
} Ppoint_list3;
INQUIRE ELEMENT CONTENT (3P)

Ppoint3 *points; /* array of points */
} Ppoint_list3;

POLYMARKER COLOUR (PELEM_MARKER_COLR)
A Pgcolr structure provided in Pelem_data as:

Pgcolr colr; /* extended colour model */

where

typedef struct {
  Pint type; /* indirect, RGB, CIE, HSV, HLS */
  union {
    Pint ind; /* index in workstation colour bundle table */
    struct {
      Pfloat x; /* red, hue, and so on */
      Pfloat y; /* green, saturation, lightness, and so on */
      Pfloat z; /* blue, value, saturation, and so on */
    } general;
  } val;
} Pgcolr;

POLYMARKER INDEX (PELEM_MARKER_IND)
An integer provided in Pelem_data as:

Pint int_data; /* integer valued data */

QUADRILATERAL MESH 3 WITH DATA (PELEM_QUAD_MESH3_DATA)
A qmd3 structure defined in Pelem_data as:

struct {
  Pint fflag; /* data per facet flag */
  Pint vflag; /* data per vertex flag */
  Pint colr_model; /* colour type */
  Pint_dim; /* dimension of cells */
  Pfacet_data_arr3 fdata; /* facet data */
  Pfacet_vdata_arr3 vdata; /* vertex data */
} qmd3;

REFLECTANCE EQUATION (PELEM_INT_REFL_EQN)
An integer provided in Pelem_data as:

modified 2 April 1993
Pint int_data; /* integer valued data */

The predefined reflectance equation values are:

1  PREFL_NONE  No Reflectance Calculation Performed
2  PREFL_AMBIENT Use Ambient Term
3  PREFL_AMB_DIFF Use Ambient and Diffuse Terms
4  PREFL_AMB_DIFF_SPEC Use Ambient, Diffuse, and Specular Terms

REFLECTANCE PROPERTIES (PELEM_REFL_PROPS)
A Prefl_props structure provided in Pelem_data as:
    Prefl_props properties; /* area properties */ where
typedef struct {
    Pfloat ambient_coef; /* ambient reflectance coefficient */
    Pfloat diffuse_coef; /* diffuse reflectance coefficient */
    Pfloat specular_coef; /* specular reflectance coefficient */
    Pgcolr specular_colr; /* specular colour */
    Pfloat specular_exp; /* specular exponent */
    Pfloat transpar_coef; /* transparency coefficient */
} Prefl_props;

REMOVE NAMES FROM SET (PELEM_REMOVE_NAMES_SET)
A Pint_list structure provided in Pelem_data as:
    Pint_list names /* name sets */ where
typedef struct {
    Pint num_ints; /* number of names in the set */
    Pint *ints; /* name set */
} Pint_list

RENDERING COLOUR MODEL (PELEM_RENDERING_COLR_MODEL)
An integer provided in Pelem_data as:
    Pint int_data; /* integer valued data */

RESTORE MODELLING CLIPPING VOLUME
(PELEM_RESTORE_MODEL_CLIP_VOL)
No data involved

SET OF FILL AREA SET 3 WITH DATA (PELEM_SET_OF_FILL_AREA_SET3_DATA)
A sofas3 structure defined in Pelem_data as:
    struct {

modified 2 April 1993
INQUIRE ELEMENT CONTENT (3P)

Pint fflag;
Pint eflag;
Pint vflag;
Pint colr_model;
Pint num_sets;
Pfacet_data_arr3 fdata;
Pedge_data_list_list *edata;
Pint_list_list *vlist;
Pfacet_vdata_list3 vdata;
{
} sofas3;

SURFACE APPROXIMATION CRITERIA (PELEM_SURF_APPROX_CRIT)
A surf_approx structure defined in Pelem_data as:
struct {
Pint type;  /* approximation type */
Pfloat u_val;  /* u approximation value */
Pfloat v_val;  /* v approximation value */
} surf_approx;

TEXT (PELEM_TEXT)
A text structure defined in Pelem_data as:
struct {
Ppoint pos;  /* text pt */
char *char_string;  /* text string */
} text;

TEXT 3 (PELEM_TEXT3)
A text3 structure defined in Pelem_data as:
struct {
Ppoint3 pos;  /* text pt */
Pvec3 dir[2];  /* direction vectors */
char *char_string;  /* text string */
} text3;

TEXT ALIGNMENT (PELEM_TEXT_ALIGN)
A Ptext_align structure provided in Pelem_data as:
Ptext_align text_align;  /* text alignment */
where typedef struct {
Phor_text_align hor;  /* horizontal component */
Pvert_text_align vert;  /* vertical component */
} Ptext_align;
TEXT COLOUR (PELEM_TEXT_COLR)
A Pgcolr structure provided in Pelem_data as:

```c
typedef struct {
    Pint type; /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint ind; /* index in workstation colour bundle table */
        struct {
            Pfloat x; /* red, hue, and so on */
            Pfloat y; /* green, saturation, lightness, and so on */
            Pfloat z; /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

TEXT COLOUR INDEX (PELEM_TEXT_COLR_IND)
An integer provided in Pelem_data as:

```c
Pint int_data; /* integer valued data */
```

TEXT FONT (PELEM_TEXT_FONT)
An integer provided in Pelem_data as:

```c
Pint int_data; /* integer valued data */
```

The predefined text font values are:

- 1 PFONT_MONO
- 2 PFONT_SIMPLEX
- -1 PFONT_DUPLEX
- -2 PFONT_COMPLEX
- -3 PFONT_SCRIPT_SIMPLEX
- -4 PFONT_ITALIC_SIMPLEX
- -5 PFONT_ITALIC_TRIPLEX
- -6 PFONT_SCRIPT_COMPLEX
- -7 PFONT_SCRIPT_TRIPLEX

TEXT INDEX (PELEM_TEXT_IND)
An integer provided in Pelem_data as:

```c
Pint int_data; /* integer valued data */
```

TEXT PATH (PELEM_TEXT_PATH)
A Ptext_path data structure provided in Pelem_data as:

```c
Ptext_path text_path; /* text path */
```
typedef enum {
    PPATH_RIGHT,
    PPATH_LEFT,
    PPATH_UP,
    PPATH_DOWN
} Ptext_path;

TEXT PRECISION (PELEM_TEXT_PREC)
A Ptext_prec data structure provided in Pelem_data as:
    Ptext_prec    text_prec; /* text precision */
where
typedef enum {
    PPREC_STRING,
    PPREC_CHAR,
    PPREC_STROKE
} Ptext_prec;

TRIANGLE STRIP 3 WITH DATA (PELEM_TRI_STRIP3_DATA)
A tsd3 structure defined in Pelem_data as:
    struct {
        Pint    fflag;     /* data per facet flag */
        Pint    vflag;     /* data per vertex flag */
        Pint    colr_model; /* colour type */
        Pint    nv;        /* number of vertices */
        Pfacet_data_arr3  fdata; /* facet data */
        Pfacet_vdata_arr3  vdata; /* vertex data */
    } tsd3;

VIEW INDEX (PELEM_VIEW_IND)
An integer provided in Pelem_data as:
    Pint    int_data; /* integer valued data */
FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

**STRID**  
Identifier of the structure containing the element for which to return the contents.

**ELENUM**  
Sequence number in the specified structure of the element for which to return the contents.

**IIL**  
Dimension of integer array **IA** in which the specified element integer data will be returned. The appropriate array size for the data to be returned can be obtained by calling INQUIRE ELEMENT TYPE AND SIZE and using the value returned in its **IL** parameter.

**IRL**  
Dimension of real array **RA** in which the specified element real data will be returned. The appropriate array size for the data to be returned can be obtained by calling INQUIRE ELEMENT TYPE AND SIZE and using the value returned in its **RL** parameter.

**ISL**  
Dimension of integer array **LSTR** and character array **STR** in which the specified element character data will be returned. The appropriate array size for the data to be returned can be obtained by calling INQUIRE ELEMENT TYPE AND SIZE and using the value returned in its **SL** parameter.

FORTRAN Output Parameters

**ERRIND**  
The error number of any error detected by this function.

**IL**  
The number of entries returned in the **IA** array.

**IA**  
The integer values contained in the specified element.

**RL**  
The number of entries returned in the **RA** array.

**RA**  
The real values contained in the specified element.

**SL**  
The number of entries returned in the **LSTR** and **STR** arrays.

**LSTR**  
**SL** integers specifying the lengths of the **SL** character strings returned in **STR**.

**STR**  
The character data contained in the specified element.

The contents of the various arrays are determined by the element type, as described below.

**ADD NAMES TO SET (PEADS)**

- **IL** = number of names in the set
- **IA** = array of name set elements
- **RL** = 0
- **RA** = ()
- **SL** = 0
- **LSTR** = ()
- **STR** = ()

modified 2 April 1993
ANNOTATION STYLE (PEANST)

\[ IL = 1 \]
\[ IA(1) = \text{annotation style} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

ANNOTATION TEXT ALIGNMENT (PEATAL)

\[ IL = 2 \]
\[ IA(1) = \text{horizontal text alignment (PAHNOR, PALEFT, PACENT, PARITE)} \]
\[ IA(2) = \text{vertical text alignment (PAVNOR, PATOP, PACAP, PAHALF, PABASE, PABOTT)} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

ANNOTATION TEXT CHARACTER HEIGHT (PEATCH)

\[ IL = 0 \]
\[ IA = () \]
\[ RL = 1 \]
\[ RA(1) = \text{annotation text character height} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

ANNOTATION TEXT CHARACTER UP VECTOR (PEATCU)

\[ IL = 0 \]
\[ IA = () \]
\[ RL = 2 \]
\[ RA(1) = x \text{ component of annotation text character up vector} \]
\[ RA(2) = y \text{ component of annotation text character up vector} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

ANNOTATION TEXT PATH (PEATP)

\[ IL = 1 \]
\[ IA(1) = \text{annotation text path (PRIGHT, PLEFT, PUP, PDOWN)} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]
ANNE EXTRE TEXT RELATIVE (PEATR)

\[\begin{align*}
IL &= 0 \\
IA &= () \\
RL &= 4 \\
RA(1) &= \text{x-coordinate of reference point (MC)} \\
RA(2) &= \text{y-coordinate of reference point (MC)} \\
RA(3) &= \text{x-coordinate of annotation point (NPC)} \\
RA(4) &= \text{y-coordinate of annotation point (NPC)} \\
SL &= 1 \\
LSTR(1) &= \text{length of string} \\
STR(1) &= \text{string}
\end{align*}\]

ANNE EXTRE TEXT RELATIVE 3 (PEATR3)

\[\begin{align*}
IL &= 0 \\
IA &= () \\
RL &= 6 \\
RA(1) &= \text{x-coordinate of reference point (MC)} \\
RA(2) &= \text{y-coordinate of reference point (MC)} \\
RA(3) &= \text{z-coordinate of reference point (MC)} \\
RA(4) &= \text{x-coordinate of annotation point (NPC)} \\
RA(5) &= \text{y-coordinate of annotation point (NPC)} \\
RA(6) &= \text{z-coordinate of annotation point (NPC)} \\
SL &= 1 \\
LSTR(1) &= \text{length of string} \\
STR(1) &= \text{string}
\end{align*}\]

APPLICATION DATA (PEAP)

\[\begin{align*}
IL &= 0 \\
IA &= () \\
RL &= 0 \\
RA &= () \\
SL &= \text{number of application data records (LDR for PAP subroutine)} \\
LSTR(1) \text{ to } & \text{LSTR(}\text{SL}) = 80 \\
STR(1) \text{ to } & \text{STR(}\text{SL}) = \text{application data records (DATREC(1) to DATREC(}\text{SL}) \text{ for PAP subroutine)}
\end{align*}\]

BACK REFLECTANCE PROPERTIES (PEBAPR)

\[\begin{align*}
IL &= 2 \\
IA(1) &= \text{specular colour model} \\
IA(2) &= \text{indirect colour index (whether used or not)} \\
RL &= 8 \\
RA(1) &= \text{ambient reflectance coefficient} \\
RA(2) &= \text{diffuse reflectance coefficient} \\
RA(3) &= \text{specular reflectance coefficient} \\
RA(4) &= \text{specular exponent} \\
RA(5) &= \text{transparency coefficient}
\end{align*}\]

modified 2 April 1993
RA(6) through RA(8) contains specular direct colour components (whether used or not)

\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**BACK INTERIOR COLOUR (PEBIC)**

\[ IL = 2 \]
\[ IA(1) = \text{colour model} \]
\[ IA(2) = \text{indirect colour index (whether used or not)} \]
\[ RL = 3 \]
\[ RA(1) \text{ through } RA(3) \text{ contain direct colour components (whether used or not)} \]

\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**BACK INTERIOR REFLECTANCE EQUATION (PEBIRE)**

\[ IL = 1 \]
\[ IA(1) = \text{back interior reflectance equation} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**BACK INTERIOR SHADING METHOD (PEBISM)**

\[ IL = 1 \]
\[ IA(1) = \text{back interior shading method} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**BACK INTERIOR STYLE (PEBIS)**

\[ IL = 1 \]
\[ IA(1) = \text{back interior style (PHOLLO, PSOLID, PPATTR, PHATCH, PISEMP, PGENER)} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**BACK INTERIOR STYLE INDEX (PEBISI)**

\[ IL = 1 \]
\[ IA(1) = \text{back interior style index} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**CELL ARRAY (PECA)**

\[ IL = 2 + (IA(1) \times IA(2)) \]
\[ IA(1) = x \text{ dimension of cell index array} \]
\[ IA(2) = y \text{ dimension of cell index array} \]
\[ IA(3) \text{ to } IA(((IA(1) \times IA(2)) + 2) = \text{cell index array in column major order} \]
\[ \text{e.g. } IA(3) = COLIA(1,1), IA(4) = COLIA(2,1), ... \]

\[ RL = 4 \]
\[ RA(1) = x\text{-coordinate of P (MC)} \]
\[ RA(2) = y\text{-coordinate of P (MC)} \]
\[ RA(3) = x\text{-coordinate of Q (MC)} \]
\[ RA(4) = y\text{-coordinate of Q (MC)} \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**CELL ARRAY 3 (PECA3)**

\[ IL = 2 + (IA(1) \times IA(2)) \]
\[ IA(1) = x \text{ dimension of cell index array} \]
\[ IA(2) = y \text{ dimension of cell index array} \]
\[ IA(3) \text{ to } IA(((IA(1) \times IA(2)) + 2) = \text{cell index array in column major order} \]
\[ \text{e.g. } IA(3) = COLIA(1,1), IA(4) = COLIA(2,1), ... \]

\[ RL = 9 \]
\[ RA(1) = x\text{-coordinate of P (MC)} \]
\[ RA(2) = y\text{-coordinate of P (MC)} \]
\[ RA(3) = z\text{-coordinate of P (MC)} \]
\[ RA(4) = x\text{-coordinate of Q (MC)} \]
\[ RA(5) = y\text{-coordinate of Q (MC)} \]
\[ RA(6) = z\text{-coordinate of Q (MC)} \]
\[ RA(7) = x\text{-coordinate of R (MC)} \]
\[ RA(8) = y\text{-coordinate of R (MC)} \]
\[ RA(9) = z\text{-coordinate of R (MC)} \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**CELL ARRAY 3 PLUS (PECAP3)**

\[ IL = 5 + IA(4) \]
\[ IA(1) = \text{colourtype} \]
\[ IA(2) = x \text{ dimension of cell index array} \]
\[ IA(3) = y \text{ dimension of cell index array} \]
\[ IA(4) = \text{number of indirect colour cells} \]
\[ IA(5) = \text{number of direct colour cells} \]
INQUIRE ELEMENT CONTENT (3P)  SunPHIGS Release 3.0

\[ IA(6) \text{ to } IA(IA(4)+2) = \text{indirect colour cells in column major} \]
\[ RL = 9+IA(5) \]
\[ RA(1) = x\text{-coordinate of } P \text{ (MC)} \]
\[ RA(2) = y\text{-coordinate of } P \text{ (MC)} \]
\[ RA(3) = z\text{-coordinate of } P \text{ (MC)} \]
\[ RA(4) = x\text{-coordinate of } Q \text{ (MC)} \]
\[ RA(5) = y\text{-coordinate of } Q \text{ (MC)} \]
\[ RA(6) = z\text{-coordinate of } Q \text{ (MC)} \]
\[ RA(7) = x\text{-coordinate of } R \text{ (MC)} \]
\[ RA(8) = y\text{-coordinate of } R \text{ (MC)} \]
\[ RA(9) = z\text{-coordinate of } R \text{ (MC)} \]
\[ RA(10) \text{ to } RA(IA(5)+9) = \text{direct colour cells components in } (r_1,g_1,b_1,r_2,g_2,b_2...) \text{ form} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

CHARACTER EXPANSION FACTOR (PECHXP)
\[ IL = 0 \]
\[ IA = () \]
\[ RL = 1 \]
\[ RA(1) = \text{character expansion factor} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

CHARACTER HEIGHT (PECHH)
\[ IL = 0 \]
\[ IA = () \]
\[ RL = 1 \]
\[ RA(1) = \text{character height} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

CHARACTER SPACING (PECHSP)
\[ IL = 0 \]
\[ IA = () \]
\[ RL = 1 \]
\[ RA(1) = \text{character spacing} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

CHARACTER UP VECTOR (PECHUP)
\[ IL = 0 \]
\[ IA = () \]
\( RL = 2 \)
\( RA(1) = x \) component of character up vector
\( RA(2) = y \) component of character up vector
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

**COLOUR MAPPING INDEX (PSCMI)**
\( IL = 1 \)
\( IA(1) = \) colour mapping index
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

**CURVE APPROXIMATION CRITERIA (PECAC)**
\( IL = 1 \)
\( IA(1) = \) approximation type
\( RL = 1 \)
\( RA(1) = \) approximation value
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

**DEPTH CUE INDEX (PEDC)**
\( IL = 1 \)
\( IA(1) = \) depth cue index
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

**EDGE COLOUR (PEEDC)**
\( IL = 2 \)
\( IA(1) = \) colour model
\( IA(2) = \) indirect colour index (whether used or not)
\( RL = 3 \)
\( RA(1) \) through \( RA(3) \) contain direct colour components (whether used or not)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

**EDGE COLOUR INDEX (PEEDCI)**
\( IL = 1 \)
\( IA(1) = \) edge colour index
\( RL = 0 \)
INQUIRE ELEMENT CONTENT (3P) SunPHIGS Release 3.0

RA = ()
SL = 0
LSTR = ()
STR = ()

EDGE FLAG (PEEDFG)
IL = 1
IA(1) = edge flag (POFF, PON)
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

EDGE INDEX (PEEDI)
IL = 1
IA(1) = edge index
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

EDGETYPE (PEEDT)
IL = 1
IA(1) = edge type
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

EDGEWIDTH SCALE FACTOR (PEEWSC)
IL = 0
IA = ()
RL = 1
RA(1) = edgewidth scale factor
SL = 0
LSTR = ()
STR = ()

EXECUTE STRUCTURE (PEEXST)
IL = 1
IA(1) = structure identifier
RL = 0
RA = ()
SL = 0
LSTR = ()
\[ STR = () \]

**FACE CULLING MODE (PEFCM)**

\[ IL = 1 \]
\[ IA(1) = \text{face culling mode} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**FACE DISTINGUISHING MODE (PEFDM)**

\[ IL = 1 \]
\[ IA(1) = \text{face distinguishing mode} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**FILL AREA (PEFA)**

\[ IL = 1 \]
\[ IA(1) = \text{number of points in the fill area} \]
\[ RL = 2*IA(1) \]
\[ RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the fill area} \]
\[ \text{elements } IA(1)+1 \text{ through } 2*IA(1) \text{ contain the } y \text{ components of the fill area} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**FILL AREA 3 (PEFA3)**

\[ IL = 1 \]
\[ IA(1) = \text{number of points in the fill area 3} \]
\[ RL = 3*IA(1) \]
\[ RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the fill area 3} \]
\[ \text{elements } IA(1)+1 \text{ through } 2*IA(1) \text{ contain the } y \text{ components of the fill area 3} \]
\[ \text{elements } 2*IA(1)+1 \text{ through } 3*IA(1) \text{ contain the } z \text{ components of the fill area 3} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**FILL AREA SET (PEFAS)**

\[ IL = \text{number of point lists in fill area set} \]
\[ IA() = \text{array of end indices to point lists in fill area set} \]
\[ RL = 2*IA(IL) \]
\[ RA = \text{elements 1 through } IA(IL) \text{ contain the } x \text{ components of the fill area} \]

modified 2 April 1993
fill area set elements $IA(IL)+1$ through $2*IA(IL)$ contain the $y$
components of the fill area set.

The first fill area in the set is defined by the $x$, $y$ coordinates
from 1 to $IA(1)$, the second fill area is defined by the points
from $IA(1) + 1$ to $IA(2)$, and so on.

$SL = 0$
$LSTR = ()$
$STR = ()$

**FILL AREA SET 3 (PEFAS3)**

$IL = \text{number of point lists in fill area set 3}$
$IA() = \text{array of end indices to point lists in fill area set 3}$
$RL = 3*IA(IL)$

$RA = \text{elements 1 through } IA(IL) \text{ contain the } x \text{ components of the fill}$
-area set 3 elements $IA(IL)+1$ through $2*IA(IL)$ contain the $y$ components
-of the fill area set 3 elements $2*IA(IL)+1$ through $3*IA(IL)$ contain the
$z$ components of the fill area set 3.

The first fill area in the set is defined by the $x$, $y$, $z$ coordinates
from 1 to $IA(1)$, the second fill area is defined by the points
from $IA(1) + 1$ to $IA(2)$, and so on.

$SL = 0$
$LSTR = ()$
$STR = ()$

**FILL AREA SET 3 WITH DATA (PEFSD3)**

$IL = 11+IA(6)+IA(8)+IA(9)$
$IA(1) = \text{data per facet flag}$
$IA(2) = \text{data per edge flag}$
$IA(3) = \text{data per vertex flag}$
$IA(4) = \text{colour type}$
$IA(5) = \text{indirect facet colour (whether used or not)}$
$IA(6) = \text{number of fill areas}$
$IA(7) = \text{number of vertices}$
$IA(8) = \text{number of edge data}$
$IA(9) = \text{number of indirect vertex colours}$
$IA(10) = \text{number of vertex direct colour components}$
$IA(11) = \text{number of vertex normal components}$

$IA(12)$ through $IA(11+IA(6))$ contain fill area end indices
$IA(12+IA(6))$ through $IA(11+IA(6)+IA(8))$ contain edge visibility flags
$IA(12+IA(6)+IA(8))$ through $IA(11+IA(6)+IA(8)+IA(9))$ contain indirect vertex
-colours

$RL = 6+3*IA(7)+IA(10)+IA(11)$

$RA(1)$ through $RA(3)$ contain facet direct colour information (whether used
or not)

$RA(4)$ through $RA(6)$ contain facet normals (whether used or not)
RA(7) through RA(6+3*IA(7)) contain vertex coordinates in (x1,y1,z1,x2,x3,z3...) order that matches the COORDS(3,NV) format.
RA(7+3*IA(7)) through RA(6+3*IA(7)+IA(10)) contain direct vertex colour information in the order that matches the VCOLR(3,NV) format; for example, in PRGB mode the order will be (r1,g1,b1,r2,g2,b2...).
RA(7+3*IA(7)+IA(10)) through RA(6+3*IA(7)+IA(10)+IA(11)) contain vertex normals in the order that matches the VNORM(3,NV) format.

SL = 0
LSTR = ( )
STR = ( )

GDP (PEGDP)

IL = 2
IA(1) = number of points in the generalized drawing primitive
IA(2) = generalized drawing primitive identifier
RL = 2*IA(1)
RA = elements 1 through IA(1) contain the x components of the GDP point list
    elements IA(1)+1 through 2*IA(1) contain the y components of the GDP point list
SL = number of 80 character data records (LDR for GDP subroutine)
LSTR(1) to LSTR(SL) = 80
STR(1) to STR(SL) = packed GDP data record (DATREC(1) to DATREC(SL) for PGDP subroutine)

GDP 3 (PEGDP3)

IL = 2
IA(1) = number of points in the generalized drawing primitive 3
IA(2) = generalized drawing primitive 3 identifier
RL = 3*IA(1)
RA = elements 1 through IA(1) contain the x components of the GDP 3 point list
    elements IA(1)+1 through 2*IA(1) contain the y components of the GDP 3 point list
    elements 2*IA(1)+1 through 3*IA(1) contain the z components of the GDP 3 point list
SL = number of 80 character data records (LDR for PGDP3 subroutine)
LSTR(1) to LSTR(SL) = 80
STR(1) to STR(SL) = packed GDP 3 data record (DATREC(1) to DATREC(SL) for PGDP3 subroutine)

GLOBAL MODELLING TRANSFORMATION (PEGMT)

IL = 0
IA = ( )
RL = 9
RA(1) = (1,1) component of global transformation matrix
RA(2) = (2,1) component of global transformation matrix
RA(3) = (3,1) component of global transformation matrix
RA(4) = (1,2) component of global transformation matrix
$RA(5) = (2,2)$ component of global transformation matrix
$RA(6) = (3,2)$ component of global transformation matrix
$RA(7) = (1,3)$ component of global transformation matrix
$RA(8) = (2,3)$ component of global transformation matrix
$RA(9) = (3,3)$ component of global transformation matrix
$SL = 0$
$LSTR = ()$
$STR = ()$

GLOBAL MODELLING TRANSFORMATION 3 (PEGMT3)

$IL = 0$
$IA = ()$
$RL = 16$

$RA(1) = (1,1)$ component of global transformation matrix
$RA(2) = (2,1)$ component of global transformation matrix
$RA(3) = (3,1)$ component of global transformation matrix
$RA(4) = (4,1)$ component of global transformation matrix
$RA(5) = (1,2)$ component of global transformation matrix
$RA(6) = (2,2)$ component of global transformation matrix
$RA(7) = (3,2)$ component of global transformation matrix
$RA(8) = (4,2)$ component of global transformation matrix
$RA(9) = (1,3)$ component of global transformation matrix
$RA(10) = (2,3)$ component of global transformation matrix
$RA(11) = (3,3)$ component of global transformation matrix
$RA(12) = (4,3)$ component of global transformation matrix
$RA(13) = (1,4)$ component of global transformation matrix
$RA(14) = (2,4)$ component of global transformation matrix
$RA(15) = (3,4)$ component of global transformation matrix
$RA(16) = (4,4)$ component of global transformation matrix
$SL = 0$
$LSTR = ()$
$STR = ()$

GSE (PEGSE)

$IL = 1$
$IA(1) = generalized structure element identifier$
$RL = 0$
$RA = ()$

$SL =$ number of GSE
data records ($LDR$ for
  PGSE subroutine)
$LSTR(1)$ to $LSTR(SL) = 80$

$STR(1)$ to $STR(SL) =$ packed GSE data record ($DATREC(1)$ to $DATREC(SL)$ for
  PGSE subroutine)

modified 2 April 1993
HLHSR_IDENTIFIER (PEHRID)
\[ IL = 1 \]
\[ IA(1) = \text{HLHSR identifier} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

INDIVIDUAL ASF (PEIASF)
\[ IL = 2 \]
\[ IA(1) = \text{attribute identifier (PLN, PLWSC, PPLCI, PMK, PMKSC, PPMCI, PTXFN, PTXPR, PCHXP, PCHSP, PTXCI, PIS, PISI, PICI, PEDFG, PEDTY, PEWSC, PEDCI)} \]
\[ IA(2) = \text{aspect source flag value (PBUNDL, PINDIV)} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

INTERIOR COLOUR (PEIC)†
\[ IL = 2 \]
\[ IA(1) = \text{colour model} \]
\[ IA(2) = \text{indirect colour index (whether used or not)} \]
\[ RL = 3 \]
\[ RA(1) \text{ through } RA(3) \text{ contain direct colour components (whether used or not)} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

INTERIOR INDEX (PEII)
\[ IL = 1 \]
\[ IA(1) = \text{interior index} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

INTERIOR SHADING METHOD (PEISM)†
\[ IL = 1 \]
\[ IA(1) = \text{interior shading method} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]
**INTERIOR STYLE (PEIS)**

\[ IL = 1 \]
\[ IA(1) = \text{interior style (PHOLLO, PSOLID, PPATTR, PHATCH, PISEMP, PGENER)} \]
\[ RL = 0 \]
\[ RA = ( ) \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**INTERIOR STYLE INDEX (PEISI)**

\[ IL = 1 \]
\[ IA(1) = \text{interior style index} \]
\[ RL = 0 \]
\[ RA = ( ) \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**LABEL (PELB)**

\[ IL = 1 \]
\[ IA(1) = \text{label identifier} \]
\[ RL = 0 \]
\[ RA = ( ) \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**LIGHT SOURCE STATE (PELSS)**

\[ IL = 2 + IA(1) + IA(2) \]
\[ IA(1) = \text{length of activation list} \]
\[ IA(2) = \text{length of deactivation list} \]
\[ IA(3) \text{ through } IA(2 + IA(1)) \text{ contain light source activation list} \]
\[ IA(3 + IA(1)) \text{ through } IA(2 + IA(1) + IA(2)) \text{ contain light source deactivation list} \]
\[ RL = 0 \]
\[ RA = ( ) \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

**LINETYPE (PELN)**

\[ IL = 1 \]
\[ IA(1) = \text{linetype} \]
\[ RL = 0 \]
\[ RA = ( ) \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]
LINEWIDTH SCALE FACTOR (PELWSC)

\[ IL = 0 \]
\[ IA = ( ) \]
\[ RL = 1 \]
\[ RA(1) = \text{linewidth scale factor} \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

LOCAL MODELLING TRANSFORMATION (PELM)

\[ IL = 1 \]
\[ IA = \text{composition type (PCPRE, PCPOST, PCREPL)} \]
\[ RL = 9 \]
\[ RA(1) = (1,1) \text{ component of local transformation matrix} \]
\[ RA(2) = (2,1) \text{ component of local transformation matrix} \]
\[ RA(3) = (3,1) \text{ component of local transformation matrix} \]
\[ RA(4) = (1,2) \text{ component of local transformation matrix} \]
\[ RA(5) = (2,2) \text{ component of local transformation matrix} \]
\[ RA(6) = (3,2) \text{ component of local transformation matrix} \]
\[ RA(7) = (1,3) \text{ component of local transformation matrix} \]
\[ RA(8) = (2,3) \text{ component of local transformation matrix} \]
\[ RA(9) = (3,3) \text{ component of local transformation matrix} \]
\[ SL = 0 \]
\[ LSTR = ( ) \]
\[ STR = ( ) \]

LOCAL MODELLING TRANSFORMATION 3 (PELM3)

\[ IL = 1 \]
\[ IA = \text{composition type (PCPRE, PCPOST, PCREPL)} \]
\[ RL = 16 \]
\[ RA(1) = (1,1) \text{ component of local transformation matrix} \]
\[ RA(2) = (2,1) \text{ component of local transformation matrix} \]
\[ RA(3) = (3,1) \text{ component of local transformation matrix} \]
\[ RA(4) = (4,1) \text{ component of local transformation matrix} \]
\[ RA(5) = (1,2) \text{ component of local transformation matrix} \]
\[ RA(6) = (2,2) \text{ component of local transformation matrix} \]
\[ RA(7) = (3,2) \text{ component of local transformation matrix} \]
\[ RA(8) = (4,2) \text{ component of local transformation matrix} \]
\[ RA(9) = (1,3) \text{ component of local transformation matrix} \]
\[ RA(10) = (2,3) \text{ component of local transformation matrix} \]
\[ RA(11) = (3,3) \text{ component of local transformation matrix} \]
\[ RA(12) = (4,3) \text{ component of local transformation matrix} \]
\[ RA(13) = (1,4) \text{ component of local transformation matrix} \]
\[ RA(14) = (2,4) \text{ component of local transformation matrix} \]
\[ RA(15) = (3,4) \text{ component of local transformation matrix} \]
\[ RA(16) = (4,4) \text{ component of local transformation matrix} \]
SL = 0
LSTR = ()
STR = ()

MARKER COLOUR INDEX (PEPMCI)
IL = 1
IA(1) = polymarker colour index
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

MARKER SIZE SCALE FACTOR (PEMKSC)
IL = 0
IA = ()
RL = 1
RA(1) = marker size scale factor
SL = 0
LSTR = ()
STR = ()

MARKER TYPE (PEMK)
IL = 1
IA(1) = marker type
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

MODELLING CLIPPING INDICATOR (PEMCLI)
IL = 1
IA(1) = modelling clipping indicator (PNCLIP, PCLIP)
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

MODELLING CLIPPING VOLUME (PEMCV)
IL = 2
IA(1) = modelling clipping operator
IA(2) = number of modelling clipping half-spaces in list
RL = 4*IA(2)
for i = 0 to IA(2)\(mi1
RA((4*i)+1) = x-coordinate of point defining plane of half-space (MC)
RA((4*i)+2) = y-coordinate of point defining plane of half-space (MC)

modified 2 April 1993
$RA((4*i)+3) = dx$ component of normal vector defining the plane of half-space (MC)

$RA((4*i)+4) = dy$ component of normal vector defining the plane of half-space (MC)

**MODELLING CLIPPING VOLUME 3 (PEMVC3)**

$IL = 2$

$IA(1) =$ modelling clipping operator

$IA(2) =$ number of modelling clipping half-spaces in list

$RL = 6*iA(2)$

for $i = 0$ to $IA(2)-1$

$RA((6*i)+1) =$ $x$-coordinate of point defining plane of half-space (MC)

$RA((6*i)+2) =$ $y$-coordinate of point defining plane of half-space (MC)

$RA((6*i)+3) =$ $z$-coordinate of point defining plane of half-space (MC)

$RA((6*i)+4) =$ $dx$ component of normal vector defining the plane of half-space (MC)

$RA((6*i)+5) =$ $dy$ component of normal vector defining the plane of half-space (MC)

$RA((6*i)+6) =$ $dz$ component of normal vector defining the plane of half-space (MC)

**NIL (PENIL)**

$IL = 0$

$IA = ()$

$RL = 0$

$RA = ()$

$SL = 0$

$LSTR = ()$

$STR = ()$

**NON-UNIFORM B-SPLINE CURVE (PENBSC)**

$IL = 6$

$IA(1) =$ curve order

$IA(2) =$ number of knots

$IA(3) =$ index of first knot in the real array $RA$

$IA(4) =$ rationality selector (PRAT, PNRAT)

$IA(5) =$ number of control points

$IA(6) =$ index of first control point in the real array $RA$

$RL = 2+IA(2)+4*IA(5)$

$RA(1) =$ lower limit of parameter range

$RA(2) =$ upper limit of parameter range

$RA(IA(3))$ through $RA(IA(3)+IA(2))$ contains the knot values

$RA(IA(6))$ through $RA(IA(6)+IA(5))$ contains the control points.

$RA(IA(6))$ contains the $x$ coordinate of the first control point,

$RA(IA(6)+1)$ contains the $y$ coordinate of the first control point,

and so on
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**NON-UNIFORM B-SPLINE SURFACE (PENBSS)**

The trimming curve data is stored as separate lists within the \( IA \) and \( RA \) arrays for each parameter, as is specified in the corresponding element creation function. The indices of the first elements of these lists (embedded in the returned arrays) is returned by this function.

\[ IL = 21 + \text{variable size of trimming curve data} \]

\( IA(1) = \) surface \( u \) order
\( IA(2) = \) surface \( v \) order
\( IA(3) = \) rationality selector (\( PRAT, PNRAT \))
\( IA(4) = \) number of \( u \) knots
\( IA(5) = \) number of \( v \) knots
\( IA(6) = \) number of control points in \( u \) dimension
\( IA(7) = \) number of control points in \( v \) dimension
\( IA(8) = \) number of trimming loops
\( IA(9) = \) index of first surface \( u \) knot in the real array \( RA \)
\( IA(10) = \) index of first surface \( v \) knot in the real array \( RA \)
\( IA(11) = \) index of first control point in the real array \( RA \)
\( IA(12) = \) starting index in \( IA \) of the list of number of trimming curves per loop
\( IA(13) = \) starting index in \( IA \) of the list of trimming curve visibilities
\( IA(14) = \) starting index in \( IA \) of the list of trimming curve orders
\( IA(15) = \) starting index in \( IA \) of the list of trimming curve approximation types
\( IA(16) = \) starting index in \( IA \) of the list of trimming curve rationalities
\( IA(17) = \) starting index in \( IA \) of the list of trimming curve knot counts
\( IA(18) = \) starting index in \( IA \) of the list of trimming curve knot starting indices
\( IA(19) = \) starting index in \( IA \) of the list of trimming curve control point counts
\( IA(20) = \) starting index in \( IA \) of the list of trimming curve control point starting indices
\( IA(21) = \) starting index in \( RA \) of the list of trimming curve approximation values
\( IA(22) = \) starting index of the list of trimming curve lower parameter limits in the real array \( RA \)
\( IA(23) = \) starting index of the list of trimming curve upper parameter limits in the real array \( RA \)

\( RL = 2 + \text{\( IA(4) \)} + \text{\( IA(5) \)} + 4 \times \text{\( IA(6) \)} + \text{\( IA(7) \)} + \text{variable size of trimming curve data} \)

\( RA(IA(9)) \) through \( RA(IA(9) + IA(4)) \) contains the \( u \) knot values
\( RA(IA(10)) \) through \( RA(IA(10) + IA(5)) \) contains the \( v \) knot values

\( RA(IA(11)) \) through \( RA(IA(11) + 4 \times IA(6) + IA(7)) \) contains the surface control points.

\( RA(IA(11)) \) contains the \( x \) coordinate of the first control point,
\( RA(IA(11) + 1) \) contains the \( y \) coordinate of the first control point, and so on
The rest of $RA$ contains the variable length trimming curve data. The indices returned in the $IA$ array must be used to access this data.

$SL = 0$
$LSTR = ()$
$STR = ()$

PARAMETRIC SURFACE CHARACTERISTICS (PSPSC)

$IL = 5$
$IA(1) = parametric surface characteristics type$
$IA(2) = curve placement (for type PSCISO)$
$IA(3) = u_count (for type PSCISO)$
$IA(4) = v_count (for type PSCISO)$
$IA(5) = number of parameters (for type PSCLMC and PSCLWC)$

For parametric surface characteristics type PSCLMC and PSCLWC

$RL = 6 + IA(5)$
$RA(1) - RA(3) = x, y, z components of origin$
$RA(4) - RA(6) = x, y, z components of direction$
$RA(7) - RA(IA(5) + 6) = surface characteristics parameters$

otherwise,

$RL = 0$
$RA = ()$
$SL = 0$
$LSTR = ()$
$STR = ()$

PATTERN REFERENCE POINT (PEPARF)

$IL = 0$
$IA = ()$
$RL = 2$
$RA(1) = x-coordinate of pattern reference point (MC)$
$RA(2) = y-coordinate of pattern reference point (MC)$
$SL = 0$
$LSTR = ()$
$STR = ()$

PATTERN REFERENCE POINT AND VECTORS (PEPRPV)

$IL = 0$
$IA = ()$
$RL = 9$
$RA(1) = x-coordinate of pattern reference point (MC)$
$RA(2) = y-coordinate of pattern reference point (MC)$
$RA(3) = z-coordinate of pattern reference point (MC)$
$RA(4) = x-component of pattern reference vector 1 (MC)$
$RA(5) = y-component of pattern reference vector 1 (MC)$
$RA(6) = z-component of pattern reference vector 1 (MC)$

modified 2 April 1993
\[
RA(7) = x\text{-component of pattern reference vector 2 (MC)}
\]
\[
RA(8) = y\text{-component of pattern reference vector 2 (MC)}
\]
\[
RA(9) = z\text{-component of pattern reference vector 2 (MC)}
\]
\[
SL = 0
\]
LSTR = ( )
STR = ( )

PATTERN SIZE (PEPA)
\[
IL = 0
\]
\[
IA = ( )
\]
\[
RL = 2
\]
\[
RA(1) = x\text{-component of pattern size (MC)}
\]
\[
RA(2) = y\text{-component of pattern size (MC)}
\]
\[
SL = 0
\]
LSTR = ( )
STR = ( )

PICK ID (PEPKID)
\[
IL = 1
\]
\[
IA(1) = \text{pick identifier}
\]
\[
RL = 0
\]
\[
RA = ( )
\]
\[
SL = 0
\]
LSTR = ( )
STR = ( )

POLYLINE (PEPL)
\[
IL = 1
\]
\[
IA(1) = \text{number of points in the polyline}
\]
\[
RL = 2*IA(1)
\]
\[
RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the polyline}
\]
\[
\text{elements } IA(1)+1 \text{ through } 2*IA(1) \text{ contain the } y \text{ components of the polyline}
\]
\[
SL = 0
\]
LSTR = ( )
STR = ( )

POLYLINE 3 (PEPL3)
\[
IL = 1
\]
\[
IA(1) = \text{number of points in the polyline 3}
\]
\[
RL = 3*IA(1)
\]
\[
RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the polyline 3}
\]
\[
\text{elements } IA(1)+1 \text{ through } 2*IA(1) \text{ contain the } y \text{ components of the polyline 3}
\]
\[
\text{elements } 2*IA(1)+1 \text{ through } 3*IA(1) \text{ contain the}
\]
\[
z \text{ components of the polyline 3}
\]
\[
SL = 0
\]
LSTR = ( )
\[ STR = () \]

**POLYLINE COLOUR (PEPLC)**

\[ IL = 2 \]

\[ IA(1) = \text{colour model} \]

\[ IA(2) = \text{indirect colour index (whether used or not)} \]

\[ RL = 3 \]

\[ RA(1) \text{ through } RA(3) \text{ contain direct colour components (whether used or not)} \]

\[ SL = 0 \]

\[ LSTR = () \]

\[ STR = () \]

**POLYLINE COLOUR INDEX (PEPLCI)**

\[ IL = 1 \]

\[ IA(1) = \text{polyline colour index} \]

\[ RL = 0 \]

\[ RA = () \]

\[ SL = 0 \]

\[ LSTR = () \]

\[ STR = () \]

**POLYLINE INDEX (PEPLI)**

\[ IL = 1 \]

\[ IA(1) = \text{polyline index} \]

\[ RL = 0 \]

\[ RA = () \]

\[ SL = 0 \]

\[ LSTR = () \]

\[ STR = () \]

**POLYLINE SET 3 WITH DATA (PEPSD3)**

\[ IL = 6+IA(3)+IA(5) \]

\[ IA(1) = \text{data per vertex flag} \]

\[ IA(2) = \text{colour type} \]

\[ IA(3) = \text{number of point lists (polylines)} \]

\[ IA(4) = \text{number of vertices} \]

\[ IA(5) = \text{number of indirect vertex colours} \]

\[ IA(6) = \text{number of vertex direct colour components} \]

\[ IA(7) \text{ through } IA(6+IA(3)) \text{ contain the end indices for point lists} \]

\[ IA(7+IA(3)) \text{ through } IA(6+IA(3)+IA(5)) \text{ contain indirect vertex colour for each point} \]

\[ RL = 3*IA(4)+IA(6) \]

\[ RA(1) \text{ through } RA(3*IA(4)) \text{ contain vertex coordinates in (x1,y1,z1,x2,x3,z3...)} \] order that matches the COORDS(3,NV) format.

\[ RA(1+3*IA(4)) \text{ through } RA(3*IA(4)+IA(6)) \text{ contain direct vertex colour information in the order that matches the VCOLR(3,NV) format; for example, in PRGB mode the order will be (r1,g1,b1,r2,g2,b2...).} \]
INQUIRE ELEMENT CONTENT (3P)

\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

POLYLINE SHADING METHOD (PEPLSM)†

\[ IL = 1 \]
\[ IA(1) = \text{polyline shading method} \]
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

POLYMARKER (PEPM)

\[ IL = 1 \]
\[ IA(1) = \text{number of points in the polymarker} \]
\[ RL = 2 \times IA(1) \]
\[ RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the polymarker} \]
\[ \text{elements } IA(1)+1 \text{ through } 2 \times IA(1) \text{ contain the } y \text{ components of the polymarker} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

POLYMARKER 3 (PEPM3)

\[ IL = 1 \]
\[ IA(1) = \text{number of points in the polymarker 3} \]
\[ RL = 3 \times IA(1) \]
\[ RA = \text{elements 1 through } IA(1) \text{ contain the } x \text{ components of the polymarker 3} \]
\[ \text{elements } IA(1)+1 \text{ through } 2 \times IA(1) \text{ contain the } y \text{ components of the polymarker 3} \]
\[ \text{elements } 2 \times IA(1)+1 \text{ through } 3 \times IA(1) \text{ contain the } z \text{ components of the polymarker 3} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

POLYMARKER COLOUR (PEPMC)†

\[ IL = 2 \]
\[ IA(1) = \text{colour model} \]
\[ IA(2) = \text{indirect colour index (whether used or not)} \]
\[ RL = 3 \]
\[ RA(1) \text{ through } RA(3) \text{ contain direct colour components (whether used or not)} \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]
POLYMARKER INDEX (PEPMI)

IL = 1
IA(1) = polymarker index
RL = 0
RA = ()
SL = 0
LSTR = ()
STR = ()

QUADRILATERAL MESH 3 WITH DATA (PEQMD3+)

IL = 11+IA(6)+IA(7)
IA(1) = data per facet flag
IA(2) = data per vertex flag
IA(3) = colour type
IA(4) = number of vertices along y axis
IA(5) = number of vertices along x axis
IA(6) = number of indirect facet colours
IA(7) = number of indirect vertex colours
IA(8) = number of facet direct colour components
IA(9) = number of facet normal components
IA(10) = number of vertex direct colour components
IA(11) = number of vertex normal components
IA(12) through IA(11+IA(6)) contain indirect facet colours
IA(12+IA(6)) through IA(11+IA(6)+IA(7)) contain indirect vertex colours
RL = IA(8)+IA(9)+3*IA(4)*IA(5)+IA(10)+IA(11)
RA(1) through RA(IA(8)) contain facet direct colour information in the order that matches the FCOLR(3,*) format.
RA(1+IA(8)) through RA(IA(8)+IA(9)) contain facet normals in the order that matches the FNORM(3,*) format.
RA(1+IA(8)+IA(9)) through RA(IA(8)+IA(9)+3*IA(4)*IA(5)) contain vertex coordinates in (x1,y1,z1,x2,x3,z3,..) order that matches the COORDS(3,VROWS*VCOLS) format.
RA(1+IA(8)+IA(9)+3*IA(4)*IA(5)) through RA(IA(8)+IA(9)+3*IA(4)*IA(5)+IA(10)) contain direct vertex colour information in the order that matches the VCOLR(3,VROWS*VCOLS) format; for example, in PRGB mode the order will be (r1,g1,b1,r2,g2,b2,..).
RA(1+IA(8)+IA(9)+3*IA(4)*IA(5)+IA(10)) through RA(IA(8)+IA(9)+3*IA(4)*IA(5)+IA(10)+IA(11)) contain vertex normals in the order that matches the VNORM(3,VROWS*VCOLS) format.

SL = 0
LSTR = ()
STR = ()

REFLECTANCE EQUATION (PEIRE+)

IL = 1
IA(1) = interior reflectance equation
\( RL = 0 \)
\( RA = () \)
\( SL = 0 \)
\( LSTR = () \)
\( STR = () \)

**Reflectance Properties (PEAPR)**

- \( IL = 2 \)
- \( IA(1) = \) specular colour model
- \( IA(2) = \) indirect colour index (whether used or not)
- \( RL = 8 \)
- \( RA(1) = \) ambient reflectance coefficient
- \( RA(2) = \) diffuse reflectance coefficient
- \( RA(3) = \) specular reflectance coefficient
- \( RA(4) = \) specular exponent
- \( RA(5) = \) transparency coefficient
- \( RA(6) \) through \( RA(8) \) contains specular direct colour components (whether used or not)

\( SL = 0 \)
\( LSTR = () \)
\( STR = () \)

**Remove Names from Set (PERES)**

- \( IL = \) number of names in the set
- \( IA = \) array of name set elements
- \( RL = 0 \)
- \( RA = () \)
- \( SL = 0 \)
- \( LSTR = () \)
- \( STR = () \)

**Rendering Colour Model (PERCM)**

- \( IL = 1 \)
- \( IA(1) = \) rendering colour model
- \( RL = 0 \)
- \( RA = () \)
- \( SL = 0 \)
- \( LSTR = () \)
- \( STR = () \)

**Restore Modelling Clipping Volume (PERMVC)**

- \( IL = 0 \)
- \( IA = () \)
- \( RL = 0 \)
- \( RA = () \)
- \( SL = 0 \)
- \( LSTR = () \)
STR = (

SET OF FILL AREA SET 3 WITH DATA (PSFSD3)

IL = 15+IA(6)+IA(5)+IA(9)+IA(10)+IA(11)+IA(13)
IA(1) = data per facet flag
IA(2) = data per edge flag
IA(3) = data per vertex flag
IA(4) = colour type
IA(5) = number of facets (FILL AREA SETS)
IA(6) = number of indirect facet colours
IA(7) = number of facet colour components
IA(8) = number of facet normal components
IA(9) = dimension of end indices per facet array
IA(10) = dimension of vertex indices per facet array
IA(11) = dimension of edge data array
IA(12) = number of vertices
IA(14) = number of indirect vertex colours
IA(15) = number of vertex normal components
IA(16) through IA(15+IA(6)) contain indirect facet colours
IA(16+IA(6)) through IA(15+IA(6)+IA(5)) contain bounds per facet
IA(16+IA(6)+IA(5)) through IA(15+IA(6)+IA(5)+IA(9)) contain facets end indices
IA(16+IA(6)+IA(5)+IA(9)) through IA(15+IA(6)+IA(5)+IA(9)+IA(10)) contain vertex indices per facet
IA(16+IA(6)+IA(5)+IA(9)+IA(10)) through IA(15+IA(6)+IA(5)+IA(9)+IA(10)+IA(11)) contain edge visibility flags
IA(16+IA(6)+IA(5)+IA(9)+IA(10)+IA(11)) through IA(15+IA(6)+IA(5)+IA(9)+IA(10)+IA(11)+IA(13)) contain indirect vertex colours
RL = IA(7)+IA(8)+3*IA(12)+IA(14)+IA(15)
RA(1) through RA(IA(7)) contain facet direct colour information
RA(1+IA(7)) through RA(IA(7)+IA(8)) contain facet normals
RA(1+IA(7)+IA(8)) through RA(IA(7)+IA(8)+3*IA(12)) contain vertex coordinates in (x1,y1,z1,x2,x3,z3..) order that matches the COORDS(3,NV) format.
RA(1+IA(7)+IA(8)+3*IA(12)) through RA(IA(7)+IA(8)+3*IA(12)+IA(14)) contain direct vertex colour information in the order that matches the VCOLR(3,NV) format, for example, in PRGB mode the order will be (r1,g1,b1,r2,g2,b2...).
RA(1+IA(7)+IA(8)+3*IA(12)+IA(14)) through RA(IA(7)+IA(8)+3*IA(12)+IA(14)+IA(15)) contain vertex normals in the order that matches the VNORM(3,NV) format.

SL = 0
LSTR = ( )
STR = ( )

SURFACE APPROXIMATION CRITERIA (PESAC)

IL = 1
IA(1) = approximation type
\[
\begin{align*}
RL &= 2 \\
RA(1) &= u \text{ approximation value} \\
RA(2) &= v \text{ approximation value} \\
SL &= 0 \\
LSTR &= ( ) \\
STR &= ( ) \\
\end{align*}
\]

**TEXT (PETX)**

\[
\begin{align*}
IL &= 0 \\
IA &= ( ) \\
RL &= 2 \\
RA(1) &= x\text{-coordinate of text point (MC)} \\
RA(2) &= y\text{-coordinate of text point (MC)} \\
SL &= 1 \\
LSTR(1) &= \text{length of string} \\
STR(1) &= \text{string} \\
\end{align*}
\]

**TEXT 3 (PETX3)**

\[
\begin{align*}
IL &= 0 \\
IA &= ( ) \\
RL &= 9 \\
RA(1) &= x\text{-coordinate of text point (MC)} \\
RA(2) &= y\text{-coordinate of text point (MC)} \\
RA(3) &= z\text{-coordinate of text point (MC)} \\
RA(4) &= x\text{-coordinate of first text direction vector (MC)} \\
RA(5) &= y\text{-coordinate of first text direction vector (MC)} \\
RA(6) &= z\text{-coordinate of first text direction vector (MC)} \\
RA(7) &= x\text{-coordinate of second text direction vector (MC)} \\
RA(8) &= y\text{-coordinate of second text direction vector (MC)} \\
RA(9) &= z\text{-coordinate of second text direction vector (MC)} \\
SL &= 1 \\
LSTR(1) &= \text{length of string} \\
STR(1) &= \text{string} \\
\end{align*}
\]

**TEXT ALIGNMENT (PETXAL)**

\[
\begin{align*}
IL &= 2 \\
IA(1) &= \text{horizontal text alignment (PAHNR, PALEFT, PACENT, PARITE)} \\
IA(2) &= \text{vertical text alignment (PAVNR, PATOP, PACAP, PAHALF, PABASE, PABOTT)} \\
RL &= 0 \\
RA &= ( ) \\
SL &= 0 \\
LSTR &= ( ) \\
STR &= ( ) \\
\end{align*}
\]
TEXT COLOUR (PETXC)†

\( IL = 2 \)
\( IA(1) = \text{colour model} \)
\( IA(2) = \text{indirect colour index (whether used or not)} \)
\( RL = 3 \)
\( RA(1) \) through \( RA(3) \) contain direct colour components (whether used or not)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

TEXT COLOUR INDEX (PETXCI)

\( IL = 1 \)
\( IA(1) = \text{text colour index} \)
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

TEXT FONT (PETXFN)

\( IL = 1 \)
\( IA(1) = \text{text font} \)
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

TEXT INDEX (PETXI)

\( IL = 1 \)
\( IA(1) = \text{text index} \)
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

TEXT PATH (PETXP)

\( IL = 1 \)
\( IA(1) = \text{text path (PRIGHT, PLEFT, PUP, PDOWN)} \)
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)
TEXT PRECISION (PETXPR)

\( IL = 1 \)
\( IA(1) = \) text precision (STRP, PCHARP, STRKP)
\( RL = 0 \)
\( RA = ( ) \)
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

TRIANGLE STRIP 3 WITH DATA (PETRSD)†

\( IL = 10 + IA(5) + IA(6) \)
\( IA(1) = \) data per facet flag
\( IA(2) = \) data per vertex flag
\( IA(3) = \) colour type
\( IA(4) = \) number of vertices
\( IA(5) = \) number of indirect facet colours
\( IA(6) = \) number of indirect vertex colours
\( IA(7) = \) number of facet direct colour components
\( IA(8) = \) number of facet normal components
\( IA(9) = \) number of vertex direct colour components
\( IA(10) = \) number of vertex normal components
\( IA(11) \) through \( IA(10 + IA(5)) \) contain indirect facet colours
\( IA(11 + IA(5)) \) through \( IA(10 + IA(5) + IA(6)) \) contain indirect vertex colours
\( RL = IA(7) + IA(8) + 3 \times IA(4) + IA(9) + IA(10) \)
\( RA(1) \) through \( RA(IA(7)) \) contain facet direct colour information in the order which match the FCOLR(3,NV-2) format.
\( RA(1 + IA(7)) \) through \( RA(IA(7) + IA(8)) \) contain facet normals in the order that matches the FNORM(3,NV-2) format.
\( RA(1 + IA(7) + IA(8)) \) through \( RA(IA(7) + IA(8) + 3 \times IA(4)) \) contain vertex coordinates in \( (x_1,y_1,z_1,x_2,x_3,z_3,\ldots) \) order that matches the COORDS(3,NV) format.
\( RA(1 + IA(7) + IA(8) + 3 \times IA(4)) \) through \( RA(IA(7) + IA(8) + 3 \times IA(4) + IA(9)) \) contain direct vertex colour information in the order that matches the VCOLR(3,NV) format; for example, in PRGB mode the order will be \( (r_1,g_1,b_1,r_2,g_2,b_2,\ldots) \).
\( RA(1 + IA(7) + IA(8) + 3 \times IA(4) + IA(9)) \) through \( RA(IA(7) + IA(8) + 3 \times IA(4) + IA(9) + IA(10)) \) contain vertex normals in the order that matches the VNORM(3,NV) format
\( SL = 0 \)
\( LSTR = ( ) \)
\( STR = ( ) \)

†This is a SunPHIGS Extension that is based on PHIGS PLUS and is not a part of the PHIGS standard.

VIEW INDEX (PEVWI)

\( IL = 1 \)
\( IA(1) = \) view index
\[ RL = 0 \]
\[ RA = () \]
\[ SL = 0 \]
\[ LSTR = () \]
\[ STR = () \]

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state ((\text{PHOP}, *, *, *))</td>
</tr>
<tr>
<td>201</td>
<td>Ignoring function, the specified structure does not exist</td>
</tr>
<tr>
<td>202</td>
<td>Ignoring function, the specified element does not exist</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- UNPACK DATA RECORD (3P)
- INQUIRE ELEMENT TYPE AND SIZE (3P)
- INQUIRE CURRENT ELEMENT CONTENT (3P)
## NAME

INQUIRE ELEMENT POINTER – obtain current element pointer value

## SYNOPSIS

### C Syntax

```c
void
pinq_elem_ptr ( error_ind, ep_value )
Pint *error_ind;  OUT error indicator
Pint *ep_value;  OUT element pointer value
```

### FORTRAN Syntax

```fortran
SUBROUTINE pqep ( ERRIND, EP )
INTEGER ERRIND  OUT error indicator
INTEGER EP  OUT element pointer value
```

## Required PHIGS Operating States

(PHOP, *, STOP, *)

## DESCRIPTION

### Purpose

Use INQUIRE ELEMENT POINTER to determine the current element pointer value.

### C Output Parameters

- `error_ind`
  
  A pointer to the location to store the error number of any error that this function detects.

- `ep_value`
  
  Returns current element pointer value.

### FORTRAN Output Parameters

- `ERRIND`
  
  The error number of any error that this function detects.

- `EP`
  
  Returns current element pointer value.

### ERRORS

- 005 Ignoring function, function requires state (PHOP, *, STOP, *)

## SEE ALSO

- INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
- SET ELEMENT POINTER (3P)
- OFFSET ELEMENT POINTER (3P)
- SET ELEMENT POINTER AT LABEL (3P)
NAME

INQUIRE ELEMENT TYPE AND SIZE – obtain type and size of specified element

SYNOPSIS

C Syntax

```c
void pinq_elem_type_size ( struct_id, element, error_ind, type, size )
Pint struct_id; // structure identifier
Pint element; // element number
Pint *error_ind; // OUT error indicator
Pelem_type *type; // OUT element type
size_t *size; // OUT element size
```

FORTRAN Syntax

```fortran
SUBROUTINE pqets ( STRID, ELENUM, ERRIND, ELTYPE, IL, RL, SL )
INTEGER STRID // structure identifier
INTEGER ELENUM // element number
INTEGER ERRIND // OUT error indicator
INTEGER ELTYPE // OUT element type
INTEGER IL // OUT dimension of integer array (this may be passed to PQCECO as ILL)
INTEGER RL // OUT dimension of real array (this may be passed to PQCECO as IRL)
INTEGER SL // OUT dimension of character array (this may be passed to PQCECO as ISL)
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE ELEMENT TYPE AND SIZE returns the type and size of the specified element in the specified structure.

C Input Parameters

- **struct_id**
  
  Identifier of the structure containing the element for which to return a type and size.

- **element**
  
  Sequence number in the specified structure of the element for which to return a type and size.

C Output Parameters

- **error_ind**
  
  A pointer to the location to store the error number of any error that this function detects.

- **type**
  
  Returns element type. For example, a polyline primitive element would be returned as PELEM_POLYLINE; a character height attribute, as PEL_CHARACTER_HEIGHT; a modelling transformation, as PEL_LOCAL_MODELLING_TRANSFORMATION3; and so forth. The Pelem_type enumerated type is defined in phigs.h as follows:

modified 2 April 1993
typedef enum {
    PELEM_ALL,
    PELEM_NIL,
    PELEM_POLYLINE3,
    PELEM_POLYLINE,
    PELEM_POLYMARKER3,
    PELEM_POLYMARKER,
    PELEM_TEXT3,
    PELEM_TEXT,
    PELEM_ANNO_TEXT_REL3,
    PELEM_ANNO_TEXT_REL,
    PELEM_FILL_AREA3,
    PELEM_FILL_AREA,
    PELEM_FILL_AREA_SET3,
    PELEM_FILL_AREA_SET,
    PELEM_CELL_ARRAY3,
    PELEM_CELL_ARRAY,
    PELEM_GDP3,
    PELEM_GDP,
    PELEM_LINE_IND,
    PELEM_MARKER_IND,
    PELEM_TEXT_IND,
    PELEM_INT_IND,
    PELEM_EDGE_IND,
    PELEM_LINETYPE,
    PELEM_LINEWIDTH,
    PELEM_LINE_COLR_IND,
    PELEM_MARKER_TYPE,
    PELEM_MARKER_SIZE,
    PELEM_MARKER_COLR_IND,
    PELEM_TEXT_FONT,
    PELEM_TEXT_PREC,
    PELEM_CHAR_EXPAN,
    PELEM_CHAR_SPACE,
    PELEM_TEXT_COLR_IND,
    PELEM_CHAR_HT,
    PELEM_CHAR_UP_VEC,
    PELEM_TEXT_PATH,
    PELEM_TEXT_ALIGN,
    PELEM_ANNO_CHAR_HT,
    PELEM_ANNO_CHAR_UP_VEC,
    PELEM_ANNO_PATH,
    PELEM_ANNO_ALIGN,
    PELEM_ANNO_STYLE,
}
PELEM_INT_STYLE,
PELEM_INT_STYLE_IND,
PELEM_INT_COLR_IND,
PELEM_EDGE_FLAG,
PELEM_EDGETYPE,
PELEM_EDGESTYLE,
PELEM_EDGE_WIDTH,
PELEM_EDGE_COLR_IND,
PELEM_PATTERN_SIZE,
PELEM_PATTERN_REF_POINT_VECS,
PELEM_PATTERN_REF_POINT,
PELEM_ADD_NAMES_SET,
PELEM_REMOVE_NAMES_SET,
PELEM_INDIV ASF,
PELEM_HILIGHT_ID,
PELEM_LOCAL_MODEL_TRAN3,
PELEM_LOCAL_MODEL_TRAN,
PELEM_GLOBAL_MODEL_TRAN3,
PELEM_GLOBAL_MODEL_TRAN,
PELEM_MODEL_CLIP_VOL3,
PELEM_MODEL_CLIP_VOL,
PELEM_MODEL_CLIP_IND,
PELEM_RESTORE_MODEL_CLIP_VOL,
PELEM_VIEW_IND,
PELEM_EXEC_STRUCT,
PELEM_LABEL,
PELEM_APPL DATA,
PELEM_GSE,
PELEM_PICK_ID,
PELEM_POLYLINE_SET3_DATA,
PELEM_FILL_AREA_SET3_DATA,
PELEM_TRI_STRIP3_DATA,
PELEM_QUAD_MESH3_DATA,
PELEM_SET_OF_FILL_AREA_SET3_DATA,
PELEM_NUNI_BSP_CURVE,
PELEM_NUNI_BSP_SURF,
PELEM_CELL ARRAY3 PLUS,
PELEM_TEXT_COLR,
PELEM_MARKER_COLR,
PELEM_EDGE_COLR,
PELEM_LINE_COLR,
PELEM_CURVE APPROX CRIT,
PELEM_LINE SHAD METH,
PELEM_INT_COLR,
PELEM_BACK_INT_COLR,
INQUIRE ELEMENT TYPE AND SIZE (3P)

INQUIRE ELEMENT TYPE AND SIZE (3P)

Returns the size, in bytes, that the application will have to allocate in order to call
INQUIRE ELEMENT CONTENT to retrieve the contents of the specified element from
the specified structure. If the element type is such that it is not necessary to
allocate any dynamic memory to retrieve its contents, a value of zero is returned.

FORTRAN Input
Parameters

STRID Identifier of the structure containing the element for which to return a type and
size.

ELENUM Sequence number in the specified structure of the element for which to return a
type and size.

FORTRAN Output
Parameters

ERRIND The error number of any error that this function detects.

ELTYPE Returns the type of the specified element. For example, a polyline primitive
element would be returned as PEPL; a character height attribute, as PECHH; a
local modelling transformation, as PELMT; and so forth. The FORTRAN
definitions for element types are in phigs77.h; a mapping from the six-character
FORTRAN definitions to the actual PHIGS element type names is given below.
These definitions have been alphabetized for ease of use in this listing only.

ADD NAMES TO SET PEADS
ALL PEALL
ANNOTATION STYLE PEANST
ANNOTATION TEXT ALIGNMENT PEATAL
ANNOTATION TEXT CHARACTER HEIGHT PEATCH
ANNOTATION TEXT CHARACTER UP VECTOR PEATCU
ANNOTATION TEXT PATH PEATP

modified 2 April 1993
<table>
<thead>
<tr>
<th>ANNOTATION TEXT RELATIVE</th>
<th>PEARTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANNOTATION TEXT RELATIVE 3</td>
<td>PEARTR3</td>
</tr>
<tr>
<td>APPLICATION DATA</td>
<td>PEAP</td>
</tr>
<tr>
<td>BACK REFLECTANCE PROPERTIES†</td>
<td>PEBAPR</td>
</tr>
<tr>
<td>BACK INTERIOR COLOUR†</td>
<td>PEBIC</td>
</tr>
<tr>
<td>BACK INTERIOR REFLECTANCE EQUATION†</td>
<td>PEBIRE</td>
</tr>
<tr>
<td>BACK INTERIOR SHADING METHOD†</td>
<td>PEBISM</td>
</tr>
<tr>
<td>BACK INTERIOR STYLE†</td>
<td>PEBIS</td>
</tr>
<tr>
<td>BACK INTERIOR STYLE INDEX†</td>
<td>PEBISI</td>
</tr>
<tr>
<td>CELL ARRAY</td>
<td>PECAP</td>
</tr>
<tr>
<td>CELL ARRAY 3</td>
<td>PECAP3</td>
</tr>
<tr>
<td>CELL_ARRAY 3 PLUS†</td>
<td>PECAPA</td>
</tr>
<tr>
<td>CHARACTER EXPANSION FACTOR</td>
<td>PECHXP</td>
</tr>
<tr>
<td>CHARACTER HEIGHT</td>
<td>PECHH</td>
</tr>
<tr>
<td>CHARACTER SPACING</td>
<td>PECHSP</td>
</tr>
<tr>
<td>CHARACTER UP VECTOR</td>
<td>PECHUPI</td>
</tr>
<tr>
<td>COLOUR MAPPING INDEX†</td>
<td>PECAM</td>
</tr>
<tr>
<td>CURVE APPROXIMATION CRITERIA†</td>
<td>PECAC</td>
</tr>
<tr>
<td>DEPTH CUE INDEX†</td>
<td>PECDC</td>
</tr>
<tr>
<td>EDGE COLOUR†</td>
<td>PEEDC</td>
</tr>
<tr>
<td>EDGE COLOUR INDEX</td>
<td>PEEDCI</td>
</tr>
<tr>
<td>EDGE FLAG</td>
<td>PEEDFG</td>
</tr>
<tr>
<td>EDGE INDEX</td>
<td>PEEDI</td>
</tr>
<tr>
<td>EDGETYPE</td>
<td>PEEDT</td>
</tr>
<tr>
<td>EDGEBREADTH SCALE FACTOR</td>
<td>PEEWBS</td>
</tr>
<tr>
<td>EXECUTE STRUCTURE</td>
<td>PEESTS</td>
</tr>
<tr>
<td>FACE CULLING MODE†</td>
<td>PEFCM</td>
</tr>
<tr>
<td>FACE DISTINGUISHING MODE†</td>
<td>PEFDM</td>
</tr>
<tr>
<td>FILL_AREA</td>
<td>PEFA</td>
</tr>
<tr>
<td>FILL AREA 3</td>
<td>PEFA3</td>
</tr>
<tr>
<td>FILL AREA SET</td>
<td>PEFAS</td>
</tr>
<tr>
<td>FILL AREA SET 3</td>
<td>PEFA3S</td>
</tr>
<tr>
<td>GDP</td>
<td>PEGDP</td>
</tr>
<tr>
<td>GDP 3</td>
<td>PEGDP3</td>
</tr>
<tr>
<td>GLOBAL MODELLING TRANSFORMATION</td>
<td>PEGMT</td>
</tr>
<tr>
<td>GLOBAL MODELLING TRANSFORMATION 3</td>
<td>PEGMT3</td>
</tr>
<tr>
<td>GSE</td>
<td>PEGSE</td>
</tr>
<tr>
<td>HLHSR_IDENTIFIER</td>
<td>PEHRID</td>
</tr>
<tr>
<td>INDIVIDUAL ASF</td>
<td>PEIASF</td>
</tr>
<tr>
<td>INTERIOR COLOUR†</td>
<td>PEIC</td>
</tr>
<tr>
<td>INTERIOR COLOUR INDEX</td>
<td>PEICI</td>
</tr>
<tr>
<td>INTERIOR INDEX</td>
<td>PEII</td>
</tr>
<tr>
<td>INTERIOR SHADING METHOD†</td>
<td>PEISM</td>
</tr>
<tr>
<td>INTERIOR STYLE</td>
<td>PEIS</td>
</tr>
</tbody>
</table>

modified 2 April 1993

571
<table>
<thead>
<tr>
<th>INQUIRE ELEMENT TYPE AND SIZE (3P)</th>
<th>SunPHIGS Release 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERIOR STYLE INDEX</td>
<td>PEISI</td>
</tr>
<tr>
<td>LABEL</td>
<td>PELB</td>
</tr>
<tr>
<td>LIGHT SOURCE STATE†</td>
<td>PELSS</td>
</tr>
<tr>
<td>LINETYPE</td>
<td>PELN</td>
</tr>
<tr>
<td>LINEWIDTH SCALE FACTOR</td>
<td>PELWSC</td>
</tr>
<tr>
<td>LOCAL MODELLING TRANSFORMATION</td>
<td>PELMT</td>
</tr>
<tr>
<td>LOCAL MODELLING TRANSFORMATION 3</td>
<td>PELMT3</td>
</tr>
<tr>
<td>MARKER COLOUR INDEX</td>
<td>PEPMC1</td>
</tr>
<tr>
<td>MARKER SIZE SCALE FACTOR</td>
<td>PEMKSC</td>
</tr>
<tr>
<td>MARKER TYPE</td>
<td>PEMK</td>
</tr>
<tr>
<td>MODELLING CLIPPING INDICATOR</td>
<td>PEMCLI</td>
</tr>
<tr>
<td>MODELLING CLIPPING VOLUME</td>
<td>PEMCV</td>
</tr>
<tr>
<td>MODELLING CLIPPING VOLUME 3</td>
<td>PEMCV3</td>
</tr>
<tr>
<td>NIL</td>
<td>PENIL</td>
</tr>
<tr>
<td>NON-UNIFORM B-SPLINE CURVE†</td>
<td>PENBSC</td>
</tr>
<tr>
<td>NON-UNIFORM B-SPLINE SURFACE†</td>
<td>PENBSS</td>
</tr>
<tr>
<td>PARAMETRIC SURFACE CHARACTERISTICS</td>
<td>PESPSC</td>
</tr>
<tr>
<td>PATTERN REFERENCE POINT</td>
<td>PEPAF</td>
</tr>
<tr>
<td>PATTERN REFERENCE POINT AND VECTORS</td>
<td>PEPRPV</td>
</tr>
<tr>
<td>PATTERN SIZE</td>
<td>PEPA</td>
</tr>
<tr>
<td>PICK ID</td>
<td>PEKPID</td>
</tr>
<tr>
<td>POLYLINE</td>
<td>PEPL</td>
</tr>
<tr>
<td>POLYLINE 3</td>
<td>PEPL3</td>
</tr>
<tr>
<td>POLYLINE COLOUR†</td>
<td>PEPLC</td>
</tr>
<tr>
<td>POLYLINE COLOUR INDEX</td>
<td>PEPLCI</td>
</tr>
<tr>
<td>POLYLINE INDEX</td>
<td>PEPLI</td>
</tr>
<tr>
<td>POLYLINE SET 3 WITH DATA†</td>
<td>PEPSD3</td>
</tr>
<tr>
<td>POLYLINE SHADING METHOD†</td>
<td>PEPLSM</td>
</tr>
<tr>
<td>POLYMARKER</td>
<td>PEMP</td>
</tr>
<tr>
<td>POLYMARKER 3</td>
<td>PEM3</td>
</tr>
<tr>
<td>POLYMARKER COLOUR†</td>
<td>PEMC</td>
</tr>
<tr>
<td>POLYMARKER INDEX</td>
<td>PEMPi</td>
</tr>
<tr>
<td>QUADRILATERAL MESH 3 WITH DATA†</td>
<td>PEQMD3</td>
</tr>
<tr>
<td>REFLECTANCE EQUATION†</td>
<td>PEIRE</td>
</tr>
<tr>
<td>REFLECTANCE PROPERTIES†</td>
<td>PEAPR</td>
</tr>
<tr>
<td>REMOVE NAMES FROM SET</td>
<td>PERES</td>
</tr>
<tr>
<td>RENDERING COLOUR MODEL†</td>
<td>PERCM</td>
</tr>
<tr>
<td>RESTORE MODELLING CLIPPING VOLUME</td>
<td>PERMCV</td>
</tr>
<tr>
<td>SET OF FILL AREA SET 3 WITH DATA†</td>
<td>PEFS3</td>
</tr>
<tr>
<td>SURFACE APPROXIMATION CRITERIA†</td>
<td>PESAC</td>
</tr>
<tr>
<td>TEXT</td>
<td>PETX</td>
</tr>
<tr>
<td>TEXT 3</td>
<td>PETX3</td>
</tr>
<tr>
<td>TEXT ALIGNMENT</td>
<td>PETXAL</td>
</tr>
<tr>
<td>TEXT COLOUR†</td>
<td>PETXC</td>
</tr>
</tbody>
</table>

modified 2 April 1993
IL Returns the dimension required for the integer array argument to INQUIRE
ELEMENT CONTENT in order to retrieve the specified element contents from the
specified structure.

RL Returns the dimension required for the real array argument to INQUIRE
ELEMENT CONTENT in order to retrieve the specified element contents from the
specified structure.

SL Returns the dimension required for the character array argument to INQUIRE
ELEMENT CONTENT in order to retrieve the specified element contents from the
specified structure.

ERRORS

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

201 Ignoring function, the specified structure does not exist

202 Ignoring function, the specified element does not exist

SEE ALSO

INQUIRE ELEMENT CONTENT (3P)
INQUIRE ELEMENT POINTER (3P)
ELEMENT SEARCH (3P)
INQUIRE ERROR HANDLING MODE – obtain current error handling mode

C Syntax

```c
void pinq_err_hand_mode ( error_ind, mode )
Pint *error_ind;  OUT error indicator
Perr_mode *mode;  OUT error mode
```

FORTRAN Syntax

```fortran
SUBROUTINE pqerhm ( ERRIND, ERHM )
INTEGER ERRIND  OUT error indicator
INTEGER ERHM    OUT error handling mode (POFF, PON)
```

Required PHIGS Operating States

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use **INQUIRE ERROR HANDLING MODE** to determine the current error handling mode. Error handling may be ON or OFF.

See the description of the subroutine **SET ERROR HANDLING MODE** for more information.

**C Output Parameters**

- `error_ind`: A pointer to the location to store the error number of any error that this function detects.
- `mode`: A pointer to a Perr_mode enumerated type, in which the system returns the current error handling mode. Values for Perr_mode are defined in phigs.h as follows:
  - PERR_OFF
  - PERR_ON

**FORTRAN Output Parameters**

- `ERRIND`: The error number of any error that this function detects.
- `ERHM`: The current error handling mode, defined in phigs77.h as follows:
  - POFF
  - PON

**ERRORS**

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

**SEE ALSO**

ESCAPE -1 (3P)
ERROR HANDLING (3P)
ERROR LOGGING (3P)

modified 2 April 1993
**NAME**

INQUIRE GENERALIZED DRAWING PRIMITIVE – obtain generalized drawing primitive (GDP) attributes

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqgdp ( WTYPE, GDP, ERRIND, NBND, BNDL )
INTEGER WTYPE  /* workstation type */
INTEGER GDP    /* GDP identifer */
INTEGER ERRIND /* OUT error indicator */
INTEGER NBND   /* OUT number of sets of attributes used */
INTEGER BNDL(5) /* OUT list of sets of attributes used */
```

**Required PHIGS Operating States**

( PHOP, *, *, * )

**DESCRIPTION**

**Purpose**

INQUIRE GENERALIZED DRAWING PRIMITIVE determines the sets of attributes used by the specified generalized drawing primitive. The representation of the generalized drawing primitive at a given workstation will be controlled by zero or more sets of polyline, polymarker, text, interior, and edge attributes, as indicated by this inquiry function. Whether bundle indices or the corresponding individual attributes are used will depend on the values of the appropriate aspect source flags at traversal time.

**C Input Parameters**

- **ws_type** Type of workstation for which to inquire the generalized drawing primitive attributes.
- **gdp** The identifier of the generalized drawing primitive for which to inquire attributes.

**C Output Parameters**

- **error_ind** A pointer to the location to store the error number of any error that this function detects.
- **num_attrs** A pointer to an integer that indicates how many attributes are listed in `attrs`.
- **attrs** A Pattrs structure indicating which set of attributes are used by the specified generalized drawing primitive on workstations of type `type`. Pattrs is an enumerated type defined in phigs.h as follows:

```c
enum Pattrs {
    POLYLINE, POLYMARKER, TEXT, INTERIOR, EDGE
};
```

modified 2 April 1993

575
typedef enum {
    PATTR_LINE, /* polyline attributes */
    PATTR_MARKER, /* marker attributes */
    PATTR_TEXT, /* text attributes */
    PATTR_INT, /* interior attributes */
    PATTR_EDGE /* edge attributes */
} Pattrs;

FORTRAN Input
Parameters

WTYPE Type of workstation for which to inquire the generalized
drawing primitive attributes.

GDP The identifier of the generalized drawing primitive for which to inquire
attributes.

FORTRAN Output
Parameters

ERRIND The error number of any error that this function detects.

NBND An integer in which the number of sets of attributes used by the specified
generalized drawing primitive on workstations of type WTYPE is returned.

BNDL An integer array in which num_atts values are returned indicating which of
five possible sets of attributes are used by the specified generalized drawing
primitive on workstations of type WTYPE. Note that the application must
specify an array that is large enough for five values (the maximum number
that could be returned).

The following values may be returned in BNDL to indicate the attribute sets
used:

- PPLATT Polyline attributes
- PPMATT Polymarker attributes
- PTXATT Text attributes
- PINATT Interior attributes
- PEDATT Edge attributes

ERRORS

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

051 Ignoring function, this information is not yet available for this generic
workstation type; open a workstation of this type and use the specific
workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that
is, the workstation category is not OUTPUT, OUTIN, or MO)

064 Ignoring function, the specified workstation type is not able to generate the
specified generalized drawing primitive

062 Ignoring function, this information is not available for this MO workstation type

modified 2 April 1993
SEE ALSO

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
GENERALIZED DRAWING PRIMITIVE (3P)
INQUIRE GENERALIZED DRAWING PRIMITIVE 3 (3P)
### NAME
INQUIRE GENERALIZED DRAWING PRIMITIVE 3 – obtain generalized drawing primitive 3 (GDP3) attributes

### SYNOPSIS

#### C Syntax
```c
void pinq_gdp3 ( ws_type, gdp, error_ind, num_attrs, attrs )
Pint ws_type;  // workstation type
Pint gdp;      // 3D GDP function number
Pint *error_ind;  // OUT error indicator
Pint *num_attrs;  // OUT number of attributes
Pattrs attrs[5];  // OUT list of attributes
```

#### FORTRAN Syntax
```fortran
SUBROUTINE pqgdp3 ( WTYPE, GDP, ERRIND, NBND, BNDL )
INTEGER WTYPE  // workstation type
INTEGER GDP    // 3D GDP identifier
INTEGER ERRIND // OUT error indicator
INTEGER NBND   // OUT number of sets of attributes used
INTEGER BNDL(5) // OUT list of sets of attributes used
```

### FORTRAN Operating States
(PHOP, *, *, *)

### DESCRIPTION

#### Purpose
Use INQUIRE GENERALIZED DRAWING PRIMITIVE 3 to determine the sets of attributes used by the specified 3D generalized drawing primitive. The representation of the generalized drawing primitive at a given workstation will be controlled by zero or more sets of polyline, polymarker, text, interior, and edge attributes, as indicated by this inquiry function. Whether bundle indices or the corresponding individual attributes are used will depend on the values of the appropriate aspect source flags at traversal time.

#### C Input Parameters
- **ws_type**  
  Type of workstation for which to inquire the generalized drawing primitive 3 attributes.

- **gdp**  
  The identifier of the generalized drawing primitive 3 for which to inquire attributes.

#### C Output Parameters
- **error_ind**  
  A pointer to the location to store the error number of any error that this function detects.

- **num_attrs**  
  A pointer to an integer which returns the number of attributes listed in `attrs`.

- **attrs**  
  An array of five `Pattrs`. A `Pattrs` structure in which `num_attrs` values are returned indicating which set of attributes are used by the specified generalized drawing primitive on workstations of type `ws_type`. `Pattrs` is an enumerated type defined

---

578 modified 2 April 1993
in phigs.h as follows:

typedef enum {
    PATTR_LINE,  /* polyline attributes */
    PATTR_MARKER, /* polymarker attributes */
    PATTR_TEXT,   /* text attributes */
    PATTR_INT,    /* interior attributes */
    PATTR_EDGE    /* edge attributes */
} Pattrs;

**FORTRAN Input Parameters**

- **WTYPE** Type of workstation for which to inquire the generalized drawing primitive 3 attributes.
- **GDP** The identifier of the generalized drawing primitive 3 for which to inquire attributes.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **NBND** An integer in which the number of sets of attributes used by the specified generalized drawing primitive 3 on workstations of type WTYPE is returned.
- **BNDL** An integer array in which num_atts values are returned indicating which of five possible sets of attributes are used by the specified generalized drawing primitive 3 on workstations of type WTYPE. Note that the application must specify an array that is large enough for five values (the maximum number that could be returned).

The following values may be returned in BNDL to indicate the attribute sets used:

- **PPLATT** Polyline attributes
- **PPMATT** Polymarker attributes
- **PTXATT** Text attributes
- **PINATT** Interior attributes
- **PEDATT** Edge attributes

**ERRORS**

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 052 Ignoring function, workstation type not recognized by the implementation
- 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- 062 Ignoring function, this information is not available for this MO workstation type
- 064 Ignoring function, the specified workstation type is not able to generate the specified generalized drawing primitive

modified 2 April 1993
SEE ALSO

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
GENERALIZED DRAWING PRIMITIVE 3 (3P)
INQUIRE GENERALIZED DRAWING PRIMITIVE (3P)
NAME  
INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES – obtain list of generalized structure element (GSE) facilities

SYNOPSIS  
C Syntax

```c
void pinq_gse_facs ( length, start, error_ind, gse, total_length )

Pint length;  // length of application list
Pint start;   // starting position
Pint *error_ind;  // OUT error indicator
Pgse_id_dep_list *gse;  // OUT list of GSE ids and their dependencies
Pint *total_length;  // OUT length of list in PHIGS
```

FORTRAN Syntax

```fortran
SUBROUTINE pqgsef ( N, ERRIND, OL, GSEID, WSDIND )

INTEGER N  // element of the list of available GSEs
INTEGER ERRIND  // OUT error indicator
INTEGER OL  // OUT number of available GSEs
INTEGER GSEID  // OUT GSE identifier of Nth element of the list of available GSEs
INTEGER WSDIND  // OUT workstation dependency indicator of Nth element of the list of available GSEs (PWKI, PWKD)
```

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION  
Purpose

Use INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES to obtain a list of the GSE identifiers supported, and whether each is workstation-independent (that is, supported on all workstations) or workstation-dependent (that is, supported on some, but not all workstations).

C Input Parameters

- `length`: The number of elements for which memory is allocated in the output parameter `gse`. 0 can be specified, in order to get the `total length` of the list.
- `start`: Starting position in the list at which to begin the inquiry.

C Output Parameters

- `error_ind`: A pointer to the location to store the error number of any error that this function detects.
- `gse`: A pointer to a Pgse_id_dep_list in which the system returns the portion of the list of GSE identifiers from the PHIGS description table, starting with `start`. Pgse_id_dep_list is defined in phigs.h as follows:

modified 2 April 1993
typedef struct {
    Pint num_id_facs; /* number of identifiers/dependency element */
    Pgse_id_dep *id_facs; /* list of GSE facilities */
} Pgse_id_dep_list;

The pointer gse->id_facs must be initialized to an array of length Pgse_id_dep elements, defined as:

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

Pws_dep_ind is an enumerated type, with values:

    PWS_INDEP    Workstation independent
    PWS_DEP     Workstation dependent

**total_length**
A pointer to an integer in which to return the total length of the list. This is the value required for length if all the items in the list are to be returned.

**FORTRAN Input Parameters**

- $N$ Get the $N$th element from the list.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **OL** The total length of the list of available GSEs.
- **GSEID** The GSE identifier of the $N$th element in the list of available GSEs.
- **WSDIND** The workstation dependency indicator of the $N$th element in the list of available GSEs, one of:
  - PWKI  Workstation independent
  - PWKD  Workstation dependent

**ERRORS**

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 021 Ignoring function, this information is unavailable for this workstation type
- 022 Ignoring function, workstation type is not recognized by the implementation
- 023 Ignoring function, specified workstation type does not exist
- 039 Ignoring function, specified workstation category is not OUTPUT or OUTIN
- 2201 C: Start index is out of range

**SEE ALSO**

- GENERALIZED STRUCTURE ELEMENT (3P)
- INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS (3P)
- PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NAME

INQUIRE HIGHLIGHTING FILTER – inquire inclusion and exclusion name sets for a workstation's highlighting filter

SYNOPSIS

C Syntax

```c
void ping_highl_filter ( ws, store, error_ind, highl_filter )
```

- `Pint ws;` workstation identifier
- `Pstore store;` handle to Store object
- `Pint *error_ind;` OUT error indicator
- `Pfilter **highl_filter;` OUT highlighting filter

FORTRAN Syntax

```fortran
SUBROUTINE pqhlft ( WKID, ISBSZ, ESBSZ, ERRIND, ISN, IS, ESN, ES )
```

- `INTEGER WKID` workstation identifier
- `INTEGER ISBSZ` inclusion set buffer size
- `INTEGER ESBSZ` exclusion set buffer size
- `INTEGER ERRIND` OUT error indicator
- `INTEGER ISN` OUT number of names in the inclusion set
- `INTEGER IS(ISBSZ)` OUT inclusion set
- `INTEGER ESN` OUT number of names in the exclusion set
- `INTEGER ES(ESBSZ)` OUT exclusion set

Required PHIGS Operating States

`(PHOP, WSOP, *, *)`

DESCRIPTION

Purpose

Use INQUIRE HIGHLIGHTING FILTER to obtain the highlighting filter from a specified workstation's state list. This is the filter which is compared to the traversal-time current name set of each primitive to determine if the primitive is highlighted.

The filter contains an inclusion set and an exclusion set of names. During traversal, a primitive is eligible for highlighting if at least one name in the current name set in the inclusion set and no name in the current name set in the exclusion set. Each name in the name set, inclusion set, and exclusion set is a small positive integer.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by `highl_filter`.

- `ws` The workstation identifier of the workstation whose highlighting filter is to be returned.

modified 2 April 1993 583
INQUIRE HIGHLIGHTING FILTER (3P)

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind A pointer to the location to store the error number of any error that this function detects.

highl_filter A pointer to a pointer to a list of highlighting filters. PfILTER is defined in phigs.h as follows:

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

Pint_list is defined in phigs.h as follows:

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

FORTRAN Input Parameters

WKID The workstation identifier of the workstation whose state list is queried.

ISBSZ Size of the IS array in which the returned inclusion set elements will be stored. If this value is smaller than the actual number of elements in the inclusion set (ISN), no data will be returned in the IS array, but ISN will be set to indicate the array size required. If you call this function with an array size of zero, ISN is returned with the required array size. Error 2001 is returned if ISBSZ is too small, but not if it's zero.

ESBSZ Size of the ES array in which the returned exclusion set elements will be stored. If this value is smaller than the actual number of elements in the exclusion set (ESN), no data will be returned in the ES array, but ESN will be set to indicate the array size required. If you call this function with an array size of zero, ESN is returned with the required array size. Error 2001 is returned if ESBSZ is too small, but not if it's zero.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

ISN The number of inclusion set elements returned in the IS array.

IS An array of integers in which the elements of the filter's inclusion set are returned.

ESN The number of exclusion set elements returned in the ES array.

ES An array of integers in which the elements of the filter's exclusion set are returned.
## ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state *(PHOP, WSOP, *, <em>)</em></td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
</tbody>
</table>

## SEE ALSO

- SET HIGHLIGHTING FILTER *(3P)*
NAME
INQUIRE HLHSR IDENTIFIER FACILITIES – inquire for available hidden line and hidden surface removal identifiers of specified workstation type

SYNOPSIS
C Syntax
void pinq_hlhsr_id_facs ( type, length, start, error_ind, ids, length_list )
Pint type; 
   workstation type
Pint length; 
   length of id list
Pint start; 
   starting position of id list
Pint *error_ind; 
   OUT error indicator
Pint_list *ids; 
   OUT list of HLHSR identifiers
Pint *length_list; 
   OUT length of id list in PHIGS

FORTRAN Syntax
SUBROUTINE pqhrif ( WTYPE, NI, ERRIND, NHRID, HRID, NHRMD, HRMD )
INTEGER WTYPE 
   workstation type
INTEGER NI 
   sequence number of HLHSR identifier list element requested
INTEGER ERRIND 
   OUT error indicator
INTEGER NHRID 
   OUT number of available HLHSR identifiers
INTEGER HRID 
   OUT NITH element of list of available HLHSR identifiers

Required PHIGS Operating States (PHOP, *, *, *)

DESCRIPTION
Purpose
INQUIRE HLHSR IDENTIFIER FACILITIES obtains the available HLHSR (hidden line and hidden surface removal) identifiers on the specified workstation type. See SET HLHSR IDENTIFIER and SET HLHSR MODE for a description of how to enable hidden surface removal and set the id.

C Input Parameters
type Type of workstation.
length The number of ints in the ids output parameter for which the application has allocated memory. length is the number of list elements that the system can return in ids→ints. If a value of 0 is used here, no data will be returned in the ids→ints list, but the total number of identifiers in the PHIGS state list will be returned in length_list.
start Starting position of inquiry into the PHIGS state list of current identifiers. The elements of the list of identifiers, beginning with the item number specified by start, are copied sequentially into ids→ints until ids→ints is full or all the identifiers have been copied.

C Output Parameters
error_ind A pointer to the location to store the error number of any error that this function detects.

modified 2 April 1993
ids
A pointer to a Pint_list data structure in which the system returns a list of the HLHSR identifiers available on this workstation type. The returned list starts with the start item in the list of identifiers and returns the next length items. Pint_list is defined in phigs.h as follows:

typedef struct {
    Pint num_ints; /* number of identifiers in list */
    Pint *ints; /* list of identifiers */
} Pint_list;

Prior to calling this function, the ints field must contain a pointer to an application supplied buffer. This buffer must be able to hold length ids. num_ints is the number of ids returned in the buffer.

Defined ids are:

0 PHIGS_HLHSR_ID_OFF Do not do Hidden Line or Hidden Surface Removal
1 PHIGS_HLHSR_ID_ON Do Hidden Line and Hidden Surface Removal according to the current HLHSR mode

length_list
A pointer to an integer in which the system returns the total number of elements in the PHIGS state list of currently-used identifiers. This is the value required for length if all identifiers are to be returned.

FORTRAN Input Parameters
WTYP E Get the HLHSR facilities for this workstation type.
NI The number of entry desired from the list of HLHSR identifiers.

FORTRAN Output Parameters
ERRIND The error number of any error that this function detects.

NHRID The total number of items in the workstation list of HLHSR identifiers.
HRID The Nth element of the list of HLHSR identifiers; one of

0 PHIGSHLHSRIDOFF Do not do Hidden Line or Hidden Surface Removal
1 PHIGSHLHSRIDON Do Hidden Line and Hidden Surface Removal according to the current HLHSR Mode

ERRORS
002 Ignoring function, function requires state (PHOP, *, *, *)
052 Ignoring function, workstation type not recognized by the implementation
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
057 Ignoring function, specified workstation is of category MI
062 Ignoring function, this information is not available for this MO workstation type

modified 2 April 1993
SEE ALSO

- INQUIRE WORKSTATION CONNECTION AND TYPE (3P)
- SET HLHSR MODE (3P)
- SET HLHSR IDENTIFIER (3P)
NAME
INQUIRE HLHSR MODE FACILITIES – inquire for hidden line and hidden surface removal modes of specified workstation type

SYNOPSIS
C Syntax
void
pinq_hlhsr_mode_facs ( type, length, start, error_ind, modes, length_list )
Pint type;  // workstation type
Pint length;  // length of mode list
Pint start;  // starting position of mode list
Pint *error_ind;  // OUT error indicator
Pint_list *modes;  // OUT list of HLHSR modes
Pint *length_list;  // OUT length of mode list in PHIGS

FORTRAN Syntax
SUBROUTINE pqhrmf ( WTYPE, NM, ERRIND, NHRMD, HRMD )
INTEGER WTYPE  // workstation type
INTEGER NM  // sequence number of HLHSR mode list element requested
INTEGER ERRIND  // OUT error indicator
INTEGER NHRMD  // OUT number of available HLHSR modes
INTEGER HRMD  // OUT NMth element of list of available HLHSR modes

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
INQUIRE HLHSR FACILITIES obtains the available HLHSR (hidden line and hidden surface removal) modes on the specified workstation type. See SET HLHSR IDENTIFIER and SET HLHSR MODE for a description of how to enable hidden surface removal and select the mode.

C Input Parameters
type  // Type of workstation.
length  // The number of ints in the modes output parameter for which the application has allocated memory. length is the number of list elements that the system can return in modes→ints. If a value of 0 is used here, no data is returned in the modes→ints list, but the total number of modes in the PHIGS state list is returned in length_list.
start  // Starting position of inquiry into the PHIGS state list of current mode. The elements of the list of mode, beginning with the item number specified by start, are copied sequentially into mode→ints until modes→ints is full or all the modes have been copied.
C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error that this function detects.
- **modes**
  A pointer to a Pint_list data structure in which the system returns a list of the HLHSR modes available on this workstation type. The returned list starts with the start item in the list of modes and returns the next length items. Pint_list is defined in phigs.h as follows:
  ```
  typedef struct {
      Pint num_ints; /* number of modes in list */
      Pint *ints;   /* list of modes */
  } Pint_list;
  ```
  Possible modes are:
  - 0 PHIGS_HLHSR_MODE_NONE Disable HLHSR
  - 1 PHIGS_HLHSR_MODE_ZBUFF Enable HLHSR using Z-buffer
  - 5 PHIGS_HLHSR_MODE_ZBUFF_NO_ID Enable Z-buffering without HLHSR IDS

- **length_list**
  A pointer to an integer in which the system returns the total number of elements in the PHIGS state list of currently used modes. This is the value required for length if all modes are to be returned.

FORTRAN Input Parameters

- **WTYPE**
  Get the HLHSR facilities for this workstation type.
- **NM**
  The number of entry desired from the list of HLHSR modes.

FORTRAN Output Parameters

- **ERRIND**
  The error number of any error that this function detects.
- **NHRMD**
  The total number of elements in the list of HLHSR modes.
- **HRMD**
  The NMth element of the list of HLHSR modes; one of
  - 0 PHIGS_HLHSRMDNONE Disable HLHSR
  - 1 PHIGS_HLHSRMDZBUFF Enable HLHSR using Z-buffer
  - 5 PHIGS_HLHSRMDZBUFF_NO_ID Enable Z-buffering without HLHSR IDS

ERRORS

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 052 Ignoring function, workstation type not recognized by the implementation
- 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- 057 Ignoring function, specified workstation is of category MI
- 062 Ignoring function, this information is not available for this MO workstation type

modified 2 April 1993
SEE ALSO

INQUIRE WORKSTATION CONNECTION AND TYPE (3P)
INQUIRE HLHSR IDENTIFIER FACILITIES (3P)
INQUIRE HLHSR MODE (3P)
SET HLHSR MODE (3P)
SET HLHSR IDENTIFIER (3P)

modified 2 April 1993
NAME
INQUIRE HLHSR MODE – obtain workstation’s hidden line and hidden surface removal (HLHSR) mode

SYNOPSIS
C Syntax
void pinq_hlhsr_mode ( ws, error_ind, state, cur_mode, req_mode )

Pint ws; workstation identifier
Pint *error_ind; OUT error indicator
Pupd_st *state; OUT HLHSR update state
Pint *cur_mode; OUT current HLHSR mode
Pint *req_mode; OUT requested HLHSR mode

FORTRAN Syntax
SUBROUTINE pqhrm ( WKID, ERRIND, HUPD, CHRM, RHRM )
INTEGER WKID workstation identifier
INTEGER ERRIND OUT error indicator
INTEGER HUPD OUT HLHSR mode update state (PNPEND, PPEND)
INTEGER CHRM OUT current HLHSR mode
INTEGER RHRM OUT requested HLHSR mode

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE HLHSR MODE to determine the current HLHSR (hidden line and hidden surface removal) mode from a specified workstation’s state list.

C Input Parameters
ws The workstation identifier of the workstation whose workstation state list is queried.

C Output Parameters
error_ind A pointer to the location to store the error number of any error that this function detects.
state A pointer to a location in which the system returns the HLHSR update state. Pupd_st is one of the values PUPD_NOT_PENDING (Not Pending) or PUPD_PENDING (Pending).
cur_mode A pointer to a location in which the system returns the current HLHSR mode.
req_mode A pointer to a location in which the system returns the requested HLHSR mode. The requested mode can differ from the current mode if the update state is PENDING.

592 modified 2 April 1993
FORTRAN Input Parameters

| WKID | The workstation identifier of the workstation whose state list is queried. |

FORTRAN Output Parameters

| ERRIND | The error number of any error that this function detects. |
| HUPD | The HLHSR update state; one of the values PNPEND (Not Pending) or PPEND (Pending). |
| CHRM | The current HLHSR mode. |
| RHRM | The requested HLHSR mode. The requested mode can differ from the current mode if the update state is PENDING. |

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
057 Ignoring function, specified workstation is of category MI

SEE ALSO

SET HLHSR MODE (3P)
SET HLHSR IDENTIFIER (3P)
INQUIRE HLHSR IDENTIFIER FACILITIES (3P)
INQUIRE HLHSR MODE FACILITIES (3P)
NAME
INQUIRE INPUT QUEUE OVERFLOW – obtain input queue overflow state of PHIGS error state list

SYNOPSIS
C Syntax
void
pinq_in_overf ( error_ind, ws, class, dev )

Pint *error_ind; \textit{OUT} error indicator
Pint *ws; \textit{OUT} workstation identifier
Pin_class *class; \textit{OUT} input class
Pint *dev; \textit{OUT} input device number

FORTRAN Syntax
SUBROUTINE pqiqov ( ERRIND, WKID, ICL, IDN )

INTEGER ERRIND \textit{OUT} error indicator
INTEGER WKID \textit{OUT} workstation identifier
INTEGER ICL \textit{OUT} input class
INTEGER IDN \textit{OUT} input device number

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use \textsc{inquire input queue overflow} to retrieve \textit{input queue overflow} information from the PHIGS error state list.

C Output Parameters
\textit{error\_ind}
A pointer to the location to store the error number of any error that this function detects.

\textit{ws}
A pointer to a Pint in which to store the workstation identifier of the workstation associated with the input device that caused the overflow.

\textit{class}
A pointer to a Pin_class structure in which to store the class of the input device that caused the overflow.

\textit{Pin\_class} is an enumerated type that may assume the following values:
- PIN\_NONE
- PIN\_LOC
- PIN\_STROKE
- PIN\_VAL
- PIN\_CHOICE
- PIN\_PICK
- PIN\_STRING

\textit{dev}
A pointer to a Pint in which to store the number of the input device that overflowed the input queue.

594 modified 2 April 1993
FORTRAN Output Parameters

**ERRIND**  The error number of any error that this function detects.

**WKID**  The workstation identifier of the workstation associated with the input device that caused the overflow.

**ICL**  The class of the input device that caused the overflow. Valid classes as defined in phigs77.h are:

- **PNCLAS**  None
- **PLOCAT**  Locator
- **PSTROK**  Stroke
- **PVALUA**  Valuator
- **PCHOIC**  Choice
- **PPICK**  Pick
- **PSTRIN**  String

**IDN**  The number of the input device that overflowed the input queue.

Execution

When the input queue overflows, the identification of one of the logical input devices that caused the overflow is placed in the error state list. This information remains in the error state list until:

- The workstation associated with the input device is closed.
- **PHIGS** is closed.
- This function is called.

Calling this function clears the input queue overflow information in the error state list. If the input queue has not overflowed since this function or **OPEN PHIGS** was last called, the appropriate error will be returned in error indicator.

**Note:** When the input queue overflows, no more events can be added to it until **AWAIT EVENT** is called with the queue empty. That is, all events must be removed from the queue and **AWAIT EVENT** called while the queue is empty before any new events can be added.

**ERRORS**

- **003**  Ignoring function, function requires state (**PHOP**, **WSOP**, *, *)
- **257**  Ignoring function, input queue has not overflowed
- **258**  Warning, input queue has overflowed, but associated workstation has been closed

**SEE ALSO**

- **AWAIT EVENT** (3P)
- **FLUSH DEVICE EVENTS** (3P)
NAME
INQUIRE INTERIOR FACILITIES – obtain list of workstation interior facilities from workstation description table

SYNOPSIS
C Syntax

void
pinq_int_facs ( type, h_len, h_st, error_ind, facil, tot_h_len )

Pint type;  // workstation type
Pint h_len;  // length of hatch style list
Pint h_st;  // starting position
Pint *error_ind;  // OUT error indicator
Pint_facs *facil;  // OUT interior facilities
Pint *tot_h_len;  // OUT len of hatch list in PHIGS

FORTRAN Syntax

SUBROUTINE pqif ( WTYPE, NI, NH, ERRIND, NIS, IS, NHS, HS, NPFAI )

INTEGER WTYPE  // workstation type
INTEGER NI  // list element of interior styles requested
INTEGER NH  // list element of hatch styles requested
INTEGER ERRIND  // OUT error indicator
INTEGER NIS  // OUT number of available interior styles
INTEGER IS  // OUT NIHth element of list of available interior styles
INTEGER NHS  // OUT number of available hatch styles
INTEGER HS  // OUT NIHth element of list of available hatch style indices
INTEGER NPFAI  // OUT number of predefined interior indices

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
INQUIRE INTERIOR FACILITIES obtains a list of the available interior styles and hatch styles for the specified type of workstation.

C Input Parameters

- **type** Get the interior facilities for this **workstation type**.
- **h_len** The number of elements for which memory is allocated in the output parameter facil→hatch_styles.ints. 0 may be specified in order to get the **tot_h_len**.
- **h_st** Starting position in the list of hatch styles at which to begin the inquiry.

C Output Parameters

- **error_ind** A pointer to the location to store the error number of any error that this function detects.
- **facil** A pointer to an allocated data structure in which the system returns a portion of the list of available hatch styles, starting with **h_st**. The structure is defined as follows:
  
  typedef struct {

  modified 2 April 1993
SunPHIGS Release 3.0

INQUIRE INTERIOR FACILITIES (3P)

Pint num_int_styles; /* number of interior styles */
Pint_style *int_styles[5]; /* list of available interior styles */
Pint_list hatch_styles; /* list of available hatch styles */
Pint num_pred_inds; /* number of predefined interior bundles */

Pint_style is an enumerated type for the following interior styles; one of:

PSTYLE_HOLLOW Hollow
PSTYLE_SOLID Solid
PSTYLE_PAT Patterned
PSTYLE_HATCH Hatched
PSTYLE_EMPTY Empty

Pint_list is defined in phigs.h as follows:

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

Prior to calling this function, the ints field of the Pint_list structure must contain a pointer to an application supplied buffer. This buffer must be at least as large as the h_len parameter.

tot_h_len
A pointer to an integer in which the system returns the total number of items in the workstation list of hatch styles. This is the value required by h_len if all the items in the list are to be returned.

FORTRAN Input Parameters

WTYPE Get the interior facilities for this workstation type.

NI The number of entry desired from the list of interior styles.

NH The number of entry desired from the list of hatch styles.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

NIS The total number of items in the workstation list of interior styles.

IS The Nth element of the list of interior styles; one of

PHOLLO Hollow
PSOLID Solid
PPATTR Patterned
PHATCH Hatched
PISEMP Empty
PGENER† General

† This is a SunPHIGS Extension that is based on PHIGS PLUS and is not part of the PHIGS standard.

NHS The total number of elements in the list of hatch styles.

modified 2 April 1993
**HS**  The NHth element of the list of hatch styles.

**NPFAI**  The number of predefined interior bundles.

**ERRORS**

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

051  Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type

052  Ignoring function, workstation type not recognized by the implementation

059  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062  Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

PHIGS WORKSTATION DESCRIPTION TABLE (7P)

INQUIRE PREDEFINED INTERIOR REPRESENTATION (3P)
NAME
INQUIRE INTERIOR REPRESENTATION – obtain interior representation on specified workstation

SYNOPSIS
C Syntax
void
pinq_int_rep ( ws, index, type, error_ind, rep )
Pint ws;  // workstation identifier
Pint index;  // interior index
Pinq_type type;  // type of returned value
Pint *error_ind;  // OUT error indicator
Pint_bundle *rep;  // OUT interior representation

FORTRAN Syntax
SUBROUTINE pqir ( WKID, II, TYPE, ERRIND, INTS, ISTYLI, COLI )
INTEGER WKID  // workstation identifier
INTEGER II  // interior index
INTEGER TYPE  // type of returned values (PSET, PREALI)
INTEGER ERRIND  // OUT error indicator
INTEGER INTS  // OUT interior style
INTEGER ISTYLI  // OUT interior style index
INTEGER COLI  // OUT interior color index

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE INTERIOR REPRESENTATION to determine the current attribute values for a specified entry in a specified workstation’s table of defined interior representations. See the description of the subroutine SET INTERIOR REPRESENTATION for information about the meaning of these attribute values.

C Input Parameters
ws  // Workstation identifier.
index  // Entry to be returned from the workstation’s table of interior representations; if this entry is not present in the table and the type parameter is REALIZED, the representation for interior index 1 is returned.
type  // An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:
PINQ_SET  // Return application-specified value
PINQ_REALIZED  // Return value available on the workstation

modified 2 April 1993
C Output Parameters

error_ind  
A pointer to the location to store the error number of any error that this function detects.

rep  
A pointer to a Pint_bundle structure in which the system returns the interior representation at index in the workstation’s table of interior representations. Pint_bundle is defined in phigs.h as follows:

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

Pint_style is defined in phigs.h as:

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

FORTRAN Input Parameters

WKID  
Workstation identifier.

II  
Entry to be returned from the workstation’s table of interior representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for interior index 1 is returned.

TYPE  
An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

    PSET  Return application-specified value
    PREALI Return value available on the workstation

FORTRAN Output Parameters

ERRIND  
The error number of any error detected by this function.

INTS  
The interior style at index II in the workstation’s table of interior representations.

STYLI  
The interior style index at index II in the workstation’s table of interior representations.

COLI  
The interior colour index at index II in the workstation’s table of interior representations.
| ERRORS | 003 Ignoring function, function requires state \((PHOP, WSOP, *, *)\) |
|        | 054 Ignoring function, the specified workstation is not open |
|        | 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO) |
|        | 100 Ignoring function, the bundle index value is less than one |
|        | 101 Ignoring function, the specified representation has not been defined |
|        | 134 Ignoring function, the requested entry contains a general colour specification with \textit{colour type} other than INDIRECT. |

**SEE ALSO**

- SET INTERIOR REPRESENTATION (3P)
- INQUIRE PREDEFINED INTERIOR REPRESENTATION (3P)
- INQUIRE INTERIOR REPRESENTATION PLUS (3PP)
NAME

INQUIRE INVISIBILITY FILTER – obtain inclusion and exclusion name sets for specified workstation’s invisibility filter

SYNOPSIS

C Syntax

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

FORTRAN Syntax

SUBROUTINE pqivft ( WKID, ISBSZ, ESBSZ, ERRIND, ISN, IS, ESN, ES )
INTEGER WKID workstation identifier
INTEGER ISBSZ inclusion set buffer size
INTEGER ESBSZ exclusion set buffer size
INTEGER ERRIND OUT error indicator
INTEGER IS(ISBSZ) OUT number of names in the inclusion set
INTEGER ISN OUT inclusion set
INTEGER ESN OUT number of names in the exclusion set
INTEGER ES(ESBSZ) OUT exclusion set

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE INVISIBILITY FILTER to obtain the invisibility filter from a specified workstation’s state list. This is the filter which is compared to the traversal-time current name set of each primitive to determine if the primitive is invisible.

The filter contains an inclusion set and an exclusion set of names. During traversal, a primitive is eligible for invisibility if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. Each name in the name set, inclusion set, and exclusion set is a small positive integer.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by invis_filter.

ws The workstation identifier of the workstation whose invisibility filter is to be returned.

store The memory buffer PHIGS is to use for storing the information returned. This
buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

**error_ind**
A pointer to the location to store the error number of any error that this function detects.

**invis_filter**
A pointer to a pointer that points to the invisibility filter. Pint_list is defined in phigs.h as follows:

```c
typedef struct {
    Pint_list incl_set; // inclusion set
    Pint_list excl_set; // exclusion set
} Filter;
```

Pint_list is defined in phigs.h as follows:

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

### FORTRAN Input Parameters

**WKID**
The *workstation identifier* of the workstation whose state list is queried.

**ISBSZ**
Size of the IS array in which the returned *inclusion set* elements will be stored. If this value is smaller than the actual number of elements in the *inclusion set* (ISN), no data will be returned in the IS array, but ISN will be set to indicate the array size required. If you call this function with an array size of zero, ISN is returned with the required array size. Error 2001 is returned if ISBSZ is too small, but not if it’s zero.

**ESBSZ**
Size of the ES array in which the returned *exclusion set* elements will be stored. If this value is smaller than the actual number of elements in the *exclusion set* (ESN), no data will be returned in the ES array, but ESN will be set to indicate the array size required. If you call this function with an array size of zero, ESN is returned with the required array size. Error 2001 is returned if ESBSZ is too small, but not if it’s zero.

### FORTRAN Output Parameters

**ERRIND**
The error number of any error that this function detects.

**ISN**
The number of *inclusion set* elements returned in the IS array.

**IS**
An array of integers in which the elements of the filter’s *inclusion set* are returned.

**ESN**
The number of *exclusion set* elements returned in the ES array.

**ES**
An array of integers in which the elements of the filter’s *exclusion set* are returned.

modified 2 April 1993
## Errors

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
</tbody>
</table>

## See Also

SET INVISIBILITY FILTER (3P)
### NAME

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES – obtain list of workstation-available 2D generalized drawing primitives

### SYNOPSIS

**C Syntax**

```c
void pinq_list_avail_gdp ( type, length, start, error_ind, gdps, total_length )
```

- `Pint type;`  
  workstation type
- `Pint length;`  
  length of application list
- `Pint start;`  
  starting position
- `Pint *error_ind;`  
  OUT error indicator
- `Pint_list *gdps;`  
  OUT list of GDP identifiers
- `Pint *total_length;`  
  OUT length of list in PHIGS

**FORTRAN Syntax**

```fortran
SUBROUTINE pqegdp ( WTYPE, N, ERRIND, NGDP, GDPL )
```

- `INTEGER WTYPE`  
  workstation type
- `INTEGER N`  
  list element requested
- `INTEGER ERRIND`  
  OUT error indicator
- `INTEGER NGDP`  
  OUT number of available generalized drawing primitives
- `INTEGER GDPL`  
  OUT Nth element of list of GDP identifiers

**Required PHIGS Operating States**

`(PHOP, *, *, *)`

### DESCRIPTION

#### Purpose

Use INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES to obtain a list of the identifiers of the generalized drawing primitives (GDP)s available on the specified workstation type.

#### C Input Parameters

- `type`  
  Workstation type.
- `length`  
  The number of `ints` items in the `gdps` output parameter for which the application has allocated memory. The `length` is the number of list elements (GDP identifiers) that the system can return in the list of integers. If a value of zero is used here, no data will be returned in the list of integers, but the total number of GDPs available on the specified workstation type will be returned in `total_length`.
- `start`  
  Starting position in the workstation type list of generalized drawing primitive ids at which the inquiry is to begin. The elements of the list of GDP identifiers, beginning with the item number specified by `start`, are copied sequentially into the list of integers until it is full or all the GDP identifiers have been copied.

#### C Output Parameters

- `error_ind`  
  A pointer to the location to store the error number of any error that this function detects.
- `gdps`  
  A pointer to a Pint_list structure in which the system returns the list of identifiers

*modified 2 April 1993*
of the GDPs available on the specified type of workstation. Pint_list is defined in phigs.h as follows:

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

The `num_ints` component specifies the number of GDP identifiers in the list. The `ints` component is a pointer to a list, `num_ints` long, of the GDP identifiers.

The application must allocate memory for `length` elements in the list of integers.

```
total_length
```

A pointer to an integer in which the total number of elements in the specified workstation type list of available GDPs is returned. This is the value required for `length` if all GDP identifiers are to be returned.

**FORTRAN Input Parameters**

- `WTYPE` Workstation type.
- `N` Element of the specified workstation type list of available GDPs to return. Only one GDP identifier may be queried for per subroutine call. If a value of zero is used here, no GDP identifier will be returned, but the total number of elements in the workstation type list of available GDPs will be returned in `NGDP`.

**FORTRAN Output Parameters**

- `ERRIND` The error number of any error that this function detects.
- `NGDP` The total number of elements in the specified workstation type list of available GDPs.
- `GDPL` The identifier of the `N`th GDP from the specified workstation type list of available GDPs.

**ERRORS**

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- 052 Ignoring function, workstation type not recognized by the implementation
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- 062 Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

- PHIGS WORKSTATION DESCRIPTION TABLE (7P)
- GENERALIZED DRAWING PRIMITIVE (3P)
- INQUIRE GENERALIZED DRAWING PRIMITIVE (3P)

606 modified 2 April 1993
**NAME**
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 – obtain list of workstation-available 3D generalized drawing primitives

**SYNOPSIS**

**C Syntax**
```c
void
pinq_list_avail_gdp3 ( type, length, start, error_ind, gdps, total_length )
Pint type;  /* workstation type */
Pint length; /* length of application list */
Pint start;  /* starting position */
Pint *error_ind; /* OUT error indicator */
Pint_list *gdps; /* OUT list of 3D GDP identifiers */
Pint *total_length; /* OUT length of list in PHIGS */
```

**FORTRAN Syntax**
```fortran
SUBROUTINE pqegd3 ( WTYPE, N, ERRIND, NGDP, GDPL )
INTEGER WTYPE  /* workstation type */
INTEGER N  /* list element requested */
INTEGER ERRIND /* OUT error indicator */
INTEGER NGDP /* OUT number of available 3D generalized drawing primitives */
INTEGER GDPL /* OUT Nth element of list of 3D GDP identifiers */
```

**Required PHIGS Operating States**
(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**
Use INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 to obtain a list of the identifiers of the 3D generalized drawing primitives (GDP)s available on the specified workstation type.

**C Input Parameters**
- **type** Workstation type.
- **length** The number of ints items in the *gdps* output parameter for which the application has allocated memory. The **length** is the number of list elements (GDP 3 identifiers) that the system can return in the list of integers. If a value of zero is used here, no data will be returned in the list of integers, but the total number of GDP 3s available on the specified workstation type will be returned in **total_length**.
- **start** Starting position in the workstation type list of 3D generalized drawing primitive ids at which the inquiry is to begin. The elements of the list of GDP 3 identifiers, beginning with the item number specified by **start**, are copied sequentially into the list of integers until it is full or all the GDP 3 identifiers have been copied.

**C Output Parameters**
- **error_ind** A pointer to the location to store the error number of any error that this function detects.
- **gdps** A pointer to a Pint_list structure in which the system returns the list of identifiers.
of the GDP 3s available on the specified type of workstation. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of GDP 3 identifiers in the list. The ints component is a pointer to a list, num_ints long, of the GDP 3 identifiers.

The application must allocate memory for length elements in the list of integers.

total_length

A pointer to an integer in which the total number of elements in the specified workstation type list of available GDP 3s is returned. This is the value required for length if all GDP 3 identifiers are to be returned.

FORTRAN Input Parameters

WTYPE Workstation type.
N Element of the specified workstation type list of available GDP 3s to return. Only one GDP 3 identifier may be inquired per subroutine call. If a value of zero is used here, no GDP 3 identifier will be returned, but the total number of elements in the workstation type list of available GDP 3s will be returned in NGDP.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.
NGDP The total number of elements in the specified workstation type list of available GDP 3s.
GDPL The identifier of the Nth GDP 3 from the specified workstation type list of available GDP 3s.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
052 Ignoring function, workstation type not recognized by the implementation
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
GENERALIZED DRAWING PRIMITIVE 3 (3P)
INQUIRE GENERALIZED DRAWING PRIMITIVE 3 (3P)
NAME
INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS – obtain list of workstation-dependent generalized structure elements

SYNOPSIS
C Syntax
void
pinq_list_avail_gse ( type, length, start, error_ind, gses, total_length )
Pint type;  workstation type
Pint length; length of application list
Pint start; starting position
Pint *error_ind; OUT error indicator
Pint_list *gses; OUT list of GSEs
Pint *total_length; OUT length of list in PHIGS

FORTRAN Syntax
SUBROUTINE pqegse ( WTYPE, N, ERRIND, NGSE, GSEL )
INTEGER WTYPE  workstation type
INTEGER N  list element requested
INTEGER ERRIND  OUT error indicator
INTEGER NGSE  OUT number of available generalized structure elements
INTEGER GSEL  OUT nth element of list of GSE identifiers

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS to obtain a list of the identifiers of the workstation-dependent generalized structure elements (GSEs) available on the specified workstation type. This function returns a value of zero (0) for SunPHIGS available generalized structure elements because they are currently all workstation-independent. For a list of values that this function can return, refer to PHIGS DESCRIPTION TABLE.

C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Workstation type.</td>
</tr>
<tr>
<td>length</td>
<td>The number of ints items in the gses output parameter for which the application has allocated memory. The length is the number of list elements (GSE identifiers) that the system can return in the list of integers. If a value of zero is used here, no data will be returned in the list of integers, but the total number of GSEs available on the specified workstation type will be returned in total_length.</td>
</tr>
<tr>
<td>start</td>
<td>Starting position in the workstation type list of GSE identifiers at which the inquiry is to begin. The elements of the list of GSE identifiers, beginning with the item number specified by start, are copied sequentially into the list of integers until it is full or all the GSE identifiers have been copied.</td>
</tr>
</tbody>
</table>

modified 2 April 1993
C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error that this function detects.

- **gses**
  A pointer to a Pint_list structure in which the system returns the list of identifiers of the GSEs available on the specified type of workstation. Pint_list is defined in phigs.h as follows:

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

  The `num_ints` component specifies the number of GSE identifiers in the list. The `ints` component is a pointer to a list, `num_ints` long, of the GSE identifiers.

- **total_length**
  A pointer to an integer in which the total number of elements in the specified workstation type list of available GSEs is returned. This is the value required for length if all GSE identifiers are to be returned.

FORTRAN Input Parameters

- **WTYPE**
  Workstation type.

- **N**
  Element of the specified workstation type list of available GSEs to return. Only one GSE identifier may be inquired per subroutine call. If a value of zero is used here, no GSE identifier will be returned, but the total number of elements in the workstation type list of available GSEs will be returned in `NGSE`.

FORTRAN Output Parameters

- **ERRIND**
  The error number of any error that this function detects.

- **NGSE**
  The total number of elements in the specified workstation type list of available GSEs.

- **GSEL**
  The identifier of the Nth GSE from the specified workstation type list of available GSEs.

ERRORS

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

- **051**
  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

- **052**
  Ignoring function, workstation type not recognized by the implementation

- **059**
  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

- **062**
  Ignoring function, this information is not available for this MO workstation type

modified 2 April 1993
SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
GENERALIZED STRUCTURE ELEMENT (3P)
INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES (3P)
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES—obtain current list of workstation types defined in PHIGS description table

**SYNOPSIS**

**C Syntax**

```c
void
pinq_list_avail_ws_types ( length, start, error_ind, types, length_list )
```

- `length`: length of application list
- `start`: starting position
- `*error_ind`: OUT error indicator
- `*types`: OUT list of ws types
- `*length_list`: OUT length of list in PHIGS

**FORTRAN Syntax**

```fortran
SUBROUTINE pqewk ( N, ERRIND, NUMBER, WKTYP )
```

- `N`: list element requested
- `ERRIND`: OUT error indicator
- `NUMBER`: OUT number of workstation types
- `WKTYP`: OUT Nth element of list of available workstation types

**Required PHIGS Operating States**

`(PHOP, *, *, *)`

**DESCRIPTION**

**Purpose**

Use `INQUIRE LIST OF AVAILABLE WORKSTATION TYPES` to obtain a list of the currently available workstation types stored in the PHIGS description table. This list contains the predefined workstation types described in OPEN WORKSTATION and user-created workstation types created with WORKSTATION TYPE CREATE. User-created types are removed from this list by WORKSTATION TYPE DESTROY.

**C Input Parameters**

- `length`: The number of `ints` in the `types` output parameter for which the application has allocated memory. `length` is the number of list elements that the system can return in `types→ints`. If a value of 0 is used here, no data will be returned in the `types→ints` list, but the total number of elements will be returned in `length_list`.
- `start`: Starting position of inquiry. The elements in the list, beginning with the item number specified by `start`, are copied sequentially into `types→ints` until `types→ints` is full or all the elements have been copied.

**C Output Parameters**

- `*error_ind`: A pointer to the location for storing the error number of any error that this function detects. See the `ERRORS` section below for possible return values.
- `*types`: A pointer to a Pint_list structure in which to return the number and list of available workstation types. Pint_list is defined in phigs.h as follows:

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

modified 2 April 1993
Pint *ints; /* list of Pints */

The \texttt{num\_ints} component specifies the number of elements in the list. The \texttt{ints} component is a pointer to a list \texttt{num\_ints} long.

The application must allocate memory for \texttt{length} elements in the list of \texttt{ints}.

\textbf{length\_list}

A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for \texttt{length} if all elements in the list are to be returned.

\textbf{FORTRAN Input Parameters}

The \texttt{FORTRAN} function does not return the complete list of workstation types. Instead, it returns only one element of this list. The element to return is indicated by the calling program via the argument \texttt{N}.

\texttt{N} The index of the workstation type list entry to return.

\textbf{FORTRAN Output Parameters}

\texttt{ERRIND} The error indicator. See the Execution section below for a description of its use. See the \texttt{ERRORS} section below for possible return values.

\texttt{NUMBER} The number of available workstation types.

\texttt{WKTyp} The workstation type corresponding to the \texttt{N}th position in the list of workstation types.

\textbf{Execution}

\texttt{INQUIRE LIST OF AVAILABLE WORKSTATION TYPES} returns the list of available workstation types stored in the \texttt{PHIGS} description table. See \texttt{OPEN WORKSTATION} and \texttt{WORKSTATION TYPE CREATE} for a description of the available workstation types and information on how to create new ones.

If this function detects an error, then the \texttt{error indicator} indicates the error number of the detected error and no other output data is returned, except in the cases mentioned in the \texttt{C Parameters} section above. If the function detects no error, then the \texttt{error indicator} is set to zero and the inquired information is available in the output parameters. Since this is an inquiry function, \texttt{ERROR HANDLING} is not invoked when this function detects an error.

\textbf{ERRORS}

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

2200 C: Buffer overflow in input or inquiry function

\textbf{SEE ALSO}

\texttt{OPEN WORKSTATION (3P)}

\texttt{PHIGS WORKSTATION DESCRIPTION TABLE (7P)}

\texttt{WORKSTATION TYPE CREATE (3P)}

\texttt{WORKSTATION TYPE DESTROY (3P)}

\texttt{WORKSTATION TYPE SET (3P)}

\textit{modified 2 April 1993}
NAME

INQUIRE LIST OF COLOUR INDICES – obtain list of colour indices defined on specified workstation

SYNOPSIS

C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pqeci ( WKID, N, ERRIND, OL, COLI )
INTEGER  WKID  /* workstation identifier */
INTEGER  N  /* list element requested */
INTEGER  ERRIND  /* OUT error indicator */
INTEGER  OL  /* OUT number of colour table entries */
INTEGER  COLI  /* OUT Nth element of list of colour indices */
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE LIST OF COLOUR INDICES to obtain a list of the currently defined colour indices on the specified workstation. The defined colour table indices are conceptually sparse: a PHIGS implementation could allow definition of \( n \) colours with arbitrary indices. However, colour indices are nonsparse in SunPHIGS: the \( n \) colour representations necessarily have indices 0 through \( n - 1 \).

C Input Parameters

- **ws**: Workstation identifier of the workstation whose colour indices are to be returned.
- **length**: Number of items for which the application has allocated memory in the output parameter indices. Specify 0 to get the total length of the list.
- **start**: Starting position in the list at which to begin the inquiry.

C Output Parameters

- **error_ind**: Pointer to the location for storing the error number of any error that this function detects.
- **indices**: A pointer to a Pint_list in which the system returns the portion of the list of currently defined colour indices, starting at the entry specified with **start**. Pint_list is defined in phigs.h as follows:
  ```c
  typedef struct {
  ...}
  ```
  modified 2 April 1993
INQUIRE LIST OF COLOUR INDICES (3P)

SunPHIGS Release 3.0

Pint num_ints; /* number of Pints in list */
Pint *ints; /* list of integers */
)

Pint_list;

The pointer indices→*ints must be initialized to an array of length Pint elements.

total_length

Pointer to an integer in which to return the total length of the list. This is the value required for length if all the items in the list are to be returned.

<table>
<thead>
<tr>
<th>FORTRAN Input Parameters</th>
<th>WKID</th>
<th>Workstation identifier of the workstation whose list of colour indices is queried.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>Position in the list of the item requested. Nth defined colour index is returned in COLI.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORTRAN Output Parameters</th>
<th>ERREND</th>
<th>Error number of any error that this function detects.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OL</td>
<td>Total length of the list.</td>
</tr>
<tr>
<td></td>
<td>COLI</td>
<td>Nth defined colour index.</td>
</tr>
</tbody>
</table>

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO

SET COLOUR REPRESENTATION (3P)
INQUIRE PREDEFINED COLOUR REPRESENTATION (3P)

modified 2 April 1993
INQUIRE LIST OF EDGE INDICES

NAME
INQUIRE LIST OF EDGE INDICES—obtain list of edge indices defined on specified workstation

SYNOPSIS

C Syntax

void
pinq_list_edge_inds ( ws, length, start, error_ind, indices, total_length )

Pint wS; // workstation identifier
Pint length; // length of application list
Pint start; // starting position
Pint *error_ind; // OUT error indicator
Pint_list *indices; // OUT list of edge indices
Pint *total_length; // OUT length of list in PHIGS

FORTRAN Syntax

SUBROUTINE pqeedi ( WKID, N, ERRIND, OL, EDI )

INTEGER WKID // workstation identifier
INTEGER N // list element requested
INTEGER ERRIND // OUT error indicator
INTEGER OL // OUT number of edge bundle table entries
INTEGER EDI // OUT Nth element of list of defined edge indices

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose
Use INQUIRE LIST OF EDGE INDICES to obtain a list of the currently defined edge indices on the specified workstation.

C Input Parameters

ws // Workstation identifier of the workstation whose edge indices are to be returned.
length // Number of items for which the application has allocated memory in the output parameter indices. Specify 0 to get the total length of the list.
start // Starting position in the list at which to begin the inquiry.

C Output Parameters

error_ind // Pointer to the location for storing the error number of any error that this function detects.
indices // Pointer to a Pint_list structure in which the system returns the portion of the list of currently defined edge indices, starting at the entry specified with start.
Pint_list is defined in phigs.h as follows:

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

modified 2 April 1993
Pointer *indices*→*ints* must be initialized to an array of length *Pint* elements.

*total_length*

Pointer to an integer in which to return the total length of the list. This is the value required for *length* if all the items in the list are to be returned.

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>WKID</th>
<th>Workstation identifier of the workstation whose list of edge indices is queried.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Position in the list of the item requested. The <em>N</em>th defined edge index is returned in <em>EDI</em>.</td>
</tr>
</tbody>
</table>

**FORTRAN Output Parameters**

<table>
<thead>
<tr>
<th>ERRIND</th>
<th>Error number of any error that this function detects.</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>OL</em></td>
<td><em>Total length</em> of the list.</td>
</tr>
<tr>
<td><em>EDI</em></td>
<td><em>N</em>th defined edge index.</td>
</tr>
</tbody>
</table>

**ERRORS**

- 003 Ignoring function, function requires state (*PHOP*, *WSOP*, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

- SET EDGE REPRESENTATION (3P)
- INQUIRE EDGE REPRESENTATION (3P)
NAME
INQUIRE LIST OF INTERIOR INDICES—obtain list of interior indices defined on a workstation

SYNOPSIS
C Syntax

```c
void pinq_list_int_inds ( ws, length, start, error_ind, indices, total_length )
```

- `Pint ws;` workstation identifier
- `Pint length;` length of application list
- `Pint start;` starting position
- `Pint *error_ind;` OUT error indicator
- `Pint_list *indices;` OUT list of interior indices
- `Pint *total_length;` OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqeii ( WKID, N, ERRIND, OL, II )
```

- `INTEGER WKID` workstation identifier
- `INTEGER N` list element requested
- `INTEGER ERRIND` OUT error indicator
- `INTEGER OL` OUT number of interior bundle table entries
- `INTEGER II` OUT Nth element of list of defined interior indices

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose

Use INQUIRE LIST OF INTERIOR INDICES to obtain a list of the currently-defined interior indices on the specified workstation.

C Input Parameters

- `ws` The workstation identifier of the workstation whose interior indices are to be returned.
- `length` Number of items for which the application has allocated memory in the output parameter `indices`. 0 can be specified, in order to get the total length of the list.
- `start` Starting position in the list at which to begin the inquiry.

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error that this function detects.
- `indices` A pointer to a Pint_list structure in which the system returns the portion of the list of currently-defined interior indices, starting at the entry specified with `start`. Pint_list is defined in phigs.h as follows:

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

modified 2 April 1993
The pointer `indices->ints` must be initialized to an array of length `Pint` elements.

`total_length`
A pointer to an integer in which to return the total length of the list. This is the value required for `length` when all the items in the list are to be returned.

**FORTRAN Input Parameters**
- **WKID**  
  The workstation identifier of the workstation whose list of interior indices is queried.
- **N**  
  Position in the list of the item requested. The Nth defined interior index will be returned in `II`.

**FORTRAN Output Parameters**
- **ERRIND**  
  The error number of any error that this function detects.
- **OL**  
  The total length of the list.
- **II**  
  The Nth defined interior index.

**ERRORS**
- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**
- SET INTERIOR REPRESENTATION (3P)
- SET INTERIOR INDEX (3P)
- INQUIRE INTERIOR REPRESENTATION (3P)
INQUIRE LIST OF PATTERN INDICES – obtain list of pattern indices defined on specified workstation

C Syntax

```c
void
pinq_list_pat_inds ( ws, length, start, error_ind, indices, total_length )
```

- `Pint ws;` workstation identifier
- `Pint length;` length of application list
- `Pint start;` starting position
- `Pint *error_ind;` OUT error indicator
- `Pint_list *indices;` OUT list of pattern indices
- `Pint *total_length;` OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqepai ( WKID, N, ERRIND, OL, PAI )
```

- `INTEGER WKID` workstation identifier
- `INTEGER N` list element requested
- `INTEGER ERRIND` OUT error indicator
- `INTEGER OL` OUT number of pattern table entries
- `INTEGER PAI` OUT Nth element of list of pattern indices

Required PHIGS Operating States

`(PHOP, WSOP, *, *)`

DESCRIPTION

Purpose

Use INQUIRE LIST OF PATTERN INDICES to obtain a list of the currently-defined pattern indices on the specified workstation.

C Input Parameters

- `ws` The `workstation identifier` of the workstation whose pattern indices are to be returned.
- `length` Number of items for which the application has allocated memory in the output parameter `indices`. 0 can be specified, in order to get the `total length` of the list.
- `start` Starting position in the list at which to begin the inquiry.

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error that this function detects.
- `indices` A pointer to a `Pint_list` structure in which the system returns the portion of the list of currently-defined pattern indices, starting at the entry specified with `start`.

Pint_list is defined in `phigs.h` as follows:

```c
typedef struct {
    Pint num_ints; /* number of Pints in list */
    Pint *ints;  /* list of integers */
} Pint_list;
```
The pointer \texttt{indices->ints} must be initialized to an array of length \texttt{Pint} elements.

\texttt{total_length}

A pointer to an integer in which to return the total length of the list. This is the value required for \texttt{length} if all the items in the list are to be returned.

**FORTRAN Input Parameters**

- \texttt{WKID} The workstation identifier of the workstation whose list of pattern indices is queried.
- \texttt{N} Position in the list of the item requested. The Nth defined pattern index will be returned in \texttt{PAI}.

**FORTRAN Output Parameters**

- \texttt{ERRIND} The error number of any error that this function detects.
- \texttt{OL} The total length of the list.
- \texttt{PAI} The Nth defined pattern index.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

- \texttt{SET PATTERN REPRESENTATION (3P)}
- \texttt{INQUIRE PATTERN REPRESENTATION (3P)}
- \texttt{INQUIRE PREDEFINED PATTERN REPRESENTATION (3P)}

modified 2 April 1993
NAME
INQUIRE LIST OF POLYLINE INDICES—obtain list of polyline indices defined on specified workstation

SYNOPSIS
C Syntax

void
pinq_list_line_inds ( ws, length, start, error_ind, indices, total_length )

Pint  ws;  /* workstation identifier */
Pint  length;  /* length of application list */
Pint  start;  /* starting position */
Pint  *error_ind;  /* OUT error indicator */
Pint_list  *indices;  /* OUT list of polyline indices */
Pint  *total_length;  /* OUT length of list in PHIGS */

FORTRAN Syntax

SUBROUTINE pqepli ( WKID, N, ERRIND, OL, PLI )

INTEGER  WKID  /* workstation identifier */
INTEGER  N  /* list element requested */
INTEGER  ERRIND  /* OUT error indicator */
INTEGER  OL  /* OUT number of polyline bundle table entries */
INTEGER  PLI  /* OUT Nth element of list of defined polyline indices */

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE LIST OF POLYLINE INDICES to obtain a list of the currently-defined polyline indices on the specified workstation.

C Input Parameters

ws  The workstation identifier of the workstation whose polyline indices are to be returned.

length  Number of items for which the application has allocated memory in the output parameter indices. 0 can be specified, in order to get the total length of the list.

start  Starting position in the list at which to begin the inquiry.

C Output Parameters

error_ind  A pointer to the location to store the error number of any error that this function detects.

indices  A pointer to a Pint_list structure in which the system returns the portion of the list of currently-defined polyline indices, starting at the entry specified with start.

Pint_list is defined in phigs.h as follows:

typedef struct {
    Pint  num_ints;  /* number of Pints in list */
    Pint  *ints;  /* list of integers */
}
The pointer $\text{indices} \rightarrow \text{ints}$ must be initialized to an array of length $\text{Pint}$ elements. A pointer to an integer in which to return the total length of the list. This is the value required for length if all the items in the list are to be returned.

**FORTRAN Input Parameters**

- **WKID**: The workstation identifier of the workstation whose list of polyline indices is queried.
- **$N$**: Position in the list of the item requested. The $N$th defined polyline index will be returned in PLI.

**FORTRAN Output Parameters**

- **ERRIND**: The error number of any error that this function detects.
- **$OL$**: The total length of the list.
- **PLI**: The $N$th defined polyline index.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

inquire_list_of_polyline_indices.3

SET POLYLINE REPRESENTATION (3P)

INQUIRE POLYLINE REPRESENTATION (3P)

INQUIRE PREDEFINED POLYLINE REPRESENTATION (3P)
NAME
INQUIRE LIST OF POLYMARKER INDICES—obtain list of polymarker indices defined on specified workstation

SYNOPSIS
C Syntax
void
pinq_list_marker_inds ( ws, length, start, error_ind, indices, total_length )
Pint ws; /* workstation identifier */
Pint length; /* length of application list */
Pint start; /* starting position */
Pint *error_ind; /* OUT error indicator */
Pint_list *indices; /* OUT list of polymarker indices */
Pint *total_length; /* OUT length of list in PHIGS */

FORTRAN Syntax
SUBROUTINE pqepmi ( WKID, N, ERRIND, OL, PMI )
INTEGER WKID /* workstation identifier */
INTEGER N /* list element requested */
INTEGER ERRIND /* OUT error indicator */
INTEGER OL /* OUT number of polymarker bundle table entries */
INTEGER PMI /* OUT Nth element of list of defined polymarker indices */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE LIST OF POLYMARKER INDICES to obtain a list of the currently-defined polymarker indices on the specified workstation.

C Input Parameters
ws The workstation identifier of the workstation whose polymarker indices are to be returned.
length Number of items for which the application has allocated memory in the output parameter indices. 0 can be specified, in order to get the total length of the list.
start Starting position in the list at which to begin the inquiry.

C Output Parameters
error_ind A pointer to the location to store the error number of any error that this function detects.
indices A pointer to a Pint_list structure in which the system returns the portion of the list of currently-defined polymarker indices, starting at the entry specified with start.

Pint_list is defined in phigs.h as follows:

typedef struct {
  Pint num_ints; /* number of Pints in list */
  Pint *ints; /* list of integers */
} Pint_list;
the pointer indices→ints must be initialized to an array of length Pint elements.

\texttt{total\_length}

A pointer to an integer in which to return the total length of the list. This is the value required for \texttt{length} if all the items in the list are to be returned.

\textbf{FORTRAN Input Parameters}

\texttt{WKID} \hspace{1cm} \textit{The workstation identifier} of the workstation whose list of polymarker indices is queried.

\texttt{N} \hspace{1cm} \textit{Position in the list of the item requested.} The \texttt{N}th defined polymarker index will be returned in \texttt{PMI}.

\textbf{FORTRAN Output Parameters}

\texttt{ERRIND} \hspace{1cm} \textit{The error number of any error that this function detects.}

\texttt{OL} \hspace{1cm} \textit{The \texttt{total \_length} of the list.}

\texttt{PMI} \hspace{1cm} \textit{The \texttt{N}th defined polymarker index.}

\textbf{ERRORS}

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

\textbf{SEE ALSO}

\texttt{SET POLYMARKER REPRESENTATION (3P)}

\texttt{INQUIRE POLYMARKER REPRESENTATION (3P)}

\texttt{INQUIRE PREDEFINED POLYMARKER REPRESENTATION (3P)}
NAME
INQUIRE LIST OF TEXT INDICES—obtain list of text indices defined on specified workstation

SYNOPSIS
C Syntax
void
pinq_list_text_inds ( ws, length, start, error_ind, indices, total_length )

Pint  ws;  /* workstation identifier */
Pint  length;  /* length of application list */
Pint  start;  /* starting position */
Pint  *error_ind;  /* OUT error indicator */
Pint_list  *indices;  /* OUT list of text indices */
Pint  *total_length;  /* OUT length of list in PHIGS */

FORTRAN Syntax
SUBROUTINE pqetxi ( WKID, N, ERRIND, OL, TXI )

INTEGER  WKID  /* workstation identifier */
INTEGER  N  /* list element requested */
INTEGER  ERRIND  /* OUT error indicator */
INTEGER  OL  /* OUT number of text bundle table entries */
INTEGER  TXI  /* OUT Nth element of list of defined text indices */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE LIST OF TEXT INDICES to obtain a list of the currently-defined text indices on
the specified workstation.

C Input Parameters
ws  The workstation identifier of the workstation whose text indices are to be returned.
length  Number of items for which the application has allocated memory in the output
parameter indices. 0 can be specified, in order to get the total length of the list.
start  Starting position in the list at which to begin the inquiry.

C Output Parameters
error_ind  A pointer to the location to store the error number of any error that this function
detects.
indices  A pointer to a Pint_list structure in which the system returns the portion of the
list of currently-defined text indices, starting at the entry specified with start.
Pint_list is defined in phigs.h as follows:
typedef struct {
    Pint  num_ints;  /* number of Pints in list */
    Pint  *ints;  /* list of integers */
} Pint_list;

626  modified 2 April 1993
The pointer `indices→ints` must be initialized to an array of `length Pint` elements.

### `total_length`
A pointer to an integer in which to return the `total length` of the list. This is the value required for `length` if all the items in the list are to be returned.

### FORTRAN Input Parameters
- **WKID**
  - The workstation identifier of the workstation whose list of text indices is queried.
- **N**
  - Position in the list of the item requested. The Nth defined text index will be returned in `TXI`.

### FORTRAN Output Parameters
- **ERRIND**
  - The error number of any error that this function detects.
- **OL**
  - The `total length` of the list.
- **TXI**
  - The Nth defined text index.

### ERRORS
- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

### SEE ALSO
- SET TEXT REPRESENTATION (3P)
- INQUIRE TEXT REPRESENTATION (3P)
- INQUIRE PREDEFINED TEXT REPRESENTATION (3P)
NAME  INQUIRE LIST OF VIEW INDICES—obtain list of view indices defined on specified workstation

SYNOPSIS
C Syntax
void
pinq_list_view_inds ( ws, length, start, error_ind, indices, total_length )

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

FORTRAN Syntax
SUBROUTINE pqevwi ( WKID, N, ERRIND, NVWIX, VIEWI )

  INTEGER WKID          /* workstation identifier */
  INTEGER N              /* list element requested */
  INTEGER ERRIND         /* OUT error indicator */
  INTEGER NVWIX          /* OUT number of view bundle table entries */
  INTEGER VIEWI          /* OUT Nth element of list of defined view indices */

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE LIST OF VIEW INDICES to obtain a list of the currently-defined view indices on
the specified workstation. The list is in order of decreasing view transformation input
priority.

The defined view table indices are nonsparse; the n settable view representations
necessarily have indices 1 through n.

C Input Parameters
ws      The workstation identifier of the workstation whose view indices are to be
        returned.
length  Number of items for which the application has allocated memory in the output
        parameter indices. 0 can be specified, in order to get the total length of the list.
start   Starting position in the ordered list at which to begin the inquiry.

C Output Parameters
error_ind  A pointer to the location to store the error number of any error that this function
detects.
indices    A pointer to a Pint_list structure in which the system returns the portion of the
           list of currently-defined view indices, starting at the entry specified with start.
           Pint_list is defined in phigs.h as follows:
           typedef struct {
Pint num_ints; /* number of Pints in list */
Pint *ints; /* list of integers */
} Pint_list;

The pointer indices->ints must be initialized to an array of length Pint elements.

total_length

A pointer to an integer in which to return the total length of the list. This is the value required for length if all the items in the list are to be returned.

**FORTRAN Input Parameters**

- **WKID**
  The workstation identifier of the workstation whose list of view indices is queried.

- **N**
  Position in the ordered list of the item requested. The Nth defined view index will be returned in VIEWI.

**FORTRAN Output Parameters**

- **ERRIND**
  The error number of any error that this function detects.

- **NVWIX**
  The total length of the list.

- **VIEWI**
  The Nth defined view index.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

- SET VIEW REPRESENTATION (3P)
- INQUIRE VIEW REPRESENTATION (3P)
- INQUIRE PREDEFINED VIEW REPRESENTATION (3P)
### NAME

INQUIRE LOCATOR DEVICE STATE – inquire state of a locator device

### SYNOPSIS

**C Syntax**

```c
void
pinq_loc_st ( ws, dev, type, store, err, op_mode, echo_switch, init_view_ind,
              init_loc_pos, prompt_echo, echo_area, loc_data )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Pint ws;</code></td>
<td>workstation identifier</td>
</tr>
<tr>
<td><code>Pint dev;</code></td>
<td>locator device number</td>
</tr>
<tr>
<td><code>Pinq_type type;</code></td>
<td>type of returned value</td>
</tr>
<tr>
<td><code>Pstore store;</code></td>
<td>pointer to buffer</td>
</tr>
<tr>
<td><code>Pint *err;</code></td>
<td>OUT error indicator</td>
</tr>
<tr>
<td><code>Pop_mode *op_mode;</code></td>
<td>OUT operating mode</td>
</tr>
<tr>
<td><code>Pecho_switch *echo_switch;</code></td>
<td>OUT echo switch</td>
</tr>
<tr>
<td><code>Pint *init_view_ind;</code></td>
<td>OUT initial view index</td>
</tr>
<tr>
<td><code>Ppoint *init_loc_pos;</code></td>
<td>OUT initial locator position</td>
</tr>
<tr>
<td><code>Pint *prompt_echo;</code></td>
<td>OUT prompt/echo type</td>
</tr>
<tr>
<td><code>Plimit *echo_area;</code></td>
<td>OUT echo area</td>
</tr>
<tr>
<td><code>Ploc_data **loc_data;</code></td>
<td>OUT data record</td>
</tr>
</tbody>
</table>

**FORTRAN Syntax**

```fortran
SUBROUTINE pqlcs ( WKID, LCDNR, TYPE, MLDR, ERRIND, MODE, ESW,
                   IVIEWI, IPX, IPY, PET, EAREA, LDR, DATREC )
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>INTEGER WKID</code></td>
<td>workstation identifier</td>
</tr>
<tr>
<td><code>INTEGER LCDNR</code></td>
<td>locator device number</td>
</tr>
<tr>
<td><code>INTEGER TYPE</code></td>
<td>type of returned values (PSET, PREALI)</td>
</tr>
<tr>
<td><code>INTEGER MLDR</code></td>
<td>dimension of data record array</td>
</tr>
<tr>
<td><code>INTEGER ERRIND</code></td>
<td>OUT error indicator</td>
</tr>
<tr>
<td><code>INTEGER MODE</code></td>
<td>OUT operating mode (PREQU, PSAMPL, PEVENT)</td>
</tr>
<tr>
<td><code>INTEGER IVIEWI</code></td>
<td>OUT initial view index</td>
</tr>
<tr>
<td><code>REAL IPX, IPY</code></td>
<td>OUT initial locator position in World Coordinates</td>
</tr>
<tr>
<td><code>INTEGER PET</code></td>
<td>OUT prompt/echo type</td>
</tr>
<tr>
<td><code>REAL EAREA(4)</code></td>
<td>OUT echo area in Device Coordinates</td>
</tr>
<tr>
<td><code>INTEGER LDR</code></td>
<td>OUT number of array elements used in data record</td>
</tr>
<tr>
<td><code>CHARACTER*80 DATREC(MLDR)</code></td>
<td>OUT data record</td>
</tr>
</tbody>
</table>

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

---

630 modified 2 April 1993
**DESCRIPTION**

**Purpose**

Use **INQUIRE LOCATOR DEVICE STATE** to determine the current state of the specified locator device.

**C Input Parameters**

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS API. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, the implementation manages this area such that there is sufficient memory for the specific inquiry. The locator device data record within the store buffer is accessed by the pointer pointed to by `loc_data`.

- `ws` Workstation identifier. An integer specifying the workstation with which the specified locator device is associated.
- `dev` The device number of the locator device. See the **AVAILABLE DEVICES** section of **INITIALIZE LOCATOR** for a description of the available devices.
- `type` An enumerated value specifying whether the values to be returned are those originally specified by the application (SET), or those resulting after PHIGS mapped them to ones available on the workstation (REALIZED). A `Pinq_type` structure is defined as:

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

- `store` The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see `CREATE STORE (3P)`).

**C Output Parameters**

- `err` The error indicator. See the **Execution** section below for a description of its use. See the **ERRORS** section below for the possible values it may return.
- `op_mode` The operating mode. `Pop_mode` is an enumerated type with the following values:

```c
POP_REQ
POP_SAMPLE
POP_EVENT
```

- `echo_switch` The echo state. `Pecho_switch` is an enumerated type with the following values:

```c
PSWITCH_ECHO
PSWITCH_NO_ECHO
```

**init_view_ind**

modified 2 April 1993
Initial view index.

*init_loc_pos*

Initial locator position.

*prompt_echo*

The prompt/echo type desired. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available types.

*echo_area*

A pointer to a variable of type Plimit that contains the echo area of the device. Plimit is defined in phigs.h as follows:

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

*loc_data*

Pointer to a pointer to the locator device record within the store data buffer.

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial locator value, the prompt/echo type and the echo area.

Error 2001 is returned if MLDR is too small, but not if it is zero.

*WKID* The workstation identifier of the workstation associated with the device.

*LCDNSR* The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.

*TYPE* An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PSET Set</td>
</tr>
<tr>
<td>1</td>
<td>PREALI Realized</td>
</tr>
</tbody>
</table>

*MLDR* The dimension of the data record array, DATREC.
**FORTRAN Output Parameters**

- **ERRIND**: The error indicator. See the *Execution* section below for a description of its use. See the **ERRORS** section below for the possible values it may return.

- **MODE**: The operating mode.

- **ESW**: The echo switch.

- **VIEWI**: The initial view index.

- **IPX, IPY**: The initial locator position.

- **PET**: The prompt/echo type.

- **EAREA**: An array containing the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

- **LDR**: The required dimension of the data record array, DATREC.

- **DATREC**: The data record array.

**Execution**

INQUIRE LOCATOR DEVICE STATE returns the current state of the specified locator device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial locator value, prompt/echo type, echo area and data record. See SET LOCATOR MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE LOCATOR for a description of the initial locator value, prompt/echo type, echo area and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

- **003**: Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054**: Ignoring function, the specified workstation is not open
- **061**: Ignoring function, specified workstation is not of category INPUT or OUTIN
- **250**: Ignoring function, the specified device is not available on the specified workstation
- **2200**: C: Buffer overflow in input or inquiry function
- **2001**: FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

modified 2 April 1993
<table>
<thead>
<tr>
<th>SEE ALSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE LOCATOR (3P)</td>
</tr>
<tr>
<td>SET LOCATOR MODE (3P)</td>
</tr>
<tr>
<td>INQUIRE LOCATOR DEVICE STATE 3 (3P)</td>
</tr>
</tbody>
</table>
NAME
INQUIRE LOCATOR DEVICE STATE 3 – inquire state of a locator device

SYNOPSIS
C Syntax

void
pinq_loc_st3 ( ws, dev, type, store, err, op_mode, echo_switch, init_view_ind,
init_loc_pos, prompt_echo, echo_vol, loc_data )

Pint   ws;    /* workstation identifier */
Pint  dev;  /* locator device number */
Pinq_type type;  /* type of returned value */
Pstore store;  /* store handle */
Pint *err;    /* OUT error indicator */
Pop_mode *op_mode;  /* OUT operating mode */
Pecho_switch *echo_switch;  /* OUT echo switch */
Pint *init_view_ind;  /* OUT initial view index */
Ppoint3 *init_loc_pos;  /* OUT initial locator position */
Pint *prompt_echo;  /* OUT prompt/echo type */
Plimit3 *echo_vol;  /* OUT echo volume */
Ploc_data3 **loc_data;  /* OUT data record */

FORTRAN Syntax
SUBROUTINE pqles3 ( WKID, LCDNR, TYPE, MLDR, ERRIND, MODE, ESW,
IVIEWI, IPX, IPY, IPZ, PET, EVOL, LDR, DATREC )

INTEGER WKID    /* workstation identifier */
INTEGER LCDNR  /* locator device number */
INTEGER TYPE  /* type of returned values (PSET, PREALI) */
INTEGER MLDR  /* dimension of data record array */
INTEGER ERRIND  /* OUT error indicator */
INTEGER MODE  /* OUT operating mode (PREQU, PSAMPL, PEVENT) */
INTEGER ESW  /* OUT echo switch (PNECHO, PECHO) */
INTEGER IVIEWI  /* OUT initial view index */
REAL IPX, IPY, IPZ  /* OUT initial locator position in World Coordinates */
INTEGER PET  /* OUT prompt/echo type */
REAL EVOL(6)  /* OUT echo volume in Device Coordinates */
INTEGER LDR  /* OUT number of array elements used in data record */
CHARACTER*80 DATREC(MLDR)  /* OUT data record */

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

modified 2 April 1993
DESCRIPTION

Purpose

Use INQUIRE LOCATOR DEVICE STATE 3 to determine the current state of the specified locator device.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The locator device data record within the store buffer is accessed by the pointer pointed to by loc_data.

- **ws**: Workstation identifier. An integer specifying the workstation with which the specified locator device is associated.
- **dev**: The device number of the locator device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available devices.
- **type**: An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

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

- **store**: The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

- **err**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.
- **op_mode**: A pointer to a variable of type Pop_mode, which contains the current operating mode of the device. Pop_mode is enumerated in phigs.h as follows:

  ```c
  POP_REQ
  POP_SAMPLE
  POP_EVENT
  ```

- **echo_switch**: A pointer to a variable of type Pecho_switch, which contains the state of the devices echo switch. The value returned for echo_switch will be one of the following:

  ```c
  PSWGITCH_ECHO
  PSWGITCH_NO_ECHO
  ```

636 modified 2 April 1993
init_view_ind
Initial view index.

init_loc_pos
Initial locator position.

prompt_echo
The prompt/echo type desired. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR for a description of the available types.

echo_vol
A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;

loc_data
Points to a pointer to the locator device data within the store buffer.

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial locator value, the prompt/echo type and the echo volume.

Error 2001 is returned if MLDR is too small, but not if it is zero.

WKID The workstation identifier of the workstation associated with the device.

LCDNR The device number of the LOCATOR device. See the AVAILABLE DEVICES section of INITIALIZE LOCATOR 3 for a description of the available devices.

TYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (Set), or those resulting after PHIGS mapped them to ones available on the workstation (Realized). Valid values are:

0 PSET Set
1 PREALI Realized
FORTRAN Output Parameters

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MODE**  The operating mode.

**ESW**  The echo switch.

**VIEWI**  The initial view index.

**IPX, IPY, IPZ**  The initial locator position.

**PET**  The prompt/echo type.

**EVL**  An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, in Device Coordinates.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array.

Execution

INQUIRE LOCATOR DEVICE STATE 3 returns the current state of the specified locator device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial locator value, prompt/echo type, echo volume and data record. See SET LOCATOR MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE LOCATOR 3 for a description of the initial locator value, prompt/echo type, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero, and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

061 Ignoring function, specified workstation is not of category INPUT or OUTIN

250 Ignoring function, the specified device is not available on the specified workstation

2200 C: Buffer overflow in input or inquiry function

2001 **FORTRAN**: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

638 modified 2 April 1993
SEE ALSO

INITIALIZE LOCATOR 3 (3P)
SET LOCATOR MODE (3P)
INQUIRE LOCATOR DEVICE STATE (3P)
**NAME**
INQUIRE MODELLING CLIPPING FACILITIES – inquire number of modelling clipping planes and operators available

**SYNOPSIS**
**C Syntax**
```c
void
pinq_model_clip_facs ( length, start, error_ind, num, ops, total_length )
```

<table>
<thead>
<tr>
<th>C Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>length</strong></td>
</tr>
<tr>
<td><strong>start</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>error_ind</strong></td>
</tr>
<tr>
<td><strong>num</strong></td>
</tr>
<tr>
<td><strong>ops</strong></td>
</tr>
</tbody>
</table>

**FORTRAN Syntax**
```fortran
SUBROUTINE pqmclf ( N, ERRIND, NDPMCV, OL, MCLPOP )
```

<table>
<thead>
<tr>
<th>FORTRAN Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N</strong></td>
</tr>
<tr>
<td><strong>ERRIND</strong></td>
</tr>
<tr>
<td><strong>NDPMCV</strong></td>
</tr>
<tr>
<td><strong>OL</strong></td>
</tr>
<tr>
<td><strong>MCLPOP</strong></td>
</tr>
</tbody>
</table>

**Required PHIGS Operating States**
(Phop, *, *, *)

**DESCRIPTION**
**Purpose**
INQUIRE MODELLING CLIPPING FACILITIES obtains the number of planes and available operators that can be used to determine the modelling clipping volume in the PHIGS implementation.

**C Input Parameters**
- **length**: Number of items for which the application has allocated memory in the output parameter **ops**. 0 can be specified, in order to get the total length of the list of operators.
- **start**: Starting position in the list of operators at which to begin the inquiry.

**C Output Parameters**
- **error_ind**: A pointer to the location to store the error number of any error this function detects.
- **num**: A pointer to a location in which the system return the number of distinct planes (half-spaces) that can be used to determine the modelling clipping volume.
- **ops**: A pointer to a **Pint_list** in which the system returns the portion of the list of supported modelling clipping operators, starting at the entry specified with **start**. **Pint_list** is defined in phigs.h as follows:

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

} Pint_list;
The pointer ops→ints must be initialized to an array of length Pint elements.  

**total_length**

A pointer to an integer in which to return the total length of the list of supported operators. This is the value required for length if all the items in the list are to be returned.

**FORTRAN Input Parameters**

| N | Position in the list of the item requested. The Nth supported operator will be returned in MCLPOP. |

**FORTRAN Output Parameters**

| ERRIND | The error number of any error this function detects. |
| NDPMCV | The number of modelling clipping half-spaces (planes) that can be used to determine the modelling clipping volume. |
| OL | The total length of the list of supported operators. |
| MCLPOP | The Nth element in the list of supported modelling clipping operators. |

**ERRORS**

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

**SEE ALSO**

PHIGS DESCRIPTION TABLE (7P)
SET MODELLING CLIPPING INDICATOR (3P)
SET MODELLING CLIPPING VOLUME (3P)
### NAME
INQUIRE MORE SIMULTANEOUS EVENTS – see if there are more simultaneous events on the input queue

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqsim ( ERRIND, FLAG )
INTEGER ERRIND  OUT error indicator
INTEGER FLAG  OUT more simultaneous events (PNMORE, PMORE)
```

### Required PHIGS Operating States

( PHOP, *, *, * )

### DESCRIPTION

Use INQUIRE MORE SIMULTANEOUS EVENTS to determine if there are more simultaneous events on the input queue. Simultaneous events are generated when two or more input devices in EVENT mode share a trigger and that trigger is fired. One event for each device sharing the trigger is generated in such a case, and the events are marked as simultaneous.

### C Output Parameters

- **error_ind**
  A pointer to the error indicator. If an error is detected, the number of the error is stored in this variable and the contents of `events` is undefined.

- **events**
  A pointer to a `Pmore_simult_events` type defined in `phigs.h` as:
  ```c
typedef enum {
    PSIMULT_NO_MORE,
    PSIMULT_MORE
  } Pmore_simult_events;
```

### FORTRAN Output Parameters

- **ERRIND**
  The error indicator. If an error is detected the number of the error is stored in this variable and the contents of `FLAG` is undefined.

- **FLAG**
  A value indicating the existence of more simultaneous events. Possible values as defined in `phigs77.h` are:
  ```c
  PNMORE
  PMORE
  ```

### Execution

INQUIRE MORE SIMULTANEOUS EVENTS examines the `more simultaneous events` field in the PHIGS state list to see if the next event is the next in a group of simultaneous events. When the first event in a group of simultaneous events reaches the front of the input
event queue and is removed by `AWAIT EVENT`, the *more simultaneous events* field is set to MORE. When the last event in the group is removed from the event queue the value is set to NO MORE.

Simultaneous events are generated by input devices that share the same trigger. If these input devices are in EVENT mode and the trigger is fired, then an input event will be enqueued for each of the devices and each will be marked as one in a group of simultaneous events. The order in which they are enqueued is undefined. If there is not enough space on the input queue to hold this entire group of events, then none of them will be enqueued and one of them will be placed in the *identification of one of the logical input devices that caused an overflow* field in the PHIGS error state list.

SunPHIGS has many devices that can potentially share triggers. See the corresponding *initialize input class* functions for a description of the triggers each device uses.

**ERRORS**

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

**SEE ALSO**

- `AWAIT EVENT (3P)`
- `GET CHOICE (3P)`
- `GET LOCATOR (3P)`
- `GET PICK (3P)`
- `GET STRING (3P)`
- `GET STROKE (3P)`
- `GET VALUATOR (3P)`
NAME
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES – inquire the number of available logical input devices for a specified workstation type

SYNOPSIS
C Syntax
void
pinq_num_avail_in ( type, error_ind, numbers )

Pint type;  // workstation type
Pint *error_ind;  // OUT error indicator
Pnum_in *numbers;  // OUT number of input devices

FORTRAN Syntax
SUBROUTINE pqli ( WTYPE, ERRIND, NLCD, NSKD, NVLD, NCHD, NPKD, NSTD )

INTEGER WTYPE  // workstation type
INTEGER ERRIND  // OUT error indicator
INTEGER NLCD  // OUT number of locator devices
INTEGER NSKD  // OUT number of stroke devices
INTEGER NVLD  // OUT number of valuator devices
INTEGER NCHD  // OUT number of choice devices
INTEGER NPKD  // OUT number of pick devices
INTEGER NSTD  // OUT number of string devices

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES to determine the number of available logical input devices for a specified workstation type.

C Input Parameters
type  // Type of workstation. See OPEN WORKSTATION for a list of those available.

C Output Parameters
error_ind
// The error indicator. See the Execution section below for a description of its use.
// See the ERRORS section below for the possible values it may return.

numbers
// A pointer to a Pnum_in data structure in which to return the number of devices available. Pnum_in is defined in phigs.h as follows:

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

modified 2 April 1993
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTYPE</td>
<td>Type of workstation. See <code>OPEN WORKSTATION</code> for a list of those available.</td>
</tr>
</tbody>
</table>

### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error indicator. See the Execution section below for a description of its use.</td>
</tr>
<tr>
<td>NLCDA</td>
<td>The number of LOCATOR devices available with this workstation type.</td>
</tr>
<tr>
<td>NSKDA</td>
<td>The number of STROKE devices available with this workstation type.</td>
</tr>
<tr>
<td>NVLDA</td>
<td>The number of VALUATOR devices available with this workstation type.</td>
</tr>
<tr>
<td>NCHDA</td>
<td>The number of CHOICE devices available with this workstation type.</td>
</tr>
<tr>
<td>NPKDA</td>
<td>The number of PICK devices available with this workstation type.</td>
</tr>
<tr>
<td>NSTD</td>
<td>The number of STRING devices available with this workstation type.</td>
</tr>
</tbody>
</table>

### Execution

`INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES` returns the number of input devices available of each input class for a given workstation type. If no errors are detected, the numbers are returned. If an error is detected, the error indicator will be set to one of the values specified in the ERRORS section below and the input device information will not be returned.

### ERRORS

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **052** Ignoring function, workstation type not recognized by the implementation
- **051** Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- **061** Ignoring function, specified workstation is not category INPUT or category OUTIN

### SEE ALSO

- `INQUIRE WORKSTATION CONNECTION AND TYPE (3P)`
- `INITIALIZE CHOICE (3P)`
- `INITIALIZE LOCATOR (3P)`
- `INITIALIZE PICK (3P)`
- `INITIALIZE STRING (3P)`
- `INITIALIZE STROKE (3P)`
- `INITIALIZE VALUATOR (3P)`

modified 2 April 1993
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED

NAME
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED – inquire number of display priorities supported by a workstation type

SYNOPSIS
C Syntax
void
pinq_num_disp_pris ( wst, error_ind, num_pri )

Pint wst; /* workstation type */
Pint *error_ind; /* OUT error indicator */
Pint *num_pri; /* OUT number of display priorities */

FORTRAN Syntax
SUBROUTINE pqdp ( WTYPE, ERRIND, NSPSUP )

INTEGER WTYPE /* workstation type */
INTEGER ERRIND /* OUT error indicator */
INTEGER NSPSUP /* OUT number of display priorities supported */

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED to determine the number of display priorities supported by a specified workstation type.
If the number of priorities returned is zero, this means the workstation type can support an arbitrary number of priorities.

C Input Parameters
wst Workstation type.

C Output Parameters
error_ind A pointer to the location to store the error number of any error this function detects.
num_pri A pointer to an integer that stores the number of display priorities supported on workstations of type wst.

FORTRAN Input Parameters
WTYPE Workstation type.

FORTRAN Output Parameters
ERRIND The error number of any error detected by this function.
NSPSUP Returns the number of display priorities supported on workstations of type WTYPE.

modified 2 April 1993
ERRORS

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

POST STRUCTURE (3P)
INQUIRE OPEN STRUCTURE – inquire the status of the currently-opened structure

**SYNOPSIS**

**C Syntax**

```c
void pinq_open_struct ( error_ind, status, struct_id )
Pint *error_ind;          \textit{OUT error indicator}
Popen_struct_status *status; @\textit{OUT status of open structure}
Pint *struct_id;          \textit{OUT structure identifier}
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pqopst ( ERRIND, STYPE, STRID )
INTEGER ERRIND \textit{OUT error indicator}
INTEGER STYPE \textit{OUT open structure status (PNONST, POPNST)}
INTEGER STRID \textit{OUT structure identifier}
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use **INQUIRE OPEN STRUCTURE** to determine the status of the currently-open structure.

**C Output Parameters**

- **error_ind**
  - A pointer to the location to store the error number of any error this function detects.

- **status**
  - A pointer to a Popen_struct_status enumerated type in which the system returns the status of the currently-open structure. Values of Popen_struct_status are defined in phigs.h as follows:
    - PSTRUCT_NONE: No structure is open
    - PSTRUCT_OPEN: A structure is open

- **struct_id**
  - An integer pointer in which the system returns the structure identifier of the currently open structure. This value is undefined if **status** is PSTRUCT_NONE.

**FORTRAN Output Parameters**

- **ERRIND**
  - The error number of any error this function detects.

- **STYPE**
  - The status of the currently-open structure, defined in phigs77.h as follows:
    - PNONST: No structure is open
    - POPNST: A structure is open

- **STRID**
  - The structure identifier of the currently open structure. This value is undefined if no structure is open (STYPE = PNONST).

648 modified 2 April 1993
ERRORS 002 Ignoring function, function requires state (PHOP, *, *, *)

SEE ALSO

OPEN STRUCTURE (3P)
CLOSE STRUCTURE (3P)
INQUIRE STRUCTURE STATUS (3P)
INQUIRE STRUCTURE STATE VALUE (3P)
**NAME**
INQUIRE PATHS TO ANCESTORS – obtain ancestors of specified structure

**SYNOPSIS**

**C Syntax**

```c
void
pinq_paths_ances ( struct_id, order, depth, store, error_ind, paths )
```

| Pint | struct_id; | structure identifier |
| Ppath_order | order; | path order |
| Pint | depth; | path depth |
| Pstore | store; | handle to Store object |
| Pint | *error_ind; | OUT error indicator |
| Pelem_ref_list_list | **paths; | OUT structure path list |

**FORTRAN Syntax**

```fortran
SUBROUTINE pqpan ( STRID, PTHORD, PTHDEP, IPTHSZ, N, ERRIND, OL, APTHSZ, PATHS )
```

| INTEGER | STRID | structure identifier |
| INTEGER | PTHORD | path order (PPOTOP, PPOBOT) |
| INTEGER | PTHDEP | path depth |
| INTEGER | IPTHSZ | size of path buffer |
| INTEGER | N | element of list of paths |
| INTEGER | ERRIND | OUT error indicator |
| INTEGER | OL | OUT number of paths available |
| INTEGER | APTHSZ | OUT actual size of the Nth structure path |
| INTEGER | PATHS(2, IPTHSZ) | OUT Nth structure path |

**Required PHIGS Operating States**

( PHOP, *, *, * )

**DESCRIPTION**

**Purpose**
INQUIRE PATHS TO ANCESTORS determines the path or paths in the Central Structure Store that reference the specified structure.

**C Input Parameters**
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by `paths`.

- `struct_id`: Identifier of the structure to be inquired for ancestors.
- `order`: Order in which the path lists are to be returned. This is an enumerated type defined in phigs.h to have the following values:
  - PORDER_TOP_FIRST: Top first
INQUIRE PATHS TO ANCESTORS (3P)

**PORDER_BOTTOM_FIRST**  
*Bottom first*

- **depth**  
  Depth (maximum number of references) of path lists to return.

- **store**  
  The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

**C Output Parameters**

- **error_ind**  
  A pointer to the location to store the error number of any error detected by this function.

- **paths**  
  A pointer to a pointer to a Pelem_ref_list_list structure that contains a list of the specified structure’s structure path lists. Pelem_ref_list_list is defined in phigs.h as follows:

  ```c
  typedef struct{
      Pint num_elem_ref_lists; /* number of execute reference lists */
      Pelem_ref_list *ex_ref_list; /* list of execute reference lists */
  } Pelem_ref_list_list;
  ```

  The `num_elem_ref_lists` component specifies the number of structure path lists, or execute reference lists. The `ex_ref_list` component is a pointer to a list, `num_elem_ref_lists` long, of Pelem_ref_list structures, containing the specified structure’s execute reference lists. Pelem_ref_list is defined in phigs.h as follows:

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

- **elem_refs** is a pointer to a list `num_elem_refs` long of Pelem_ref structures containing the structure identifier and element number of each execute reference structure element in the execute reference list. Pelem_ref is defined in phigs.h as follows:

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

**FORTRAN Input Parameters**

- **STRID**  
  Identifier of the structure to be inquired for ancestors.

- **PTHORD**  
  Order in which the structure path is to be returned. Valid values (defined in modified 2 April 1993
INQUIRE PATHS TO ANCESTORS (3P) SunPHIGS Release 3.0

phigs77.h) are:

<table>
<thead>
<tr>
<th>PPOTOP</th>
<th>Top first</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPOBOT</td>
<td>Bottom first</td>
</tr>
</tbody>
</table>

**PTHDEP**
Depth (maximum number of references) of the structure path to return.

**IPTHSZ** Size of the PATHS array in which the returned structure path data will be stored.
If this value is smaller than the actual size of the structure path (APTHSZ), no data will be returned in the PATHS array, but APTHSZ will be set to indicate the array size required. If you call this function with an array size of zero, APTHSZ is returned with the required array size. Error 2001 is returned if IPTHSZ is too small, but not if it is zero.

**N** List element of structure paths list to return; only one structure path may be inquired upon per subroutine call. If a value of 0 is used here, no structure path data will be returned, but the total number of structure paths will be returned in OL.

### FORTRAN Output

#### Parameters

<table>
<thead>
<tr>
<th>ERRIND</th>
<th>The error number of any error this function detects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>OL</td>
<td>The total number of structure paths for this structure identifier.</td>
</tr>
<tr>
<td>APTHSZ</td>
<td>The number of structure path elements returned in PATHS.</td>
</tr>
<tr>
<td>PATHS</td>
<td>A 2 × IPTHSZ integer array containing the Nth structure path for the specified structure, where the (1,<em>) components contain the structure identifiers, and the (2,</em>) components contain the element sequence numbers.</td>
</tr>
</tbody>
</table>

#### Execution
When INQUIRE PATHS TO ANCESTORS is called, structure path list is filled with list(s) identifying the EXECUTE STRUCTURE structure elements which refer to structure identifier, in the order of traversal. These EXECUTE STRUCTURE structure elements are represented as (structure identifier, element position) pairs, giving the parent structure identifier and the position of the EXECUTE STRUCTURE structure element. Whenever structure identifier itself is included at the bottom of a returned path of ancestors, it is represented by a (structure identifier, element position) pair with an element position of 0. path order and path depth are used to determine the portion of each path returned. The number of references returned in each path is specified by path depth; a path depth of 0 returns all the references in the path. In case of truncation, path order determines whether the head (TOP_FIRST) or the tail (BOTTOM_FIRST) portion of a path is returned. If a path truncation results in two or more partial paths with the same set of element references, only one of the identical path portions is returned.

For example, specifying TOP_FIRST and a depth of 0 returns all paths to struct_id. Specifying TOP_FIRST and a depth of 1 returns the root of each structure network that references struct_id. Specifying BOTTOM_FIRST and a depth of 2 returns all the parents of struct_id.

652 modified 2 April 1993
Errors

002  Ignoring function, function requires state (PHOP, *, *, *)
201  Ignoring function, the specified structure does not exist
207  Ignoring function, the specified path depth is less than zero
2001 FORTRAN: Ignoring function, output parameter size insufficient — A FORTRAN array or string being passed as an output parameter is too small to contain the returned information.
2002 FORTRAN: Ignoring function, list or set element not available — for a non-empty list or set, a value less than zero or greater than the size of the list or set was passed as the requested list or set element in an inquiry routine.

See Also

EXECUTE STRUCTURE (3P)
INQUIRE PATHS TO DESCENDANTS (3P)
NAME
INQUIRE PATHS TO DESCENDANTS – obtain descendants of specified structure

SYNOPSIS
C Syntax
void
pinq_paths_descs ( struct_id, order, depth, store, error_ind, paths )
Pint struct_id; structure identifier
Ppath_order order; path order
Pint depth; path depth
Pstore store; handle to Store object
Pint *error_ind; OUT error indicator
Pelem_ref_list_list **paths; OUT structure path list

FORTRAN Syntax
SUBROUTINE pqpde ( STRID, PTHORD, PTDHP, IPTHSZ, N, ERRIND, OL, APTHSZ, PATHS )
INTEGER STRID structure identifier
INTEGER PTHORD path order (PPOTOP, PPOBOT)
INTEGER PTDHP path depth
INTEGER IPTHSZ size of path buffer
INTEGER N element of list of paths
INTEGER ERRIND OUT error indicator
INTEGER OL OUT number of paths available
INTEGER APTHSZ OUT actual size of the Nth structure path
INTEGER PATHS(2, IPTHSZ) OUT Nth structure path

Required PHIGS
Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE PATHS TO DESCENDANTS to determine the path or paths in the Central Structure Store that are referenced by the specified structure.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by paths.

struct_id
Identifier of the structure to be inquired for descendants.

order Order in which the path lists are to be returned. This is an enumerated type defined in phigs.h to have the following values:
PORDER_TOP_FIRST Top first

654 modified 2 April 1993
INQUIRE PATHS TO DESCENDANTS (3P)

PORDER_BOTTOM_FIRST  Bottom first

depth  Depth (maximum number of references) of path lists to return.
store  The memory buffer PHIGS is to use for storing the information returned. This
       buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind  A pointer to the location to store the error number of any error this function
detects.
paths  A pointer to a pointer to a structure Pelem_ref_list_list, containing a list of the
specification structure’s structure path lists. Pelem_ref_list_list is defined in phigs.h
as follows:

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

The num_elem_ref_lists component specifies the number of structure path lists or
execute reference lists. The elem_ref_lists component is a pointer to a list,
num_elem_ref_lists long, of Pelem_ref_list structures containing the specified
structure’s execute reference lists. Pelem_ref_list is defined in phigs.h as follows:

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

elem_refs is a pointer to a list num_elem_refs long of Pelem_ref structures
containing the structure identifier and element number of each execute reference
structure element in the execute reference list. Pelem_ref is defined in phigs.h as
follows:

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

FORTRAN Input Parameters

STRID  Identifier of the structure to be inquired upon for descendants.

PTHORD  Order in which the structure path is to be returned. Valid values (defined in
phigs77.h) are:

PPOTOP  Top first
PPOBOT  Bottom first

modified 2 April 1993
INQUIRE PATHS TO DESCENDANTS (3P)  SunPHIGS Release 3.0

**Parameters**

- **PTHDEP**: Depth (maximum number of references) of the structure path to return.
- **IPTHSZ**: Size of the PATHS array in which the returned structure path data will be stored. If this value is smaller than the actual size of the structure path (APTHSZ), no data will be returned in the PATHS array, but APTHSZ will be set to indicate the array size required. If you call this function with an array size of zero, APTHSZ is returned with the required array size. Error 2001 is returned if IPTHSZ is too small, but not if it’s zero.
- **N**: List element of structure paths list to return; only one structure path can be queried upon per subroutine call. If a value of 0 is used here, no structure path data will be returned, but the total number of structure paths will be returned in OL.

**FORTRAN Output Parameters**

- **ERRIND**: The error number of any error this function detects.
- **OL**: The total number of structure paths for this structure identifier.
- **APTHSZ**: The number of structure path elements returned in PATHS.
- **PATHS**: A $2 \times IPTHSZ$ integer array containing the Nth structure path for the specified structure, where the $(1,*)$ components contain the structure identifiers, and the $(2,*)$ components contain the element sequence numbers.

**Execution**

When INQUIRE PATHS TO DESCENDANTS is called, structure path list is filled with list(s) identifying the EXECUTE STRUCTURE structure elements which are referenced by structure identifier, in the order of traversal. These EXECUTE STRUCTURE structure elements are represented as (structure identifier, element position) pairs, giving the parent structure identifier and the position of the EXECUTE STRUCTURE structure element. The bottom-most element of a structure network, if included in a returned path, is indicated by a (structure identifier, element position) pair containing the identifier of the bottom-most structure and an element position of 0. path order and path depth are used to determine the portion of each path returned. The number of references returned in each path is specified by path depth; a path depth of 0 returns all the references in the path. In case of truncation, path order determines whether the head (TOP_FIRST) or the tail (BOTTOM_FIRST) portion of a path is returned. If a path truncation results in two or more partial paths with the same set of element references, only one of the identical path portions is returned.

For example, specifying TOP_FIRST and a depth of 0 returns all paths from struct_id. Specifying TOP_FIRST and a depth of 1 returns each EXECUTE STRUCTURE structure element in struct_id as a separate path list. Specifying BOTTOM_FIRST and a depth of 1 returns all the bottom-most structures of the network.

**ERRORS**

- **002**: Ignoring function, function requires state (PHOP, *, *, *)
- **201**: Ignoring function, the specified structure does not exist
- **207**: Ignoring function, the specified path depth is less than zero
- **2001**: FORTRAN: Ignoring function, output parameter size insufficient — A FORTRAN array or string being passed as an output parameter is too small to contain the
returned information.

2002  *FORTRAN*: Ignoring function, list or set element not available — for a non-empty list or set, a value less than zero or greater than the size of the list or set was passed as the requested list or set element in an inquiry routine.

**SEE ALSO**

EXECUTE STRUCTURE (3P)

INQUIRE PATHS TO ANCESTORS (3P)
NAME
 INQUIRE PATTERN FACILITIES – inquire pattern facilities available on a workstation type

SYNOPSIS
 void
 pinq_pat_facs ( type, error_ind, predefined )
 Pint type;
 Pint *error_ind;  OUT error indicator
 Pint *predefined;  OUT number of predefined pattern indices

FORTRAN Syntax
 SUBROUTINE pqpaf ( WTYPE, ERRIND, NPPAI )
 INTEGER WTYPE  workstation type
 INTEGER ERRIND  OUT error indicator
 INTEGER NPPAI  OUT number of predefined pattern indices

Required PHIGS Operating States
 (PHOP, *, *, *)

DESCRIPTION
 Purpose
 Use INQUIRE PATTERN FACILITIES to determine the number of predefined pattern indices available on a specified workstation type.

C Input Parameters
 type  Workstation type.

C Output Parameters
 error_ind
 A pointer to the location to store the error number of any error this function detects.

 predefined
 A pointer to an integer that returns the number of predefined pattern indices available on workstations of type type.

FORTRAN Input Parameters
 WTYPE  Workstation type.

FORTRAN Output Parameters
 ERRIND  The error number of any error this function detects.
 NPPAI  Returns the number of predefined pattern indices available on workstations of type WTYPE.

ERRORS
 002 Ignoring function, function requires state (PHOP, *, *, *)
 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
 052 Ignoring function, workstation type not recognized by the implementation

modified 2 April 1993
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>062</td>
<td>Ignoring function, this information is not available for this MO workstation type</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET PATTERN REPRESENTATION (3P)
- INQUIRE PATTERN REPRESENTATION (3P)
INQUIRE PATTERN REPRESENTATION

NAME

INQUIRE PATTERN REPRESENTATION – inquire for pattern representation on workstation

SYNOPSIS

C Syntax

```c
void pinq_pat_rep ( ws, index, type, store, error_ind, rep, );
```

- `Pint ws;`  // workstation identifier
- `Pint index;`  // pattern index
- `Pinq_type type;`  // type of returned value
- `Pstore store;`  // handle to Store object
- `Pint *error_ind;`  // OUT error indicator
- `Ppat_rep **rep;`  // OUT pattern representation

FORTRAN Syntax

```fortran
SUBROUTINE pqpar ( WKID, PAI, TYPE, DIMX, DIMY, ERRIND, DX, DY, COLIA )
```

- `INTEGER WKID`  // workstation identifier
- `INTEGER PAI`  // pattern index
- `INTEGER TYPE`  // type of returned values (PSET, PREALI)
- `INTEGER DIMX, DIMY`  // maximum pattern array dimensions
- `INTEGER ERRIND`  // OUT error indicator
- `INTEGER DX, DY`  // OUT pattern array dimensions
- `INTEGER COLIA(DIMX, DIMY)`  // OUT pattern array

Required PHIGS

Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE PATTERN REPRESENTATION to determine the pattern representation on a specified workstation for a given pattern index.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by `rep`.

- `ws`  // Workstation identifier.
- `index`  // Entry to be returned from the workstation’s table of pattern representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for pattern index 1 is returned.
- `type`  // An enumerated value specifying whether the queried values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be
mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

```
PINQ_SET Return application-specified value
PINQ_REALIZED Return value available on the workstation
```

`store` The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error this function detects.

- **rep**
  A pointer to a pointer to a Ppat_rep structure in which the system returns the pattern representation at index in the workstation table of pattern representations. Ppat_rep is defined in phigs.h as follows:

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

The Pint_size structure used to define the pattern dimensions is defined in phigs.h as follows:

```c
typedef struct {
    Pint size_x; /* dimension (number of divisions) along X */
    Pint size_y; /* dimension (number of divisions) along Y */
} Pint_size;
```

The `colr_array` component is a pointer to an array of the colour indices defining the pattern, of the dimensions defined by the `dims` component.

### FORTRAN Input Parameters

- **WKID**
  Workstation identifier.

- **PAI**
  Entry to be returned from the workstation table of pattern representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for pattern index 1 is returned.

- **TYPE**
  An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs77.h as:

```c
PSET Return application-specified value
PREALI Return value available on the workstation
```

- **DIMX**
  The x dimension of the COLIA array in which the requested pattern representation is to be returned. If this value is smaller than the actual x dimension of the pattern representation to be returned (DX), no data will be returned in the COLIA

modified 2 April 1993
array, but DX and DY will be set to indicate the array size required.

*DIMY* The y dimension of the COLIA array in which the requested pattern representation is to be returned. If this value is smaller than the actual y dimension of the pattern representation to be returned (DY), no data will be returned in the COLIA array, but DX and DY will be set to indicate the array size required.

If you call this function with both dimensions set to zero, DX and DY will be set to indicate the array size required. Error 2001 will be returned if either dimension is too small, but not if both are zero.

**FORTRAN Output**

**Parameters**

*ERRIND* The error number of any error this function detects.

*DX* The x dimension of the pattern representation returned in COLIA.

*DY* The y dimension of the pattern representation returned in COLIA.

*COLIA* An array of integers in which the system returns the pattern representation at PAI in the workstation table of pattern representations.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

112 Ignoring function, the pattern index value is less than one

101 Ignoring function, the specified representation has not been defined

109 Ignoring function, interior style PATTERN is not supported on the workstation

**SEE ALSO**

INQUIRE LIST OF PATTERN INDICES (3P)

SET PATTERN REPRESENTATION (3P)

INQUIRE PREDEFINED PATTERN REPRESENTATION (3P)

INQUIRE PATTERN REPRESENTATION PLUS (3PP)
NAME  | INQUIRE PHIGS FACILITIES—query the current list of PHIGS facilities defined in the PHIGS description table

SYNOPSIS

C Syntax

```c
void
pinq_phigs_facs ( length, start, error_ind, open_ws, open_ar, num_names, char_sets,
                 length_list, norm_filt, inv_filt )
```

- `Pint length;`: length of application list
- `Pint start;`: starting position
- `Pint *error_ind;`: OUT error indicator
- `Pint *open_ws;`: OUT max. num. simultaneously open ws
- `Pint *open_ar;`: OUT max. num. simultaneously open archive files
- `Pint *num_names;`: OUT number of available names for name sets
- `Pint_list *char_sets;`: OUT list of character sets
- `Pint *length_list;`: OUT length of list in PHIGS
- `Pint *norm_filt;`: OUT maximum length of norm filter list for ISS
- `Pint *inv_filt;`: OUT maximum length of inverted filter list for ISS

FORTRAN Syntax

```fortran
SUBROUTINE pqphf ( NCS, ERRIND, SIMOPW, SIMOPA, NAMESN, OLCS, CS,
                    NFLN, IFLN )
```

- `INTEGER NCS`: character set requested
- `INTEGER ERRIND`: OUT error indicator
- `INTEGER SIMOPW`: OUT maximum number of simultaneously-open workstations
- `INTEGER SIMOPA`: OUT maximum number of simultaneously-open archive files
- `INTEGER NAMESN`: OUT maximum number of available names for name sets
- `INTEGER OLCS`: OUT number of available character sets
- `INTEGER CS`: OUT NCSth available character set
- `INTEGER NFLN`: OUT maximum length of normal filter list for ISS
- `INTEGER IFLN`: OUT maximum length of inverted filter list for ISS

Required PHIGS Operating States

(Phop, *, *, *)

DESCRIPTION

Purpose

Use INQUIRE PHIGS FACILITIES to obtain a list of the available PHIGS facilities from the PHIGS description table.

C Input Parameters

- `length`: The number of list elements for which the application has allocated memory in the output parameter `char_sets->ints`. 0 may be specified, in order to have `length_list` return the total number of elements in the list of character sets.
- `start`: The starting position in the list at which the inquiry is to begin.

modified 2 April 1993
**C Output Parameters**

- **error_ind**
  A pointer to the location to store the error number of any error this function detects.

- **open_ws**
  A pointer to an integer that returns the maximum number of simultaneously-open workstations supported.

- **open_ar**
  A pointer to an integer that returns the maximum number of simultaneously-open archive files supported.

- **num_names**
  A pointer to an integer that returns the number of available names for name sets.

- **char_sets**
  A pointer to a data structure that returns a portion of the list of available character sets, starting with the `start` number item. `Pint_list` is defined in `phigs.h` as follows:

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

  Prior to calling this function, the `ints` field of the `Pint_list` structure must contain a pointer to an application supplied buffer. This buffer must be at least as large as the `length` parameter.

- **length_list**
  A pointer to an integer that returns the length of the list in PHIGS.

- **norm_filt**
  A pointer to an integer that returns the maximum length of the norm filter list for INCREMENTAL SPATIAL SEARCH.

- **inv_filt**
  A pointer to an integer that returns the maximum length of the inverted filter list for INCREMENTAL SPATIAL SEARCH.

---

**FORTRAN Input Parameters**

- **NCS**
  The number of the list element desired from the list of character sets.

---

**FORTRAN Output Parameters**

- **SIMOPW**
  The maximum number of simultaneously-open workstations supported.

- **SIMOPA**
  The maximum number of simultaneously-open archive files supported.

- **OLCS**
  The number of available character sets.

- **CS**
  The NCSth list element from the list of character sets.

- **NFLN**
  The maximum length of the normal filter list for INCREMENTAL SPATIAL SEARCH.

---

664 modified 2 April 1993
IFLN  The maximum length of the inverted filter list for INCREMENTAL SPATIAL SEARCH.

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

SEE ALSO
PHIGS DESCRIPTION TABLE (7P)
NAME
INQUIRE PICK DEVICE STATE – inquire state of a pick device

SYNOPSIS
C Syntax

```c
void
pinq_pick_st ( ws, dev, type, store, err, op_mode, echo_switch, filter, init_status,
    init_pick, prompt_echo, echo_area, path_data, pick_order )
```

<table>
<thead>
<tr>
<th>Pint</th>
<th>ws;</th>
<th>workstation identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pint</td>
<td>dev;</td>
<td>pick device number</td>
</tr>
<tr>
<td>Pinq_type</td>
<td>type;</td>
<td>type of returned value</td>
</tr>
<tr>
<td>Pstore</td>
<td>store;</td>
<td>handle to store object</td>
</tr>
<tr>
<td>Pint</td>
<td>*err;</td>
<td>OUT error indicator</td>
</tr>
<tr>
<td>Pop_mode</td>
<td>*op_mode;</td>
<td>OUT operating mode</td>
</tr>
<tr>
<td>Pecho_switch</td>
<td>*echo_switch;</td>
<td>OUT echo switch</td>
</tr>
<tr>
<td>PfILTER</td>
<td>**filter;</td>
<td>OUT pick filter</td>
</tr>
<tr>
<td>Pin_status</td>
<td>*init_status;</td>
<td>OUT initial pick status</td>
</tr>
<tr>
<td>Ppick_path</td>
<td>**init_pick;</td>
<td>OUT initial pick path</td>
</tr>
<tr>
<td>Pint</td>
<td>*prompt_echo;</td>
<td>OUT prompt/echo type</td>
</tr>
<tr>
<td>Plimit</td>
<td>*echo_area;</td>
<td>OUT echo area</td>
</tr>
<tr>
<td>Ppick_data</td>
<td>**pick_data;</td>
<td>OUT data record</td>
</tr>
<tr>
<td>Ppath_order</td>
<td>*path_order;</td>
<td>OUT path order</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

```fortran
SUBROUTINE pqpks ( WKID, PKDNR, TYPE, MLDR, IPISSZ, IPRESSZ, IPPSZ,
    ERRIND, MODE, ESW, PISSZ, PINS, PESSZ, PES, ISTAT, PPD, PP, PET,
    EAREA, LDR, DATREC, PPORDR )
```

<table>
<thead>
<tr>
<th>INTEGER</th>
<th>WKID</th>
<th>workstation identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>PKDNR</td>
<td>pick device number</td>
</tr>
<tr>
<td>INTEGER</td>
<td>TYPE</td>
<td>type of returned values (PSET, PREALI)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>MLDR</td>
<td>dimension of data record array</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IPISSZ</td>
<td>pick inclusion set buffer size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IPRESSZ</td>
<td>pick exclusion set buffer size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IPPSZ</td>
<td>pick path buffer size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>ERRIND</td>
<td>OUT error indicator</td>
</tr>
<tr>
<td>INTEGER</td>
<td>MODE</td>
<td>OUT operating mode (PREQU, PSAMPL, PEVENT)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>ESW</td>
<td>OUT echo switch (PNECHO, PECHO)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PISSZ</td>
<td>OUT pick inclusion set size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PINS(PISSZ)</td>
<td>OUT pick inclusion set</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PESSZ</td>
<td>OUT pick exclusion set size</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PES(IPRESSZ)</td>
<td>OUT pick exclusion set</td>
</tr>
<tr>
<td>INTEGER</td>
<td>ISTAT</td>
<td>OUT initial status (POK, PNICK)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PPD</td>
<td>OUT pick path depth</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PP(3, IPPSZ)</td>
<td>OUT pick path</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PET</td>
<td>OUT prompt/echo type</td>
</tr>
<tr>
<td>REAL</td>
<td>EAREA(4)</td>
<td>OUT echo area in Device Coordinates</td>
</tr>
</tbody>
</table>

modified 2 April 1993
### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

#### Purpose
Use INQUIRE PICK DEVICE STATE to determine the current state of the specified pick device.

#### C Input Parameters

- **ws**: Workstation identifier. An integer specifying the workstation with which the specified pick device is associated.
- **dev**: The device number of the pick device. See the AVAILABLE DEVICES section of INITIALIZE PICK for a description of the available devices.
- **type**: An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

  ```c
typedef enum {
    PINQ_SET,
    PINQ_REALIZED
  } Pinq_type;
```
- **store**: The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

#### C Output Parameters

- **err**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.
- **op_mode**: The operating mode. Pop_mode is an enumerated type with the following values:

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

---

### OUT

- **LDR**: INTEGER
  - OUT number of array elements used in data record
- **DATREC(MLDR)**: CHARACTER∗80
  - OUT data record
- **PPORDR**: INTEGER
  - OUT pick path order (PPOTOP, PPOBOT)

---

modified 2 April 1993
**echo_switch**
The echo state. `Pecho_switch` is an enumerated type with the following values:

- `PSWITCH_ECHO`
- `PSWITCH_NO_ECHO`

**filter**
A pointer to a pointer to the location to store the device’s detectability filter. `Pfilter` is defined in `phigs.h` as follows:

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

`Pint_list` is defined in `phigs.h` as follows:

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

The `num_ints` component specifies the number of elements in the list. The `ints` component is a pointer to a list `num_ints` long.

**init_status**
A pointer to the initial pick status. `Pin_status` is an enumerated type with the following values:

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

**init_pick**
A pointer to a pointer that points to the initial pick path data structure contained with the store. `Ppick_path` is defined in `phigs.h` as follows:

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

`Ppick_path_elem` is defined in `phigs.h` as follows:

```c
typedef struct {
    Pint struct_id; /* structure identifier */
    Pint pick_id; /* hierarchical pick identifier */
    Pint elem_pos; /* element sequence number */
} Ppick_path_elem;
```
The prompt/echo type desired. See the AVAILABLE DEVICES Section of INITIALIZE PICK for a description of the available types.

**echo_area**

A pointer to a variable of type Plimit that contains the echo area of the device. Plimit is defined in phigs.h as follows:

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

**pick_data**

Pointer to a variable that points to the pick device state within store. Ppick_data is defined in phigs.h as follows:

```c
typedef struct {
    union Ppick_pets {
        struct Pick_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pick_pet_r2 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_r2;
        struct Pick_pet_r3 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_r3;
        struct Pick_pet_u1 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_u1;
        struct Pick_pet_u2 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_u2;
    }
} Ppick_data;
```
struct Pick_pet_u3 {
    Pint highl_colr;
    Pint highl_count;
    Pfloat highl_duration;
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u3;

struct Pick_pet_u4 {
    Pint highl_colr;
    Pint highl_count;
    Pfloat highl_duration;
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u4;

} pets;

} Pick_data;

path_order

Pointer to the path order. Ppath_order is an enumerated type with the following values:
typedef enum {
    PORDER_TOP_FIRST,
    PORDER_BOTTOM_FIRST
} Ppath_order;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, and all the data not in the data record. Error 2001 is returned if MLDR is too small, but not if it is zero.

INQUIRE PICK DEVICE STATE returns a number of variable length lists. Parameters to the function exist for both the allocated size of the lists and the actual size needed to return all the data. If any of the allocated sizes are too small to accept the entire list, the corresponding list will not be returned. The required size of the list, however, will be returned. The error indicator will be set to error 2001 in such a case.

WKID The workstation identifier of the workstation associated with the device.

PKDNR The device number of the PICK device. See the AVAILABLE DEVICES section of INITIALIZE PICK for a description of the available devices.

TYPE An enumerated value specifying whether the values to be returned are those
originally specified by the application (Set), or those resulting after PHIGS mapped them to ones available on the workstation (Realized). Valid values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PSET Set</td>
</tr>
<tr>
<td>1</td>
<td>PREALI Realized</td>
</tr>
</tbody>
</table>

**FORTRAN Output Parameters**

- **ERRIND**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.
- **MODE**: The operating mode.
- **ESW**: The echo switch.
- **PISSZ**: The actual pick inclusion set size.
- **PINS**: The pick filter inclusion set.
- **PESSZ**: The actual pick inclusion set size.
- **PES**: The pick filter inclusion set.
- **MLDR**: The dimension of the data record array, DATREC.
- **ISTAT**: The initial pick status.
- **PPD**: The initial pick path depth.
- **PP**: The initial pick path.
- **PET**: The prompt/echo type.
- **EAREA**: An array containing the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.
- **LDR**: The required dimension of the data record array, DATREC.
- **DATREC**: The data record array.
- **PPORDR**: The pick path order.

**Execution**

INQUIRE PICK DEVICE STATE returns the current state of the specified pick device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial pick, prompt/echo type, echo area, pick path order, data record, and pick filter. See SET PICK MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE PICK for a description of the initial pick, prompt/echo type, echo area, pick path order, and data record contents and how to set these values. See SET PICK FILTER for a description of the pick filter and how to set it.
Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the queried information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td>FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- INITIALIZE PICK (3P)
- SET PICK MODE (3P)
- SET PICK FILTER (3P)
- INQUIRE PICK DEVICE STATE 3 (3P)
NAME
INQUIRE PICK DEVICE STATE 3 – inquire state of a pick device

SYNOPSIS
C Syntax

```c
void pinq_pick_st3 ( ws, dev, type, store, err, op_mode, echo_switch, filter, init_status,
        init_pick, prompt_echo, echo_vol, path_data, pick_order )
```

FORTRAN Syntax

```
SUBROUTINE pqpks3 ( WKID, PKDNR, TYPE, MLDR, IPISSZ, IPESSZ, IPPSZ,
        ERRIND, MODE, ESW, PISSZ, PINS, PESSZ, PES, ISTAT, PPD, PP, ISTAT,
        PDD, PP, PET, EVOL, LDR, DATREC, PPORDR )
```

modified 2 April 1993
INQUIRE PICK DEVICE STATE 3 (3P)  

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

#### Purpose

Use INQUIRE PICK DEVICE STATE 3 to determine the current state of the specified pick device.

### C Input Parameters

- **ws**: Workstation identifier. An integer specifying the workstation with which the specified pick device is associated.
- **dev**: The device number of the pick device. See the AVAILABLE DEVICES section of INITIALIZE PICK for a description of the available devices.
- **type**: An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A `Pinq_type` structure is defined as:

```c
typedef enum {
    PINQ_SET,
    PINQ_REALIZED
} Pinq_type;
```
- **store**: The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

### C Output Parameters

- **err**: The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.
- **op_mode**: The operating mode. Pop_mode is an enumerated type with the following values:

```c
POP_REQ
POP_SAMPLE
POP_EVENT
```

---

modified 2 April 1993
The echo state. `Pecho_switch` is an enumerated type with the following values:

- `PSWITCH_ECHO`
- `PSWITCH_NO_ECHO`

A pointer to a pointer to the location to store the device’s detectability filter. `Pfilter` is defined in `phigs.h` as follows:

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

`Pint_list` is defined in `phigs.h` as follows:

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

A pointer to the initial pick status. `Pin_status` is an enumerated type with the following values:

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

A pointer to a pointer that points to the initial pick path data structure contained with the store. `Ppick_path` is defined in `phigs.h` as follows:

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

`Ppick_path_elem` is defined in `phigs.h` as follows:

```c
typedef struct {
    Pint struct_id; /* structure identifier */
    Pint pick_id; /* hierarchical pick identifier */
    Pint elem_pos; /* element sequence number */
} Ppick_path_elem;
```

The prompt/echo type desired. See the AVAILABLE DEVICES Section of INITIALIZE PICK for a description of the available types.
**echo_vol**

A pointer to a Plimit3 structure defining the \( x, y, \) and \( z \) components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
```

**pick_data**

Pointer to a pointer that points to the pick device state within the store. Pick_data3 is defined in phigs.h as follows:

```c
typedef struct {
    union Ppick_pets {
        struct Pick_pet_r1 {
            Pint unused;
        } pet_r1;
        struct Pick_pet_r2 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_r2;
        struct Pick_pet_r3 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_r3;
        struct Pick_pet_u1 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_u1;
        struct Pick_pet_u2 {
            Pint highl_colr;
            Pint highl_count;
            Pfloat highl_duration;
            Pfloat ap_size; /* aperture size, half-width in DC units */
        } pet_u2;
    }
} Pick_data3;
```
struct Pick_pet_u3 {
    Pint highl_colr;
    Pint highl_count;
    Pfloat highl_duration;
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u3;
struct Pick_pet_u4 {
    Pint highl_colr;
    Pint highl_count;
    Pfloat highl_duration;
    Pfloat ap_size; /* aperture size, half-width in DC units */
} pet_u4;
} pets;
typedef Ppick_data Ppick_data3;

path_order
A pointer to the path order. Ppath_order is an enumerated type with the
following values:

typedef enum {
    PORDER_TOP_FIRST,
    PORDER_BOTTOM_FIRST
} Ppath_order;

FORTRAN Input
Parameters
Applications using the FORTRAN binding must supply a CHARACTER array to this function,
into which will be placed the contents of the device’s input data record. The contents of
the data record are subsequently extracted by the application with the function UNPACK
DATA RECORD. The allocated dimension of the character array is passed in the MLDR
argument. The dimension needed is returned in the LDR argument. The caller can
determine the required dimension by calling this function with MLDR set to zero, in which
case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero,
some values will be returned. These are LDR, and all the data not in the data record.

Error 2001 is returned if MLDR is too small, but not if it is zero.

INQUIRE PICK DEVICE STATE 3 returns a number of variable length lists. Parameters
to the function exist for both the allocated size of the lists and the actual size needed to
return all the data. If any of the allocated sizes are too small to accept the entire list, the
 correponding list will not be returned. The required size of the list, however, will be
returned. The error indicator will be set to error 2001 in such a case.

WKID  The workstation identifier of the workstation associated with the device.

PKDNR  The device number of the PICK device. See the AVAILABLE DEVICES section of
INITIALIZE PICK 3 for a description of the available devices.

modified 2 April 1993
INQUIRE PICK DEVICE STATE 3 (3P) SunPHIGS Release 3.0

**TYPE** An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0  PSET  Set
1  PREALI Realized

**IPISSZ, IPESSZ**
The size of the pick filter inclusion and exclusion set buffers.

**IPPSZ** The pick path buffer size.

**FORTRAN Output Parameters**

**ERRIND** The error indicator. See the *Execution* section below for a description of its use. See the *ERRORS* section below for the possible values it may return.

**MODE** The operating mode.

**ESW** The echo switch.

**PISSZ** The actual pick inclusion set size.

**PINS** The pick filter inclusion set.

**PESSZ** The actual pick inclusion set size.

**PES** The pick filter inclusion set.

**MLDR** The dimension of the data record array, DATREC.

**ISTAT** The initial pick status.

**PPD** The initial pick path depth.

**PP** The initial pick path.

**PET** The prompt/echo type.

**EVAL** An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR** The required dimension of the data record array, DATREC.

**DATREC** The data record array.

**PPORDR** The pick path order.

**Execution** INQUIRE PICK DEVICE STATE 3 returns the current state of the specified pick device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial pick, prompt/echo type, echo volume, pick path order, data record, and pick filter. See SET PICK MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE PICK 3 for a description of the initial pick, prompt/echo type, echo volume, pick path order, and data record contents and how to set these values. See SET PICK FILTER for a description of the pick filter and how to set it.

678 modified 2 April 1993
Except in the cases mentioned in the C and FORTRAN Parameters sections above, if this function detects an error, the error indicator will indicate the error number of the error detected and no other output data will be returned. If the function detects no error, the error indicator will be set to zero and the queried information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 061 Ignoring function, specified workstation is not of category INPUT or OUTIN
- 250 Ignoring function, the specified device is not available on the specified workstation
- 2200 C: Buffer overflow in input or inquiry function
- 2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

**SEE ALSO**

- INITIALIZE PICK 3 (3P)
- SET PICK MODE (3P)
- SET PICK FILTER (3P)
- INQUIRE PICK DEVICE STATE (3P)

modified 2 April 1993
NAME
INQUIRE POLYLINE FACILITIES – obtain list of polyline facilities from workstation description table

SYNOPSIS
C Syntax
void pinq_line_facs ( type, length, start, error_ind, facilities, total_length )
Pint type; workstation type
Pint length; length of application list
Pint start; starting position
Pint *error_ind; OUT error indicator
Pline_facs *facilities; OUT polyline facilities
Pint *total_length; OUT length of list in PHIGS

FORTRAN Syntax
SUBROUTINE pqplf ( WTYPE, N, ERRIND, NLT, LT, NLW, NOMLW, RLWMIN, RLWMAX, NPPLI )
INTEGER WTYPE workstation type
INTEGER N list element requested
INTEGER ERRIND OUT error indicator
INTEGER NLT OUT number of available linetypes
INTEGER LT OUT Nth element of list of available linetypes
INTEGER NLW OUT number of available linewidths
REAL NOMLW OUT nominal linewidth (DC)
REAL RLWMIN, RLWMAX OUT range of linewidths (DC)
INTEGER NPPLI OUT number of predefined polyline indices

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
INQUIRE POLYLINE FACILITIES obtains a list of the polyline facilities supported on the specified type of workstation.

C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Get the polyline facilities for this workstation type.</td>
</tr>
<tr>
<td>length</td>
<td>The number of ints in the facilities→types.ints output parameter for which the application has allocated memory. length is the number of list elements that the system can return in facilities→types.ints. If a value of 0 is used here, no data will be returned in the facilities→types.ints list, but the total number of elements will be returned in total_length.</td>
</tr>
<tr>
<td>start</td>
<td>Starting position of inquiry. The elements in the list, beginning with the item number specified by start, are copied sequentially into facilities→types.ints until facilities→types.ints is full or all the elements have been copied.</td>
</tr>
</tbody>
</table>

680 modified 2 April 1993
C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error that this function detects.

- **facilities**
  A pointer to a structure in which the system returns the portion of the list of polyline facilities from the workstation description table, starting with start. Pline_facs is defined in phigs.h as:
  ```c
  typedef struct {
      Pint_list types;       /* list of line types */
      Pint widths;           /* number of available line widths */
      Pint num_widths;       /* number of available line widths */
      Pfloat nom_width;      /* nominal line width */
      Pfloat min_width;      /* minimum line width */
      Pfloat max_width;      /* maximum line width */
      Pint num_pred_inds;    /* number of predefined bundles */
  } Pline_facs;
  ```
  And Pint_list is defined in phigs.h as:
  ```c
  typedef struct {
      Pint num_ints;         /* number of integers */
      Pint *ints;            /* list of integers */
  } Pint_list;
  ```

Prior to calling this function, the `ints` field of the Pint_list structure must contain a pointer to an application supplied buffer. This buffer must be at least as large as the corresponding `length` parameter.

- **total_length**
  A pointer to an integer in which to return the total length of the list. This is the value required for `length` if all the items in the list are to be returned.

FORTRAN Input Parameters

- **WTYPE**
  Get the polyline facilities for this workstation type.

- **N**
  Get the Nth element from the list of polyline facilities.

FORTRAN Output Parameters

- **ERRIND**
  The error number of any error that this function detects.

- **NLT**
  The number of available linetypes.

- **LT**
  The Nth linetype from the list of available linetypes.

- **NLW**
  The number of available linewidths.

- **LW**
  The Nth linewidth from the list of available linewidths.

- **NOMLW**
  The nominal linewidth, in Device Coordinates (DC).
INQUIRE POLYLINE FACILITIES (3P) SunPHIGS Release 3.0

**RLWMIN**

The minimum linewidth, in DC.

**RLWMAX**

The maximum linewidth, in DC.

**NPPLI**

The number of predefined polyline indices.

**ERRORS**

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

PHIGS WORKSTATION DESCRIPTION TABLE (7P)

---

modified 2 April 1993
NAME
INQUIRE POLYLINE REPRESENTATION – obtain polyline representation on specified workstation

SYNOPSIS
C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pqplr ( WKID, PLI, TYPE, ERRIND, LTYPE, LWIDTH, COLI )
INTEGER WKID    /* workstation identifier */
INTEGER PLI     /* polyline index */
INTEGER TYPE    /* type of returned values (PSET,PREALI) */
INTEGER ERRIND  /* OUT error indicator */
INTEGER LTYPE   /* OUT linetype */
REAL LWIDTH     /* OUT linewidth scale factor */
INTEGER COLI    /* OUT polyline colour index */
```

DESCRIPTION
Purpose
Use INQUIRE POLYLINE REPRESENTATION to determine the current attribute values for a specified entry in a specified workstation’s table of defined polyline representations. See the description of the subroutine SET POLYLINE REPRESENTATION for information about the meaning of these attribute values.

C Input Parameters

- **ws**: Workstation identifier.
- **index**: Entry to be returned from the workstation’s table of polyline representations; if this entry is not present in the table and the **type** parameter is REALIZED, the representation for polyline index 1 is returned.
- **type**: An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

  - **PINQ_SET**: Return application-specified value
  - **PINQ_REALIZED**: Return value available on the workstation

Required PHIGS Operating States

(PHOP, WSOP, *, *)

modiﬁed 2 April 1993

683
### C Output Parameters

<table>
<thead>
<tr>
<th>error_ind</th>
<th>A pointer to the location to store the error number of any error that this function detects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rep</td>
<td>A pointer to a Pline_bundle structure in which the system returns the polyline representation at index in the workstation’s table of polyline representations. Pline_bundle is defined in phigs.h as follows:</td>
</tr>
<tr>
<td></td>
<td>typedef struct { \n</td>
</tr>
<tr>
<td></td>
<td>Values for type are:</td>
</tr>
<tr>
<td></td>
<td>1 PLINE_SOLID</td>
</tr>
<tr>
<td></td>
<td>2 PLINE_DASH</td>
</tr>
<tr>
<td></td>
<td>3 PLINE_DOT</td>
</tr>
<tr>
<td></td>
<td>4 PLINE_DASH_DOT</td>
</tr>
</tbody>
</table>

### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>WKID</th>
<th>Workstation identifier.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLI</td>
<td>Entry to be returned from the workstation’s table of polyline representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for polyline index 1 is returned.</td>
</tr>
<tr>
<td>TYPE</td>
<td>An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs77.h as:</td>
</tr>
<tr>
<td></td>
<td>PSET Return application-specified value</td>
</tr>
<tr>
<td></td>
<td>PREALI Return value available on the workstation</td>
</tr>
</tbody>
</table>

### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>ERRIND</th>
<th>The error number of any error that this function detects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTYPE</td>
<td>The line type at index PLI in the workstation’s table of polyline representations.</td>
</tr>
<tr>
<td>LWIDTH</td>
<td>The line width scale factor at index PLI in the workstation’s table of polyline representations.</td>
</tr>
<tr>
<td>COLI</td>
<td>The colour index at index PLI in the workstation’s table of polyline representations.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>003 Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td></td>
<td>054 Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td></td>
<td>059 Ignoring function, the specified workstation does not have output capability (in</td>
</tr>
</tbody>
</table>

684 modified 2 April 1993
other words, the workstation category is not OUTPUT, OUTIN, or MO)

100 Ignoring function, the bundle index value is less than one
101 Ignoring function, the specified representation has not been defined
134 Ignoring function, the requested entry contains a general colour specification
with colour type other than INDIRECT.

SEE ALSO

SET POLYLINE REPRESENTATION (3P)
INQUIRE LIST OF POLYLINE INDICES (3P)
INQUIRE PREDEFINED POLYLINE REPRESENTATION (3P)
INQUIRE POLYLINE REPRESENTATION PLUS (3PP)
NAME
INQUIRE POLYMARKER FACILITIES – obtain list of workstation polymarker facilities

SYNOPSIS
C Syntax

```c
void
pinq_marker_facs ( type, length, start, error_ind, facilities, total_length )
```

- `type`: workstation type
- `length`: length of application list
- `start`: starting position
- `error_ind`: OUT error indicator
- `facilities`: OUT polymarker facilities
- `total_length`: OUT length of list in PHIGS

FORTRAN Syntax

```fortran
SUBROUTINE pqpmf ( WTYPE, N, ERRIND, NMT, MT, NMS, NOMMS, RMSMIN, RMSMAX, NPPMI )
```

- `WTYPE`: workstation type
- `N`: list element requested
- `ERRIND`: OUT error indicator
- `NMT`: OUT number of available marker types
- `MT`: OUT nth element of list of available marker types
- `NMS`: OUT number of available marker sizes
- `NOMMS`: OUT nominal marker size (DC)
- `RMSMIN`, `RMSMAX`: OUT range of marker sizes (DC)
- `NPPMI`: OUT number of predefined polymarker indices

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose
INQUIRE POLYMARKER FACILITIES obtains a list of the polymarker facilities available on the specified type of workstation.

C Input Parameters

- `type`: Get the polymarker facilities for this workstation type.
- `length`: The number of ints in the `facilities→types` output parameter for which the application has allocated memory. `length` is the number of list elements that the system can return in `facilities→types ints`. If a value of 0 is used here, no data will be returned in the `facilities→types ints` list, but the total number of elements will be returned in `total_length`.
- `start`: Starting position of inquiry. The elements in the list, beginning with the item number specified by `start`, are copied sequentially into `facilities→types ints` until `facilities→types ints` is full or all the elements have been copied.

C Output Parameters

- `error_ind`: A pointer to the location to store the error number of any error that this function detects.

modified 2 April 1993
facilities

A pointer to a structure in which the system returns the portion of the list of polymarker facilities from the workstation description table, starting with start. Pmarker_facs is defined in phigs.h as:

typedef struct {
    Pint_list types;        /* list of marker types */
    Pint num_sizes;         /* number of available marker sizes */
    Pfloat nom_size;        /* nominal marker size */
    Pfloat min_size;        /* minimum marker size */
    Pfloat max_size;        /* maximum marker size */
    Pint num_pred_inds;     /* number of predefined bundles */
} Pmarker_facs;

And Pint_list is defined as:

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

Prior to calling this function, the ints field of the Pint_list structure must contain a pointer to an application supplied buffer. This buffer must be at least as large as the length parameter.

total_length

A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for length if all elements in the list are to be returned.

FORTRAN Input Parameters

WTYPE   Get the polymarker facilities for this workstation type.
N       Get the Nth element from the list of polymarker facilities.

FORTRAN Output Parameters

ERRIND   The error number of any error that this function detects.
NMT      The number of available marker types.
MT       The Nth marker type from the list of available marker types.
NMS      The number of available marker size.
NOMMS    The nominal marker size, in Device Coordinates (DC).
RMSMIN   The minimum marker size, in DC.

modified 2 April 1993
### RMSMAX

The maximum marker size, in DC.

### NPPMI

The number of predefined polymarker indices.

#### ERRORS

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

051  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052  Ignoring function, workstation type not recognized by the implementation

059  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062  Ignoring function, this information is not available for this MO workstation type

#### SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NAME
INQUIRE POLYMARKER REPRESENTATION – obtain polymarker representation on specified workstation

SYNOPSIS
C Syntax
void
pinq_marker_rep ( ws, index, type, error_ind, rep )

Pint ws;  // workstation identifier
Pint index;  // polymarker index
Pinq_type type;  // type of returned value
Pint *error_ind;  // OUT error indicator
Pmarker_bundle *rep;  // OUT polymarker representation

FORTRAN Syntax
SUBROUTINE pqpmr ( WKID, PMI, TYPE, ERRIND, MTYPE, MSZSF, COLI )

INTEGER WKID  // workstation identifier
INTEGER PMI  // polymarker index
INTEGER TYPE  // type of returned values (PSET, PREALI)
INTEGER ERRIND  // OUT error indicator
INTEGER MTYPE  // OUT marker type
REAL MSZSF  // OUT marker size scale factor
INTEGER COLI  // OUT polymarker colour index

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE POLYMARKER REPRESENTATION to determine the current attribute values for a specified entry in a specified workstation’s table of defined polymarker representations. See the description of the subroutine SET POLYMARKER REPRESENTATION for information about the meaning of these attribute values.

C Input Parameters
ws  // Workstation identifier.
index  // Entry to be returned from the workstation’s table of polymarker representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for polymarker index 1 is returned.
type  // An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

PINQ_SET  // Return application-specified value
PINQ_REALIZED  // Return value available on the workstation

modified 2 April 1993
### C Output Parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>error_ind</code></td>
<td>A pointer to the location to store the error number of any error that this function detects.</td>
</tr>
<tr>
<td><code>rep</code></td>
<td>A pointer to a <code>pmarker_bundle</code> structure in which the system returns the polymarker representation at <code>index</code> in the workstation’s table of polymarker representations. <code>pmarker_bundle</code> is defined in <code>phigs.h</code> as follows:</td>
</tr>
<tr>
<td></td>
<td><code>typedef struct {</code></td>
</tr>
<tr>
<td></td>
<td><code>Pint type; /* marker type */</code></td>
</tr>
<tr>
<td></td>
<td><code>Pfloat size; /* marker size scale factor */</code></td>
</tr>
<tr>
<td></td>
<td><code>Pint colr_ind; /* colour index */</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code> <code>pmarker_bundle;</code></td>
</tr>
</tbody>
</table>

Values for `type` are:

1. `PMARKER_DOT`
2. `PMARKER_PLUS`
3. `PMARKER_ASTERISK`
4. `PMARKER_CIRCLE`
5. `PMARKER_CROSS`

### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WKID</code></td>
<td>Workstation identifier.</td>
</tr>
<tr>
<td><code>PMI</code></td>
<td>Entry to be returned from the workstation’s table of polymarker representations; if this entry is not present in the table and the <code>type of returned value</code> parameter is <code>REALIZED</code>, the representation for polymarker index 1 is returned.</td>
</tr>
<tr>
<td><code>TYPE</code></td>
<td>An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (<code>SET</code>), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (<code>REALIZED</code>). Valid values are defined in <code>phigs.h</code> as:</td>
</tr>
<tr>
<td></td>
<td><code>PSET Return application-specified value</code></td>
</tr>
<tr>
<td></td>
<td><code>PREALI Return value available on the workstation</code></td>
</tr>
</tbody>
</table>

### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ERRIND</code></td>
<td>The error number of any error that this function detects.</td>
</tr>
<tr>
<td><code>MTYPE</code></td>
<td>The marker type at index <code>PMI</code> in the workstation’s table of polymarker representations.</td>
</tr>
<tr>
<td><code>MSZSF</code></td>
<td>The marker size scale factor at index <code>PMI</code> in the workstation’s table of polymarker representations.</td>
</tr>
<tr>
<td><code>COLI</code></td>
<td>The marker colour index at index <code>PMI</code> in the workstation’s table of polymarker representations.</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>003 Ignoring function, function requires state (<code>PHOP, WSOP, *, *</code>)&lt;br&gt;054 Ignoring function, the specified workstation is not open</td>
</tr>
</tbody>
</table>

690 modified 2 April 1993
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>100</td>
<td>Ignoring function, the bundle index value is less than one</td>
</tr>
<tr>
<td>101</td>
<td>Ignoring function, the specified representation has not been defined</td>
</tr>
<tr>
<td>134</td>
<td>Ignoring function, the requested entry contains a general colour specification with <em>colour type</em> other than INDIRECT.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET POLYMARKER REPRESENTATION (3P)
- INQUIRE LIST OF POLYMARKER INDICES (3P)
- INQUIRE PREDEFINED POLYMARKER REPRESENTATION (3P)
- INQUIRE POLYMARKER REPRESENTATION PLUS (3PP)
INQUIRE POSTED STRUCTURES – obtain list of structures posted to workstation

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqpost ( WKID, N, ERRIND, NUMBER, STRID, PRIORT )
INTEGER WKID  /* workstation identifier */
INTEGER N  /* list element requested */
INTEGER ERRIND  /* OUT number of structures posted to that workstation */
INTEGER NUMBER  /* OUT number of structures posted to that workstation */
INTEGER STRID  /* OUT identifier of the Nth structure posted to that workstation */
REAL PRIORT  /* OUT display priority of the Nth structure posted to that workstation */
```

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE POSTED STRUCTURES to obtain the list of currently posted structures from a workstation’s state list.

Although the PHIGS standard does not require it, SunPHIGS orders the inquired list of posted structures by increasing display priority. If two structures are posted with the same priority, the structure posted (or reposted) last is sorted last.

**C Input Parameters**

- **ws** The workstation identifier of the workstation whose state list is queried.
- **length** The number of postings in the list output parameter for which the application has allocated memory. `length` is the number of list elements (posted structures) that the system can return in `list→postings`. If a value of 0 is used here, no data will be returned in the `list→postings` list, but the total number of posted structures in the PHIGS state list will be returned in `total_length`.
- **start** Starting position of inquiry into the PHIGS state list of current structures. The elements of the list of posted structures, beginning with the item number specified by `start`, are copied sequentially into `list→postings` until `list→postings` is full or all the posted structures have been copied.

692 modified 2 April 1993
C Output Parameters

- **error_ind**: A pointer to the location to store the error number of any error that this function detects.

- **list**: A pointer to a Pposted_struct_list in which the system returns the portion of the list of currently posted structures starting at the entry specified with `start`. 
  
Pposted_struct_list is defined in phigs.h as follows:

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

  The pointer `list->postings` must be initialized to an array of `length` Pstructpost elements. Pposted_struct is defined in phigs.h as follows:

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

- **total_length**: A pointer to an integer in which the system returns the total number of elements in the PHIGS state list of currently used posted structures. This is the value required for `length` if all posted structures are to be returned.

FORTRAN Input Parameters

- **WKID**: The workstation identifier of the workstation whose state list is queried.

- **N**: Position in the list of the item requested. The Nth in the list of posted structures will be returned in STRID.

FORTRAN Output Parameters

- **ERRIND**: The error number of any error that this function detects.

- **NUMBER**: The total number of structures currently posted i to the workstation.

- **STRID**: The Nth in the list of posted structures on workstation WKID.

- **PRIORT**: The display priority with which STRID is currently posted on workstation WKID.

ERRORS

- **003**: Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054**: Ignoring function, the specified workstation is not open
- **059**: Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

modified 2 April 1993
SEE ALSO

POST STRUCTURE (3P)
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED (3P)
INQUIRE PATHS TO DESCENDANTS (3P)
NAME

INQUIRE PREDEFINED COLOUR REPRESENTATION – obtain predefined colour representation for workstation type

SYNOPSIS

C Syntax

```c
void pinq_pred_colr_rep ( type, index, error_ind, bundle )
```

Pint type;  /* workstation type */
Pint index; /* predefined index */
Pint *error_ind; /* OUT error indicator */
Pcolr_rep *bundle; /* OUT predefined colour rep */

FORTRAN Syntax

```fortran
SUBROUTINE pqpcr ( WTYPE, PCI, CCSBSZ, ERRIND, OL, CSPEC )
```

INTEGER WTYPE /* workstation type */
INTEGER PCI /* predefined colour index */
INTEGER CCSBSZ /* colour component specification buffer size */
INTEGER ERRIND /* OUT error indicator */
REAL CSPEC /* OUT colour component array */
INTEGER OL /* OUT number of components in colour specification */

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose

Use INQUIRE PREDEFINED COLOUR REPRESENTATION to determine the predefined colour representation for a specified workstation type at a given colour index.

C Input Parameters

- `type` Type of workstation.
- `index` Entry in the workstation table of predefined colour representations to be returned.

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error that this function detects.
- `bundle` A pointer to a Pcolr_rep structure in which the system returns the colour representation at `index` on the workstation table of predefined colour representations. Pcolr_rep is defined in phigs.h as:

```c
typedef union {
    Prgb rgb; /* Red Green Blue colour specification */
    Pcieluv cieluv; /* CIE L*U*V* colour specification */
    Phls hls; /* Hue Lightness Saturation colour specification */
    Phsv hsv; /* Hue Saturation Value colour specification */
} Pcolr_rep;
```

modified 2 April 1993
INQUIRE PREDEFINED COLOUR REPRESENTATION (3P)

Pdata unsupp; /* Colour in unsupported colour model */
} Pcolr_rep;

Prgb is defined in phigs.h as follows:
typedef struct {
    Pfloat red; /* red, hue, and so on */
    Pfloat green; /* green, saturation, lightness, and so on */
    Pfloat blue; /* blue, value, saturation, and so on */
} Prgb;

Pcieluv is defined in phigs.h as follows:
typedef struct {
    Pfloat cieluv_x; /* x coefficient */
    Pfloat cieluv_y; /* y coefficient */
    Pfloat cieluv_y_lum; /* y luminance */
} Pcieluv;

Phsv is defined in phigs.h as follows:
typedef struct {
    Pfloat hue; /* hue */
    Pfloat satur; /* saturation */
    Pfloat value; /* value */
} Phsv;

Phls is defined in phigs.h as follows:
typedef struct {
    Pfloat hue; /* hue */
    Pfloat lightness; /* lightness */
    Pfloat satur; /* saturation */
} Phls;

Pdata is defined in phigs.h as follows:
typedef struct {
    size_t size; /* size of data */
    char *data /* pointer to data */
} Pdata;

FORTRAN Input

Parameters

WTYPE Type of workstation.

PCI Entry in the workstation table of predefined colour representations to be returned.

CCSBSZ Size of the component array allocated by the application. This should be 3 for all colour models supported by SunPHIGS.
### FORTRAN Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERRIND</strong></td>
<td>The error number of any error that this function detects.</td>
</tr>
<tr>
<td><strong>OL</strong></td>
<td>Number of colour components in the colour specification. This will be 3 for all colour models supported by SunPHIGS.</td>
</tr>
<tr>
<td><strong>CSPEC</strong></td>
<td>Array in which the components of the colour representation at index PCI are returned.</td>
</tr>
</tbody>
</table>

### ERRORS

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **051** Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- **052** Ignoring function, workstation type not recognized by the implementation
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- **062** Ignoring function, this information is not available for this MO workstation type
- **113** Ignoring function, the colour index value is less than zero
- **102** Ignoring function, the specified representation has not be predefined on this workstation

### SEE ALSO

- INQUIRE COLOUR FACILITIES (3P)
- SET COLOUR REPRESENTATION (3P)
- INQUIRE COLOUR REPRESENTATION (3P)
**NAME**
INQUIRE PREDEFINED EDGE REPRESENTATION – obtain predefined edge representation for workstation type

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqpedr ( WTYPE, PEDI, ERRIND, EDFLAG, EDTYPE, EWIDTH, COLI )
INTEGER WTYPE  /* workstation type */
INTEGER PEDI  /* predefined edge index */
INTEGER ERRIND  /* OUT error indicator */
INTEGER EDFLAG  /* OUT edge flag (POFF, PON) */
INTEGER EDTYPE  /* OUT edgetype */
REAL EWIDTH  /* OUT edgewidth scale factor */
INTEGER COLI  /* OUT edge colour index */
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**
Use INQUIRE PREDEFINED EDGE REPRESENTATION to determine the predefined edge representation for a specified workstation type at a given edge index.

**C Input Parameters**

- **type**  Type of workstation.
- **index**  Entry to be returned from the workstation table of predefined edge representations.

**C Output Parameters**

- **error_ind**  A pointer to the location to store the error number of any error that this function detects.
- **bundle**  A pointer to a Pedge_bundle structure in which the system returns the edge representation at index in the workstation table of predefined edge representations. Pedge_bundle is defined in phigs.h as:

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

---

698 modified 2 April 1993
INQUIRE PREDEFINED EDGE REPRESENTATION (3P)

typedef enum {
    PEDGE_OFF,
    PEDGE_ON
} Pedge_flag;

FORTRAN Input Parameters

WTYPE Type of workstation.
PEDI Entry to be returned from the workstation table of predefined edge representations.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

EDFLAG The edge flag value at index PEDI in the workstation table of predefined edge representations. Valid values for the edge flag are defined in phigs77.h as:

POFF
PON

EDTYPE The edge type at index PEDI in the workstation table of predefined edge representations.

EWIDTH The edge width scale factor at index PEDI in the workstation table of predefined edge representations.

COLI The edge colour index at index PEDI in the workstation table of predefined edge representations.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
051 Ignoring function, this information is not yet available for this workstation type;
open a workstation of this type and use the specific workstation type
052 Ignoring function, workstation type not recognized by the implementation
059 Ignoring function, the specified workstation does not have output capability (that
is, the workstation category is not OUTPUT, OUTIN, or MO)
062 Ignoring function, this information is not available for this MO workstation type
100 Ignoring function, the bundle index value is less than one
102 Ignoring function, the specified representation has not be predefined on this
workstation
134 Ignoring function, the requested entry contains a general colour specification
with colour type other than INDIRECT.

modified 2 April 1993
SEE ALSO

INQUIRE EDGE FACILITIES (3P)
SET EDGE REPRESENTATION (3P)
INQUIRE PREDEFINED EDGE REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED INTERIOR REPRESENTATION

**NAME**
INQUIRE PREDEFINED INTERIOR REPRESENTATION – obtain predefined interior representation for workstation type

**SYNOPSIS**

**C Syntax**
```c
void pinq_pred_int_rep ( type, index, error_ind, bundle )
Pint type; /* workstation type */
Pint index; /* predefined index */
Pint *error_ind; /* OUT error indicator */
Pint_bundle *bundle; /* OUT predefined interior rep */
```

**FORTRAN Syntax**
```fortran
SUBROUTINE pqpir ( WTYPE, PII, ERRIND, STYLE, STYLID, COLI )
INTEGER WTYPE /* workstation type */
INTEGER PII /* predefined interior index */
INTEGER ERRIND /* OUT error indicator */
INTEGER STYLE /* OUT interior style */
INTEGER STYLID /* OUT interior style index */
INTEGER COLI /* OUT interior colour index */
```

**DESCRIPTION**

**Purpose**
Use INQUIRE PREDEFINED INTERIOR REPRESENTATION to determine the predefined interior representation for a specified workstation type at a given interior index.

**C Input Parameters**
- `type` Type of workstation.
- `index` Entry to be returned from the workstation table of predefined interior representations.

**C Output Parameters**
- `error_ind` A pointer to the location to store the error number of any error that this function detects.
- `bundle` A pointer to a Pint_bundle structure in which the system returns the interior representation at `index` in the workstation table of predefined interior representations. Pint_bundle is defined in phigs.h as:
  ```c
  typedef struct {
    Pint_style style; /* interior style */
    Pint style_ind; /* interior style index */
    Pint colr_ind; /* interior colour index */
  } Pint_bundle;
  ```
  Pint_style is defined in phigs.h as:

modified 2 April 1993
typedef enum {
    PSTYLE_HOLLOW,
    PSTYLE_SOLID,
    PSTYLE_PAT,
    PSTYLE_HATCH,
    PSTYLE_EMPTY
} Pint_style;

See SET INTERIOR STYLE for a description of each style.

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTYPE</td>
<td>Type of workstation.</td>
</tr>
<tr>
<td>PII</td>
<td>Entry to be returned from the workstation table of predefined interior representations.</td>
</tr>
</tbody>
</table>

**FORTRAN Output Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRIND</td>
<td>The error number of any error that this function detects.</td>
</tr>
<tr>
<td>STYLE</td>
<td>The interior style at index PII in the workstation table of predefined interior representations. Valid values for the interior style are defined in phigs77.h as:</td>
</tr>
<tr>
<td>0</td>
<td>PHOLLO</td>
</tr>
<tr>
<td>1</td>
<td>PSOLID</td>
</tr>
<tr>
<td>2</td>
<td>PPATTR</td>
</tr>
<tr>
<td>3</td>
<td>PHATCH</td>
</tr>
<tr>
<td>4</td>
<td>PISEMP</td>
</tr>
<tr>
<td>STYLID</td>
<td>The interior style index at index PII in the workstation table of predefined interior representations.</td>
</tr>
<tr>
<td>COLI</td>
<td>The interior colour index at index PII in the workstation table of predefined interior representations.</td>
</tr>
</tbody>
</table>

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>062</td>
<td>Ignoring function, this information is not available for this MO workstation type</td>
</tr>
<tr>
<td>100</td>
<td>Ignoring function, the bundle index value is less than one</td>
</tr>
<tr>
<td>102</td>
<td>Ignoring function, the specified representation has not be predefined on this workstation</td>
</tr>
<tr>
<td>134</td>
<td>Ignoring function, the requested entry contains a general colour specification with colour type other than INDIRECT.</td>
</tr>
</tbody>
</table>

702 modified 2 April 1993
SEE ALSO

INQUIRE INTERIOR FACILITIES (3P)
SET INTERIOR REPRESENTATION (3P)
INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS (3PP)

modified 2 April 1993
NAME
INQUIRE PREDEFINED PATTERN REPRESENTATION – obtain predefined pattern representation for workstation type

SYNOPSIS
C Syntax

```c
void pinq_pred_pat_rep ( type, index, store, error_ind, rep )
```

- `Pint type;` workstation type
- `Pint index;` predefined index
- `Pstore store;` handle to Store object
- `Pint *error_ind;` OUT error indicator
- `Ppat_rep **rep;` OUT predefined pattern rep

FORTRAN Syntax

```fortran
SUBROUTINE pqppar ( WTYPE, PPAI, DIMX, DIMY, ERRIND, DX, DY, COLIA )
```

- `INTEGER WTYPE` workstation type
- `INTEGER PPAI` predefined pattern index
- `INTEGER DIMX, DIMY` maximum pattern array dimensions
- `INTEGER ERRIND` OUT error indicator
- `INTEGER DX, DY` OUT pattern array dimensions
- `INTEGER COLIA(DIMX, DIMY)` OUT pattern array

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION
Purpose

Use INQUIRE PREDEFINED PATTERN REPRESENTATION to determine the predefined pattern representation for a specified workstation type at a given pattern index.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by `rep`.

- `type` Workstation type.
- `index` Entry to be returned from the workstation table of predefined pattern representations.
- `store` The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error that this function detects.

704 modified 2 April 1993
rep A pointer to a pointer to a Ppat_rep structure in which the system returns the pattern representation at index in the workstation table of predefined pattern representations. Ppat_rep is defined in phigs.h as:

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

The Pint_size structure used to define the pattern dimensions is defined in phigs.h as:

```c
typedef struct {
    Pint size_x; /* dimension (number of divisions) along X */
    Pint size_y; /* dimension (number of divisions) along Y */
} Pint_size;
```

The colr_array component of Ppat_rep is a pointer to an array of the colour indices defining the pattern of the dimensions defined by the dims component.

**FORTRAN Input Parameters**

- **WTYPE** Workstation type.
- **PPAI** Entry to be returned from the workstation table of predefined pattern representations.
- **DIMX** The x dimension of the COLIA array in which the requested pattern representation is to be returned. If this value is smaller than the actual x dimension of the pattern representation to be returned (DX), no data will be returned in the COLIA array, but DX and DY will be set to indicate the array size required.
- **DIMY** The y dimension of the COLIA array in which the requested pattern representation is to be returned. If this value is smaller than the actual y dimension of the pattern representation to be returned (DY), no data will be returned in the COLIA array, but DX and DY will be set to indicate the array size required.

If you call this function with both dimensions set to zero, DX and DY will be set to indicate the array size required. Error 2001 will be returned if either dimension is too small, but not if both are zero.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **DX** The x dimension of the pattern representation returned in COLIA.
- **DY** The y dimension of the pattern representation returned in COLIA.
- **COLIA** An array of integers in which the system returns the pattern representation at PPAI in the workstation table of pattern representations.
## ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>Ignoring function, function requires state (PHOP, *, *, *)</td>
</tr>
<tr>
<td>051</td>
<td>Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type</td>
</tr>
<tr>
<td>052</td>
<td>Ignoring function, workstation type not recognized by the implementation</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>062</td>
<td>Ignoring function, this information is not available for this MO workstation type</td>
</tr>
<tr>
<td>102</td>
<td>Ignoring function, the specified representation has not be predefined on this workstation</td>
</tr>
<tr>
<td>112</td>
<td>Ignoring function, the pattern index value is less than one</td>
</tr>
</tbody>
</table>

## SEE ALSO

- INQUIRE PATTERN FACILITIES (3P)
- SET PATTERN REPRESENTATION (3P)
- INQUIRE PATTERN REPRESENTATION (3P)
- INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS (3PP)
### NAME
INQUIRE PREDEFINED POLYLINE REPRESENTATION – obtain predefined polyline representation for workstation type

### SYNONYMS

#### C Syntax
```c
void pinq_pred_line_rep ( type, index, error_ind, bundle )
Pint type; /* workstation type */
Pint index; /* predefined index */
Pint *error_ind; /* OUT error indicator */
Pline_bundle *bundle; /* OUT predefined polyline rep */
```

#### FORTRAN Syntax
```fortran
SUBROUTINE ppplpr ( WTYPE, PLI, ERRIND, LTYPE, LWIDTH, COLI )
INTEGER WTYPE /* workstation type */
INTEGER PLI /* predefined polyline index */
INTEGER ERRIND /* OUT error indicator */
INTEGER LTYPE /* OUT linetype */
REAL LWIDTH /* OUT linewidth scale factor */
INTEGER COLI /* OUT polyline colour index */
```

### DESCRIPTION

#### Purpose
Use INQUIRE PREDEFINED POLYLINE REPRESENTATION to determine the predefined polyline representation for a specified workstation type at a given polyline index.

#### C Input Parameters
- **type**: Type of workstation.
- **index**: Entry to be returned from the workstation table of predefined polyline representations.

#### C Output Parameters
- **error_ind**: A pointer to the location to store the error number of any error that this function detects.
- **bundle**: A pointer to a Pline_bundle data structure in which the system returns the polyline representation at `index` in the workstation table of predefined polyline representations. Pline_bundle is defined in phigs.h as:

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

Values for `type` are:

---

modified 2 April 1993
FORTRAN Input Parameters

WTYPE Type of workstation.
PLI Entry to be returned from the workstation table of predefined polyline representations.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.
LTYPE The line type at index PLI in the workstation table of predefined polyline representations.
LWIDTH The line width scale factor at index PLI in the workstation table of predefined polyline representations.
COLI The colour index at index PLI in the workstation table of predefined polyline representations.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)
051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
052 Ignoring function, workstation type not recognized by the implementation
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062 Ignoring function, this information is not available for this MO workstation type
100 Ignoring function, the bundle index value is less than one
102 Ignoring function, the specified representation has not be predefined on this workstation
134 Ignoring function, the requested entry contains a general colour specification with colour type other than INDIRECT.

SEE ALSO

INQUIRE POLYLINE FACILITIES (3P)
SET POLYLINE REPRESENTATION (3P)
INQUIRE POLYLINE REPRESENTATION (3P)
INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS (3PP)

modified 2 April 1993
NAME
INQUIRE PREDEFINED POLYMARKER REPRESENTATION – obtain predefined polymarker representation for workstation type

SYNOPSIS
C Syntax

```c
void pinq_pred_marker_rep ( type, index, error_ind, bundle )

Pint type; /* workstation type */
Pint index; /* predefined index */
Pint *error_ind; /* OUT error indicator */
Pmarker_bundle *bundle; /* OUT predefined polymarker rep */
```

FORTRAN Syntax

```fortran
SUBROUTINE pqppmr ( WTYPE, PMI, ERRIND, MTYPE, MSZSF, COLI )

INTEGER WTYPE /* workstation type */
INTEGER PMI /* predefined polymarker index */
INTEGER ERRIND /* OUT error indicator */
INTEGER MTYPE /* OUT marker type */
REAL MSZSF /* OUT marker size scale factor */
INTEGER COLI /* OUT polymarker colour index */
```

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose
Use INQUIRE PREDEFINED POLYMARKER REPRESENTATION to determine the predefined polymarker representation for a specified workstation type at a given polymarker index.

C Input Parameters

- `type` Type of workstation.
- `index` Entry to be returned from the workstation table of predefined polymarker representations.

C Output Parameters

- `error_ind` A pointer to the location to store the error number of any error that this function detects.
- `bundle` A pointer to a Pmarker_bundle structure, in which the system returns the polymarker representation at `index` in the workstation table of predefined polymarker representations. Pmarker_bundle is defined in phigs.h as:

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

Values for `type` are:

modified 2 April 1993
INQUIRE PREDEFINED POLYMARKER REPRESENTATION (3P)

FORTRAN Input Parameters

**WTYPE** Type of workstation.

**PMI** Entry to be returned from the workstation table of predefined polymarker representations.

FORTRAN Output Parameters

**ERRIND** The error number of any error that this function detects.

**MTYPE** The marker type at index PMI in the workstation table of predefined polymarker representations.

**MSZSF** The marker size scale factor at index PMI in the workstation table of predefined polymarker representations.

**COLI** The marker colour index at index PMI in the workstation table of predefined polymarker representations.

**ERRORS**

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not be predefined on this workstation

134 Ignoring function, the requested entry contains a general colour specification with *colour type* other than INDIRECT.

SEE ALSO

INQUIRE POLYMARKER FACILITIES (3P)

SET POLYMARKER REPRESENTATION (3P)

INQUIRE POLYMARKER REPRESENTATION (3P)

INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS (3PP)
NAME
INQUIRE PREDEFINED TEXT REPRESENTATION – obtain predefined text representation for workstation type

SYNOPSIS
C Syntax
void
pinq_pred_text_rep ( type, index, error_ind, bundle )
Pint type; /* workstation type */
Pint index; /* predefined index */
Pint *error_ind; /* OUT error indicator */
Ptext_bundle *bundle; /* OUT predefined text rep */

FORTRAN Syntax
SUBROUTINE pqptxr ( WTYPE, PTXI, ERRIND, FONT, PREC, CHXP, CHSP, COLI )
INTEGER WTYPE /* workstation type */
INTEGER PTXI /* predefined text index */
INTEGER ERRIND /* OUT error indicator */
INTEGER FONT /* OUT text font */
INTEGER PREC /* OUT text precision (PSTRP, PCHARP, PSTRKP) */
REAL CHXP /* OUT character expansion factor */
REAL CHSP /* OUT character spacing */
INTEGER COLI /* OUT text colour index */

Required PHIGS
Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE PREDEFINED TEXT REPRESENTATION to determine the predefined text representation for a specified workstation type at a given text index.

C Input Parameters

  type  Type of workstation.
  index Entry to be returned from the workstation table of predefined text representations.

C Output Parameters

  error_ind A pointer to the location to store the error number of any error that this function detects.
  bundle A pointer to a Ptext_bundle structure, in which the system returns the text representation at index in the workstation table of predefined text representations. Ptext_bundle is defined in phigs.h as:

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

Ptext_prec is defined in phigs.h as:

typedef enum {
    PPREC_STRING, /* string precision */
    PPREC_CHAR,  /* character precision */
    PPREC_STROKE /* stroke precision */
}

Ptext_prec;

FORTRAN Input Parameters

WTYPE Type of workstation.

PTXI Entry to be returned from the workstation table of predefined text representations.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

FONT The text font at index PTXI in the workstation table of predefined interior representations.

PREC The text precision at index PTXI in the workstation table of predefined interior representations. Valid values for the interior style are defined in phigs77.h as:

0 PSTRP String
1 PCHARP Character
2 PSTRPK Stroke

CHXP The character expansion factor at index PTXI in the workstation table of predefined interior representations.

CHSP The character spacing at index PTXI in the workstation table of predefined interior representations.

COLI The text colour index at index PTXI in the workstation table of predefined interior representations.

ERRORS

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

modified 2 April 1993
102 Ignoring function, the specified representation has not be predefined on this workstation

134 Ignoring function, the requested entry contains a general colour specification with colour type other than INDIRECT.

SEE ALSO

INQUIRE TEXT FACILITIES (3P)
SET TEXT REPRESENTATION (3P)
INQUIRE TEXT REPRESENTATION (3P)
INQUIRE PREDEFINED TEXT REPRESENTATION PLUS (3PP)

modified 2 April 1993
NAME
INQUIRE PREDEFINED VIEW REPRESENTATION – obtain predefined view representation for workstation type

SYNOPSIS
C Syntax
void
pinq_pred_view_rep ( type, index, error_ind, rep )
Pint type; workstation type
Pint index; predefined view index
Pint *error_ind; OUT error indicator
Pview_rep3 *rep; OUT view representation

FORTRAN Syntax
SUBROUTINE pqpvwr ( WTYPE, PVWI, ERRIND, VWORMT, VWMPMT,
VWCPLM, XYCLPI, BCLIPI, FCLIPI )
INTEGER WTYPE workstation type
INTEGER PVWI predefined view index
INTEGER ERRIND OUT error indicator
REAL VWORMT(4,4) OUT view orientation matrix
REAL VWMPMT(4,4) OUT view mapping matrix
REAL VWCPLM(6) OUT view clipping limits (NPC)
INTEGER XYCLPI OUT x-y clipping indicator (PNCLIP, PCLIP)
INTEGER BCLIPI OUT back clipping indicator (PNCLIP, PCLIP)
INTEGER FCLIPI OUT front clipping indicator (PNCLIP, PCLIP)

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
INQUIRE PREDEFINED VIEW REPRESENTATION determines the predefined view representation for a specified workstation type at a given view index.

C Input Parameters
 type Type of workstation.
 index Entry to be returned from the workstation table of predefined view representations.

C Output Parameters
 error_ind A pointer to the location to store the error number of any error that this function detects.
 rep A pointer to a Pview_rep3 structure in which the system returns the view representation at index in the workstation table of predefined view representations. Pview_rep3 is defined in phigs.h as follows:

typedef struct {
  Pmatrix3 ori_matrix; /* orientation matrix */
  Pmatrix3 map_matrix; /* mapping matrix */
} Pview_rep3;

714 modified 2 April 1993
The Pmatrix3 type definition is a 4x4 matrix, defined in phigs.h as:

typedef Pfloat Pmatrix3[4][4];

The Plimit3 structure used to define the clipping limits is defined in phigs.h as follows:

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

The clipping indicators control whether the clipping limits for the associated plane are active or inactive. Valid values for the Pclip enumerated type are defined in phigs.h as:

typedef enum {
  PIND_NO_CLIP,       // Do not clip
  PIND_CLIP           // Perform clipping
} Pclip_ind;

FORTRAN Input
Parameters

WTYPE  Type of workstation.
PVWI   Entry to be returned from the workstation table of predefined view representations.

FORTRAN Output
Parameters

ERRIND The error number of any error that this function detects.

VWORMT The view orientation matrix at index PVWI in the workstation table of predefined view representations.

VWMPMT The view mapping matrix at index PVWI in the workstation table of predefined view representations.

VWCPLM The view clipping limits at index PVWI in the workstation table of predefined view representations.
view representations; the first two elements of this array give the minimum and maximum clipping values in NPC for $x$, the next two for $y$, and the last two for $z$.

**XYCLPI**

The $x$-$y$ plane clipping indicator at index PVWI in the workstation table of predefined view representations.

**BCLIPI**

The back plane clipping indicator at index PVWI in the workstation table of predefined view representations.

**FCLIPI**

The front plane clipping indicator at index PVWI in the workstation table of predefined view representations.

The clipping indicators control whether the clipping limits for the associated plane are active or inactive. Valid values for the clipping indicators are defined in phigs77.h as:

- **PCLIP**  Perform clipping
- **PNCLIP** Do not clip

**ERRORS**

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **051** Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type
- **052** Ignoring function, workstation type not recognized by the implementation
- **057** Ignoring function, specified workstation is of category MI
- **062** Ignoring function, this information is not available for this MO workstation type
- **101** Ignoring function, the specified representation has not been defined
- **114** Ignoring function, the view index value is less than zero

**SEE ALSO**

- INQUIRE VIEW FACILITIES (3P)
- SET VIEW REPRESENTATION 3 (3P)
- INQUIRE VIEW REPRESENTATION (3P)
- SET VIEW INDEX (3P)
NAME

INQUIRE SET OF OPEN WORKSTATIONS – obtain current set of open workstations

SYNOPSIS

C Syntax

void

pinq_open_wss ( length, start, error_ind, idlist, total_length )

Pint length; length of application list
Pint start; starting position
Pint *error_ind; OUT error indicator
Pint_list *idlist; OUT list of ws ids
Pint *total_length; OUT length of list in PHIGS

FORTRAN Syntax

SUBROUTINE pqopwk ( N, ERRIND, OL, WKID )

INTEGER N set member requested
INTEGER ERRIND OUT error indicator
INTEGER OL OUT number of open workstations
INTEGER WKID OUT Nth member of set of open workstations

Required PHIGS
Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use INQUIRE SET OF OPEN WORKSTATIONS to obtain a list of the currently open workstations.

C Input Parameters

length The number of ints in the idlist output parameter for which the application has allocated memory. length is the number of list elements that the system can return in idlist→ints. If a value of 0 is used here, no data will be returned in the idlist→ints list, but the total number of elements will be returned in total_length.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by start, are copied sequentially into idlist→ints until idlist→ints is full or all the elements have been copied.

C Output Parameters

error_ind A pointer to the location to store the error number of any error that this function detects.

idlist A pointer to a Pint_list in which the system returns the portion of the list of currently-open workstations starting at the entry specified with start. Pint_list is defined in phigs.h as follows:

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

modified 2 April 1993 717
The `num ints` component specifies the number of workstation identifiers in the list.

The pointer `ints` must be initialized to an array, `length` long, of `Pint` elements.

`total_length`

A pointer to an integer in which to return the total length of the list. This is the value required for `length` if all the items in the list are to be returned.

<table>
<thead>
<tr>
<th>FORTRAN Input Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>N</code></td>
<td>Position in the list of the item requested. The Nth entry in the list of open workstations will be returned in <code>WKID</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORTRAN Output Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ERRIND</code></td>
<td>The error number of any error that this function detects.</td>
</tr>
<tr>
<td><code>OL</code></td>
<td>The total number of currently-open workstations.</td>
</tr>
<tr>
<td><code>WKID</code></td>
<td>The workstation identifier of the Nth entry in the list of open workstations.</td>
</tr>
</tbody>
</table>

**ERRORS**

- **002** Ignoring function, function requires state `(PHOP, *, *, *)`
- **2201** `C`: Start index out of range

**SEE ALSO**

- `OPEN WORKSTATION (3P)`
- `CLOSE WORKSTATION (3P)`
- `INQUIRE WORKSTATION STATE VALUE (3P)`

modified 2 April 1993
NAME

INQUIRE SET OF WORKSTATIONS TO WHICH POSTED – obtain list of workstations to which structure is posted

SYNOPSIS

C Syntax

void

pinq_wss_posted ( struct_id, length, start, error_ind, ws, total_length )

Pint struct_id;   structure identifier
Pint length;     length of application list
Pint start;      starting position
Pint *error_ind; OUT error indicator
Pint_list *ws;   OUT list of workstations
Pint *total_length; OUT length of list in PHIGS

FORTRAN Syntax

SUBROUTINE pqwkpo ( STRID, N, ERRIND, OL, WKID )

INTEGER STRID   structure identifier
INTEGER N      set member requested
INTEGER ERRIND OUT error indicator
INTEGER OL     OUT number of workstations to which the structure is posted
INTEGER WKID   OUT Nth member of set of workstations to which the structure is posted

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

Use INQUIRE SET OF WORKSTATIONS TO WHICH POSTED to obtain a list of workstations to which a specified structure is posted.

C Input Parameters

struct_id

Identifier of the structure for which the list of workstations it is posted to will be returned.

length

The number of ints items in the ws output parameter for which the application has allocated memory. length is the number of list elements (workstation identifiers) that the system can return in ints. If a value of 0 is used here, no data will be returned in the ints list, but the total number of workstations the structure is posted to will be returned in total_length.

start

Starting position of inquiry into the list of workstations to which this structure is posted. The elements of the list of workstation identifiers, beginning with the item number specified by start, are copied sequentially into ints until ints is full or all the workstation identifiers have been copied.

C Output Parameters

error_ind

A pointer to the location to store the error number of any error that this function detects.

modified 2 April 1993
ws A pointer to a Pint_list structure in which the system returns the list of workstations to which struct_id is posted. Pint_list is defined in phigs.h as follows:

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

The num_ints component specifies the number of workstation identifiers in the list. The ints component is a pointer to a list, num_ints long, of the workstation identifiers.

The application must allocate memory for length elements in the ints list.

total_length A pointer to an integer in which the system returns the total number of workstations to which the specified structure is posted. This is the value required for length if the entire list is to be returned.

FORTRAN Input Parameters

STRID Identifier of the structure whose workstations posted to list will be returned.

N Element of the list of workstations to which the structure STRID is posted to return; only one workstation identifier may be inquired per subroutine call. If a value of 0 is used here, then no workstation identifier will be returned, but the total number of elements in the structure state list of workstations to which this structure is posted will be returned in OL.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

OL The total number of elements in the structure state list of workstations to which this structure is posted.

WKID The Nth workstation identifier from the structure state list of workstations to which this structure is posted.

ERRORS

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

201 Ignoring function, the specified structure does not exist

SEE ALSO

POST STRUCTURE (3P)
INQUIRE POSTED STRUCTURES (3P)
NAME
INQUIRE STRING DEVICE STATE – obtain state of STRING device

SYNOPSIS
C Syntax
void
pinq_string_st ( ws, dev, store, err, op_mode, echo_switch, init_string, prompt_echo,
      echo_area, string_data )

Pint    ws;  workstation identifier
Pint    dev;  string device number
Pstore  store;  handle to Store object
Pint    *err;  OUT error indicator
Pop_mode  *op_mode  OUT operating mode
Pecho_switch  *echo_switch;  OUT echo switch
char    **init_string;  OUT initial string
Pint      *prompt_echo;  OUT prompt/echo type
Plimit      *echo_area;  OUT echo area
Pstring_data  **string_data;  OUT data record

FORTRAN Syntax
SUBROUTINE pqsts ( WKID, STDNR, MLDR, ERRIND, MODE, ESW, LOSTR,
        ISTR, PET, EAREA, LDR, DATREC )

INTEGER    WKID  workstation identifier
INTEGER    STDNR  string device number
INTEGER    MLDR  dimension of data record array
INTEGER    ERRIND  OUT error indicator
INTEGER    MODE  OUT operating mode (PREQU, PSAMPL, PEVENT)
INTEGER    ESW  OUT echo switch (PNECHO, PECHO)
INTEGER    LOSTR  OUT number of characters returned
CHARACTER(*+)  ISTR  OUT initial string
INTEGER    PET  OUT prompt/echo type
REAL    EAREA(4)  OUT echo area in device coordinates
INTEGER    LDR  OUT number of array elements used in data record
CHARACTER*80  DATREC(MLDR)  OUT data record

FORTRAN Subset Syntax
SUBROUTINE pqsts ( WKID, STDNR, MLDR, ERRIND, MODE, ESW, LOSTR,
        ISTR, PET, EAREA, LDR, DATREC )

INTEGER    WKID  workstation identifier
INTEGER    STDNR  string device number
INTEGER    MLDR  dimension of data record array
INTEGER    ERRIND  OUT error indicator
INTEGER    MODE  OUT operating mode (PREQU, PSAMPL, PEVENT)
INTEGER    ESW  OUT echo switch (PNECHO, PECHO)
INTEGER    LOSTR  OUT number of characters returned
CHARACTER*80  ISTR  OUT initial string
INTEGER    PET  OUT prompt/echo type

modified 2 April 1993
721
INQUIRE STRING DEVICE STATE (3P)

REAL EAREA(4) OUT echo area in device coordinates
INTEGER LDR OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR) OUT data record

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE STRING DEVICE STATE to determine the current state of the specified STRING device.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The string data record within the store buffer is accessed by the pointer pointed to by string_data.

ws Workstation identifier. An integer specifying the workstation with which the specified STRING device is associated.

dev The device number of the string device. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available devices.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters
err A pointer to the location to store the error number of any error that this function detects.

op_mode The operating mode. Pop_mode is an enumerated type defined in phigs.h with the following values:

typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
} Pop_mode;

echo_switch The echo state. Pecho_switch is an enumerated type defined in phigs.h with the following values:

typedef enum {
    SWITCH_NO_ECHO,
    SWITCH_ECHO
} Pecho_switch;
**init_string**

A pointer to a `char` pointer. PHIGS sets the location pointed to by `init_string` to point to the initial string stored in `store`.

**prompt_echo**

The prompt/echo type desired. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available types.

**echo_area**

A pointer to a variable of type `Plimit` that contains the echo area of the device. `Plimit` is defined in `phigs.h` as follows:

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

**string_data**

Pointer to a pointer that points to the string state within `store`. `Pstring_data` is defined in `phigs.h` as follows:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    union {
        struct {
            Pint unused;
        } pet_r1;
    } pets;
} Pstring_data;
```

---

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the `MLDR` argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with `MLDR` set to zero.

Even if the dimension specified in `MLDR` is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial string, the prompt/echo type, the buffer length, the edit position, and the echo area.

Error 2001 is returned if `MLDR` is too small, but not if it is zero.

**WKID**  The workstation identifier of the workstation associated with the device.

**STDNR**  The device number of the STRING device. See the AVAILABLE DEVICES section of
INQUIRE STRING DEVICE STATE

**Mldr**
The dimension of the data record array, DATREC.

**Errind**
The error indicator. See the *Execution* section below for a description of its use. See the *Errors* section below for the possible values it may return.

**Mode**
The operating mode.

**Esw**
The echo switch.

**Lostr**
The number of characters in the initial string.

**Str**
The array in which to store the initial string. The size of this array should be at least as large as the buffer size of the device.

**Pet**
The prompt/echo type.

**Earea**
An array containing the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

**Ldr**
The required dimension of the data record array, DATREC.

**Datrec**
The data record array.

**Execution**
INQUIRE STRING DEVICE STATE returns the current state of the specified string device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial string, prompt/echo type, echo area, buffer length, edit position, and data record. See SET STRING MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE STRING for a description of the initial string, prompt/echo type, echo area and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN *Parameters* sections above, if an error is detected by this function the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td><strong>C</strong>: Buffer overflow in input or inquiry function</td>
</tr>
<tr>
<td>2001</td>
<td><strong>FORTRAN</strong>: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value</td>
</tr>
</tbody>
</table>

modified 2 April 1993
SEE ALSO

SET STRING MODE (3P)
INITIALIZE STRING (3P)
INQUIRE STRING DEVICE STATE 3 (3P)
INQUIRE STRING DEVICE STATE 3 – inquire state of a STRING device

**SYNOPSIS**

**C Syntax**

```c
void pinq_string_st3 ( ws, dev, store, err, op_mode, echo_switch, init_string, prompt_echo,
                        echo_vol, string_data )
```

- `Pint ws;` workstation identifier
- `Pint dev;` string device number
- `Pstore store;` handle to Store object
- `Pint *err;` OUT error indicator
- `Pop_mode *op_mode` OUT operating mode
- `Pecho_switch *echo_switch;` OUT echo switch
- `char **init_string;` OUT initial string
- `Pint *prompt_echo;` OUT prompt/echo type
- `Plimit3 *echo_vol;` OUT echo area
- `Pstring_data3 **string_data;` OUT data record

**FORTRAN Syntax**

```fortran
SUBROUTINE pqsts3 ( WKID, STDNR, MLDR, ERRIND, MODE, ESW, LOSTR,
                      ISTR, PET, EVOL, LDR, DATREC )
```

- `INTEGER WKID` workstation identifier
- `INTEGER STDNR` string device number
- `INTEGER MLDR` dimension of data record array
- `INTEGER ERRIND` OUT error indicator
- `INTEGER MODE` OUT operating mode (PREQU, PSAMPL, PEVENT)
- `INTEGER ESW` OUT echo switch (PNECHO, PECHO)
- `INTEGER LOSTR` OUT number of characters returned
- `CHARACTER*(*) ISTR` OUT initial string
- `INTEGER PET` OUT prompt/echo type
- `REAL EVOL(6)` OUT echo volume in device coordinates
- `INTEGER LDR` OUT number of array elements used in data record
- `CHARACTER*80 DATREC(MLDR)` OUT data record

**FORTRAN Subset Syntax**

```fortran
SUBROUTINE pqsts3 ( WKID, STDNR, MLDR, ERRIND, MODE, ESW, LOSTR,
                      ISTR, PET, EVOL, LDR, DATREC )
```

- `INTEGER WKID` workstation identifier
- `INTEGER STDNR` string device number
- `INTEGER MLDR` dimension of data record array
- `INTEGER ERRIND` OUT error indicator
- `INTEGER MODE` OUT operating mode (PREQU, PSAMPL, PEVENT)
- `INTEGER ESW` OUT echo switch (PNECHO, PECHO)
- `INTEGER LOSTR` OUT number of characters returned
- `CHARACTER*80 ISTR` OUT initial string
- `INTEGER PET` OUT prompt/echo type

726 modified 2 April 1993
**DESCRIPTION**

**Purpose**
Use `INQUIRE STRING DEVICE STATE 3` to determine the current state of the specified STRING device.

**C Input Parameters**
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The string data record within the store buffer is accessed by the pointer pointed to by `string_data`.

- `ws` Workstation identifier. An integer specifying the workstation with which the specified STRING device is associated.
- `dev` The device number of the STRING device. See the `AVAILABLE DEVICES` section of `INITIALIZE STRING` for a description of the available devices.
- `store` The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see `CREATE STORE (3P)`).

**C Output Parameters**

- `err` A pointer to the location to store the error number of any error that this function detects. detected by this function.
- `op_mode` The operating mode. `Pop_mode` is an enumerated type defined in phigs.h with the following values:

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

- `echo_switch` The echo state. `Pecho_switch` is an enumerated type defined in phigs.h with the following values:

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

**Required PHIGS Operating States**

`(PHOP, WSOP, *, *)`

**modified 2 April 1993**
**init_string**

A pointer to a char pointer. PHIGS sets the location pointed to by *init_string* to point to the initial string stored in *store*.

**prompt_echo**

The prompt/echo type desired. See the AVAILABLE DEVICES section of INITIALIZE STRING for a description of the available types.

**echo_vol**

A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

```c
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
```

**string_data**

Pointer to a pointer that points to the string state within *store*. Pstring_data3 is defined in phigs.h as follows:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    union {
        struct {
            Pint unused;
        } pet_r1;
    } pets;
} Pstring_data3;
```

**FORTRAN Input Parameters**

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial string, the prompt/echo type, the buffer length, the edit position, and the echo volume.
Error 2001 is returned if MLDR is too small, but not if it is zero.

**WKID**  The workstation identifier of the workstation associated with the device.

**STDNR**  The device number of the STRING device. See the AVAILABLE DEVICES section of INITIALIZE STRING 3 for a description of the available devices.

**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MODE**  The operating mode.

**ESW**  The echo switch.

**LOSTR**  The number of characters in the initial string.

**STR**  The array in which to store the initial string. The size of this array should be at least as large as the buffer size of the device.

**PET**  The prompt/echo type.

**EVOL**  An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array.

### Execution

INQUIRE STRING DEVICE STATE 3 returns the current state of the specified string device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial string, prompt/echo type, echo volume, buffer length, edit position, and data record. See SET STRING MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE STRING 3 for a description of the initial string, prompt/echo type, echo volume and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function then the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

### ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
<tr>
<td>2200</td>
<td>C: Buffer overflow in input or inquiry function</td>
</tr>
</tbody>
</table>

modified 2 April 1993
FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

SET STRING MODE (3P)
INITIALIZE STRING (3P)
INQUIRE STRING DEVICE STATE (3P)

modified 2 April 1993
**NAME**

INQUIRE STROKE DEVICE STATE – obtain state of stroke device

**SYNOPSIS**

### C Syntax

```c
void
pinq_stroke_st ( ws, dev, type, store, err, op_mode, echo_switch, init_view_ind,
                 init_stroke, prompt_echo, echo_area, stroke_data )
```

- `Pint ws;`  
  workstation identifier
- `Pint dev;`  
  stroke device number
- `Pinq_type type;`  
  type of returned value
- `Pstore store;`  
  handle to Store object
- `Pint *err;`  
  OUT error indicator
- `Pop_mode *op_mode;`  
  OUT operating mode
- `Pecho_switch *echo_switch;`  
  OUT echo switch
- `Pint *init_view_ind;`  
  OUT initial view indicator
- `Ppoint_list **init_stroke;`  
  OUT initial stroke
- `Pint *prompt_echo;`  
  OUT prompt/echo type
- `Plimit *echo_area;`  
  OUT echo area
- `Pstroke_data **stroke_data;`  
  OUT data record

### FORTRAN Syntax

```fortran
SUBROUTINE pqsk ( WKID, SKDNR, TYPE, N, MLDR, ERRIND, MODE,
                   ESW, IVIEWI, NP, IPXA, IPYA, PET, EAREA, LDR, DATREC )
```

- `INTEGER WKID`  
  workstation identifier
- `INTEGER SKDNR`  
  stroke device number
- `INTEGER TYPE`  
  type of returned values (PSET, PREALI)
- `INTEGER N`  
  maximum number of points
- `INTEGER MLDR`  
  dimension of data record array
- `INTEGER ERRIND`  
  OUT error indicator
- `INTEGER MODE`  
  OUT operating mode (PREQU, PSAMPL, PEVENT)
- `INTEGER ESW`  
  OUT echo switch (PNECHO, PECCHO)
- `INTEGER IVIEWI`  
  OUT initial view index
- `INTEGER NP`  
  OUT number of points
- `REAL IPXA(N), IPYA(N)`  
  OUT initial points in stroke in World Coordinates
- `INTEGER PET`  
  OUT prompt/echo type
- `REAL EAREA(4)`  
  OUT echo area in Device Coordinates
- `INTEGER LDR`  
  OUT number of array elements used in data record
- `CHARACTER*80 DATREC(MLDR)`  
  OUT data record

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE STROKE DEVICE STATE to determine the current state of the specified stroke device.

modified 2 April 1993
C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the `store` argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to `CREATE STORE`, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The stroke device data record within the store buffer is accessed via the pointer pointed to by `stroke_data`.

- **ws** Workstation identifier. An integer specifying the workstation with which the specified stroke device is associated.
- **dev** The device number of the stroke device. See the `AVAILABLE DEVICES` section of `INITIALIZE STROKE` for a description of the available devices.
- **type** An enumerated value specifying whether the values to be returned are those originally specified by the application (`PINQ_SET`), or those resulting after PHIGS mapped them to ones available on the workstation (`PINQ_REALIZED`). A `Pinq_type` structure is defined as:

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

- **store** The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see `CREATE STORE (3P)`).

C Output Parameters

- **err** A pointer to the location to store the error number of any error that this function detects.
- **op_mode** The operating mode. `Pop_mode` is an enumerated type defined in `phigs.h` with the following values:

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

- **echo_switch** The echo state. `Pecho_switch` is an enumerated type defined in `phigs.h` with the following values:

```c
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
```
**init_view_ind**
Initial view indicator.

**init_stroke**
A pointer to a pointer that points to the initial stroke data contained with the store. Ppoint_list is defined in phigs.h as follows:

```c
typedef struct {
    Pint num_points; /* number of Ppoint3s in the list */
    Ppoint *points; /* list of points */
} Ppoint_list;
```

Ppoint is defined in phigs.h as follows:

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

**prompt_echo**
The prompt/echo type desired. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available types.

**echo_area**
A pointer to a variable of type Plimit that contains the echo area of the device. Plimit is defined in phigs.h as follows:

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

**stroke_data**
Pointer to a pointer that points to the stroke device state within store. Pstroke_data is defined in phigs.h as follows:

```c
typedef struct {
    Pint buffer_size; /* input buffer size */
    Pint init_pos; /* initial editing position */
    Pfloat x_interval; /* x interval */
    Pfloat y_interval; /* y interval */
    Pfloat time_interval; /* time interval */
    union {
        struct {
            Pint unused;
        } pet_r1;
    }
} Pstroke_data;
```
struct {
  Pint unused;
} pet_r2;

struct {
  Pmarker_attrs marker_attrs; /* marker attributes */
} pet_r3;

struct {
  Pline_attrs line_attrs; /* line attributes */
} pet_r4;

} pets;

#define Pstroke_data;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial stroke, initial view index, the prompt/echo type and the echo area.

Error 2001 is returned if MLDR is too small, but not if it is zero.

WKID The workstation identifier of the workstation associated with the device.

SKDNR The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.

TYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (Set), or those resulting after PHIGS mapped them to ones available on the workstation (Realized). Valid values are:

0 PSET Set
1 PREALI Realized

N The maximum number of initial points to return. The initial points arrays should be at least this size.

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

MODE The operating mode.

ESW The echo switch.

modified 2 April 1993
INQUIRE STROKE DEVICE STATE returns the current state of the specified stroke device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial stroke value, prompt/echo type, echo area and data record. See SET STROKE MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE STROKE for a description of the initial stroke value, prompt/echo type, echo area and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function then the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, then the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
061 Ignoring function, specified workstation is not of category INPUT or OUTIN
250 Ignoring function, the specified device is not available on the specified workstation
2200 C: Buffer overflow in input or inquiry function
2001 FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

SET STROKE MODE (3P)
INITIALIZE STROKE (3P)
INQUIRE STROKE DEVICE STATE 3 (3P)
NAME
INQUIRE STROKE DEVICE STATE 3—obtain state of stroke device

SYNOPSIS
void

\( pinq\_\text{stroke}\_\text{st3} ( \text{ws}, \text{dev}, \text{type}, \text{store}, \text{err}, \text{op\_mode}, \text{echo\_switch}, \text{init\_view\_ind}, \text{init\_stroke}, \text{prompt\_echo}, \text{echo\_vol}, \text{stroke\_data} ) \)

- Pint ws;  
  workstation identifier
- Pint dev;  
  stroke device number
- Pinq\_type type;  
  type of returned value
- Pstore store;  
  handle to Store object
- Pint *err;  
  OUT error indicator
- P\text{op\_mode} *op\_mode;  
  OUT operating mode
- P\text{echo\_switch} *echo\_switch;  
  OUT echo switch
- Pint *\text{init\_view\_ind};  
  OUT initial view indicator
- P\text{point\_list} **\text{init\_stroke};  
  OUT initial stroke
- Pint *\text{prompt\_echo};  
  OUT prompt/echo type
- Plimit3 *\text{echo\_vol};  
  OUT echo area
- P\text{stroke\_data3} **\text{stroke\_data};  
  OUT data record

FORTRAN Syntax
SUBROUTINE pqsk3 ( \text{WKID}, \text{SKDNR}, \text{TYPE}, \text{N}, \text{MLDR}, \text{ERRIND}, \text{MODE}, \text{ESW}, \text{IVIEWI}, \text{NP}, \text{IPXA}, \text{IPYA}, \text{IPZA}, \text{PET}, \text{EVOL}, \text{LDR}, \text{DATREC} )

- INTEGER WKID  
  workstation identifier
- INTEGER SKDNR  
  stroke device number
- INTEGER TYPE  
  type of returned values (PSET, PREALI)
- INTEGER N  
  maximum number of points
- INTEGER MLDR  
  dimension of data record array
- INTEGER ERRIND  
  OUT error indicator
- INTEGER MODE  
  OUT operating mode (PREQU, PSAMPL, PEVENT)
- INTEGER ESW  
  OUT echo switch (PNECHO, PECHO)
- INTEGER IVIEWI  
  OUT initial view index
- INTEGER NP  
  OUT number of points
- REAL IPXA(\text{N}), IPYA(\text{N}), IPZA(\text{N})  
  OUT initial points in stroke in World Coordinates
- INTEGER PET  
  OUT prompt/echo type
- REAL EVOL(6)  
  OUT echo volume in Device Coordinates
- INTEGER LDR
  OUT number of array elements used in data record
- CHARACTER*80 DATREC(MLDR)  
  OUT data record

Required PHIGS Operating States
(PHOP, WSOP, *, *)

modified 2 April 1993
**DESCRIPTION**

**Purpose**

Use INQUIRE STROKE DEVICE STATE 3 to determine the current state of the specified stroke device.

**C Input Parameters**

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the *store* argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The stroke device data record within the store buffer is accessed via the pointer pointed to by *stroke_data*.

- **ws** Workstation identifier. An integer specifying the workstation with which the specified stroke device is associated.
- **dev** The device number of the stroke device. See the AVAILABLE DEVICES section of INITIALIZE STROKE for a description of the available devices.
- **type** An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

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

- **store** The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

**C Output Parameters**

- **err** A pointer to the location to store the error number of any error that this function detects.
- **op_mode** The operating mode. Pop_mode is an enumerated type defined in phigs.h with the following values:

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

- **echo_switch** The echo state. Pecho_switch is an enumerated type defined in phigs.h with the following values:

  ```c
  typedef enum {
  ```

modified 2 April 1993
PSWITCH_NO_ECHO,
PSWITCH_ECHO
}

Pecho_switch;

init_view_ind
Initial view indicator.

init_stroke
A pointer to a pointer that points to the initial stroke data contained with the store. Ppoint_list3 is defined in phigs.h as follows:

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

Ppoint3 is defined in phigs.h as follows:

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

prompt_echo
The prompt/echo type desired. See the AVAILABLE DEVICES Section of INITIALIZE STROKE for a description of the available types.

echo_vol
A pointer to a Plimit3 structure defining the x, y, and z components of the echo volume, in Device Coordinates. Plimit3 is defined in phigs.h as follows:

typedef struct {
    Pfloat x_min;    /* minimum x coordinate */
    Pfloat x_max;    /* maximum x coordinate */
    Pfloat y_min;    /* minimum y coordinate */
    Pfloat y_max;    /* maximum y coordinate */
    Pfloat z_min;    /* minimum z coordinate */
    Pfloat z_max;    /* maximum z coordinate */
} Plimit3;

stroke_data
Pointer to a pointer that points to the stroke device state within store. Pstroke_data3 is defined in phigs.h as follows:

typedef struct {
    Pint buffer_size;    /* input buffer size */
    Pint init_pos;       /* initial editing position */
    Pfloat x_interval;   /* x interval */
} Pstroke_data3;
FORTRAN Input

Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial stroke, initial view index, the prompt/echo type, and the echo volume.

Error 2001 is returned if MLDR is too small, but not if it is zero.

WKID The workstation identifier of the workstation associated with the device.

SKDNR The device number of the STROKE device. See the AVAILABLE DEVICES section of INITIALIZE STROKE 3 for a description of the available devices.

TYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (Set), or those resulting after PHIGS mapped them to ones available on the workstation (Realized). Valid values are:

0 PSET Set
1 PREALI Realized

N The maximum number of initial points to return. The initial points arrays should be at least this size.
**MLDR**  The dimension of the data record array, DATREC.

**ERRIND**  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

**MODE**  The operating mode.

**ESW**  The echo switch.

**IVIEWI**  The initial view index.

**NP**  The number of initial stroke points.

**IPXA, IPYA, IPZA**  The arrays in which to store the initial stroke points.

**PET**  The prompt/echo type.

**EVOL**  An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

**LDR**  The required dimension of the data record array, DATREC.

**DATREC**  The data record array.

**Execution**  INQUIRE STROKE DEVICE STATE 3 returns the current state of the specified stroke device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial stroke value, prompt/echo type, echo volume and data record. See SET STROKE MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE STROKE 3 for a description of the initial stroke value, prompt/echo type, echo volume, and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function, the error indicator will indicate the error number of the error detected, and no other output data will be returned. If no error is detected, the error indicator will be set to zero, and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

061 Ignoring function, specified workstation is not of category INPUT or OUTIN

250 Ignoring function, the specified device is not available on the specified workstation

2200 C: Buffer overflow in input or inquiry function

740  modified 2 April 1993
FORTRAN: Ignoring function, output parameter size insufficient — a FORTRAN array or string being passed as an output parameter is too small to contain the returned value.

SEE ALSO

SET STROKE MODE (3P)
INITIALIZE STROKE (3P)
INQUIRE STROKE DEVICE STATE (3P)
NAME
INQUIRE STRUCTURE IDENTIFIERS—obtain current structure identifiers

SYNOPSIS
C Syntax

```c
void pinq_struct_ids ( length, start, error_ind, struct_ids, length_list )
Pint length; /* length of application list */
Pint start; /* starting position */
Pint *error_ind; /* OUT error indicator */
Pint_list *struct_ids; /* OUT list of structure ids */
Pint *length_list; /* OUT length of list in PHIGS */
```

FORTRAN Syntax

```fortran
SUBROUTINE pqsid ( N, ERRIND, NUMBER, STRID )
INTEGER N /* list element requested */
INTEGER ERRIND /* OUT error indicator */
INTEGER NUMBER /* OUT number of structure identifiers */
INTEGER STRID /* OUT Nth structure identifier */
```

Required PHIGS Operating States

(HP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE STRUCTURE IDENTIFIERS to obtain a list of the current structure identifiers from the PHIGS state list.

C Input Parameters

- **length**
  The number of ints items in the `struct_ids` output parameter for which the application has allocated memory. `length` is the number of list elements (structure identifiers) that the system can return in `struct_ids->ints`. If a value of 0 is used here, no data will be returned in the `struct_ids->ints` list, but the total number of structure identifiers in the PHIGS state list will be returned in `length_list`.

- **start**
  Starting position of inquiry into the PHIGS state list of current structure identifiers. The elements of the list of structure identifiers, beginning with the item number specified by `start`, are copied sequentially into `struct_ids->ints` until `struct_ids->ints` is full or all the structure identifiers have been copied.

C Output Parameters

- **error_ind**
  A pointer to the location to store the error number of any error that this function detects.

- **struct_ids**
  A pointer to a Pint_list structure in which the system returns the list of current structure identifiers. Pint_list is defined in phigs.h as follows:

```c
typedef struct {
    Pint num_ints; /* number of Pints in list */
    Pint *ints; /* list of integers */
} Pint_list;
```
The `num_ints` component specifies the number of structure identifiers in the list. The `ints` component is a pointer to a list, `num_ints` long, of the structure identifiers. The application must allocate memory for `length` elements in the `struct_ids->ints` list prior to calling this procedure.

`length_list`

A pointer to an integer in which the system returns the total number of elements in the PHIGS state list of currently-used structure identifiers. This is the value required for `length` if all structure identifiers are to be returned.

**FORTRAN Input Parameters**

- **N** Element of the PHIGS state list of current structure identifiers to return; only one identifier may be inquired per subroutine call. If a value of 0 is used here, no structure identifier will be returned, but the total number of elements in the PHIGS state list of currently-used structure identifiers will be returned in `NUMBER`.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **NUMBER** The total number of elements in the PHIGS state list of currently-used structure identifiers.
- **STRID** The Nth structure identifier from the PHIGS state list of currently-used structure identifiers.

**ERRORS**

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

**SEE ALSO**

- CHANGE STRUCTURE IDENTIFIER (3P)
- INQUIRE STRUCTURE STATUS (3P)

modified 2 April 1993

SunPHIGS Release 3.0
**NAME**
INQUIRE STRUCTURE STATE VALUE – obtain current structure state value

**SYNOPSIS**

**C Syntax**

```c
void pinq_struct_st ( struct_state )
Pstruct_st *struct_state;       OUT structure state
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pqstrs ( STRSTA )
INTEGER STRSTA       OUT structure state value (PSTCL, PSTOP)
```

**Required PHIGS Operating States**

(*) (*) (*) (*)

**DESCRIPTION**

**Purpose**
Use INQUIRE STRUCTURE STATE VALUE to determine the current structure state. The state value may be *structure open* (STOP) or *structure closed* (STCL).

**C Output Parameters**

- `struct_state`
  
  A pointer to a Pstruct_st enumerated variable in which the current structure state value is returned. Values for Pstruct_st are defined in phigs.h as follows:

  - PSTRUCT_ST_STCL: Structure closed
  - PSTRUCT_ST_STOP: Structure open

**FORTRAN Output Parameters**

- `STRSTA`

  The current structure state value, defined in phigs77.h as follows:

  - PSTCL: Structure closed
  - PSTOP: Structure open

**ERRORS**

No Error

**SEE ALSO**

- OPEN STRUCTURE (3P)
- CLOSE STRUCTURE (3P)
- INQUIRE OPEN STRUCTURE (3P)
- INQUIRE STRUCTURE IDENTIFIERS (3P)
- INQUIRE STRUCTURE STATUS (3P)

modified 2 April 1993
## NAME

INQUIRE STRUCTURE STATUS – obtain status of specified structure

## SYNOPSIS

### C Syntax

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

### FORTRAN Syntax

```fortran
SUBROUTINE pqstst ( STRID, ERRIND, STRSTI )
INTEGER STRID  // structure identifier
INTEGER ERRIND  // OUT error indicator
INTEGER STRSTI  // OUT structure status indicator
```

## Required PHIGS Operating States

(PHOP, *, *, *)

## DESCRIPTION

### Purpose

Use INQUIRE STRUCTURE STATUS to determine the existence status of the specified structure (NON_EXISTENT, EMPTY, or NOTEMPTY).

### C Input Parameters

- `struct_id`  
  Identifier of the structure for which the status is to be returned.

### C Output Parameters

- `error_ind`  
  A pointer to the location to store the error number of any error that this function detects.

- `struct`  
  A pointer to a Pstruct_status enumerated variable in which the system returns the existence status of the specified structure. Values for Pstruct_status are defined in phigs.h as follows:

  - PSTRUCT_STATUS_NON_EXISTENT
  - PSTRUCT_STATUS_EMPTY
  - PSTRUCT_STATUS_NOT_EMPTY

### FORTRAN Input Parameters

- `STRID`  
  Identifier of the structure for which the status is to be returned.

### FORTRAN Output Parameters

- `ERRIND`  
  The error number of any error that this function detects.

- `STRSTI`  
  The existence status of the specified structure, defined in phigs77.h as follows:

  - PNOEXS  Nonexistent
  - PEMPTY  Empty
  - PNOEMP  Not empty

modified 2 April 1993
### ERRORS

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

### SEE ALSO

- **DELETE STRUCTURE (3P)**
- **EMPTY STRUCTURE (3P)**
- **INQUIRE STRUCTURE IDENTIFIERS (3P)**
NAME
INQUIRE SYSTEM STATE VALUE – obtain PHIGS state value

SYNOPSIS
C Syntax
void pinq_sys_st ( sys_state )
    Psys_st *sys_state;  /* OUT the system state */

FORTRAN Syntax
SUBROUTINE pqsys ( SYSSTA )
    INTEGER SYSSTA  /* OUT system state value (PPHCL, PPHOP) */

Required PHIGS Operating States
(*, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE SYSTEM STATE VALUE to determine the current state of the PHIGS system. The state value may be either PHIGS open (PHOP) or PHIGS closed (PHCL).
PHIGS must be opened with the OPEN PHIGS subroutine before any operations can be performed with the PHIGS subroutines. This function can be used to verify that OPEN PHIGS was successful before performing additional PHIGS operations.

C Output Parameters
sys_state
A pointer to a Psys_st enumerated variable in which the current state of the PHIGS system is returned. Values for Psys_st are defined in phigs.h as follows:

```c
typedef enum {
    PSYS_ST_PHCL, PHIGS closed
    PSYS_ST_PHOP, PHIGS open
} Psys_st;
```

FORTRAN Output Parameters
SYSSTA The current state of the PHIGS system, defined in phigs77.h as follows:

```fortran
PPHCL PHIGS closed
PPHOP PHIGS open
```

ERRORS
No Error

SEE ALSO
OPEN PHIGS (3P)
CLOSE PHIGS (3P)
EMERGENCY CLOSE PHIGS (3P)
INQUIRE ARCHIVE STATE VALUE (3P)
INQUIRE STRUCTURE STATE VALUE (3P)
INQUIRE WORKSTATION STATE VALUE (3P)

modified 2 April 1993
<table>
<thead>
<tr>
<th>NAME</th>
<th>INQUIRE TEXT EXTENT – obtain extent rectangle for text string</th>
</tr>
</thead>
</table>
| SYNOPSIS | **void**<br> void pinq_text_extent ( wst, font, exp, sp, ht, path, hor, ver, str, error_ind, rect, offset )<br>**C Syntax**<br>Pint wst; /* workstation type**<br>Pint font; /* text font**<br>Pfloat exp; /* char expansion factor**<br>Pfloat sp; /* char spacing**<br>Pfloat ht; /* char height**<br>Ptext_path path; /* text path**<br>Phor_text_align hor; /* horizontal alignment**<br>Pvert_text_align ver; /* vertical alignment**<br>char *str; /* text string**<br>Pint *error_ind; /* OUT error indicator**<br>Prect *rect; /* OUT extent rectangle**<br>Ppoint *offset; /* OUT concatenation offset**<br>**FORTRAN Syntax**<br>SUBROUTINE pqtxx ( WKTYPE, FONT, CHXP, CHSP, CHH, TXP, TXALH, TXALV, STR, ERRIND, TXEXRX, TXEXRY, COPX, COPY )<br>INTEGER WKTYPE /* workstation type**<br>INTEGER FONT /* text font**<br>REAL CHXP /* character expansion factor**<br>REAL CHSP /* character spacing**<br>REAL CHH /* character height**<br>INTEGER TXP /* text path**<br>INTEGER TXALH /* text alignment horizontal**<br>INTEGER TXALV /* text alignment vertical**<br>CHARACTER*(*) STR /* character string**<br>INTEGER ERRIND /* OUT error indicator**<br>REAL TXEXRX(2), TXEXRY(2) /* OUT text extent rectangle**<br>REAL COPX, COPY /* OUT concatenation offset**<br>**FORTRAN Subset Syntax**<br>SUBROUTINE pqtxxs ( WKTYPE, FONT, CHXP, CHSP, CHH, TXP, TXALH, TXALV, LSTR, STR, ERRIND, TXEXRX, TXEXRY, COPX, COPY )<br>INTEGER WKTYPE /* workstation type**<br>INTEGER FONT /* text font**<br>REAL CHXP /* character expansion factor**<br>REAL CHSP /* character spacing**<br>REAL CHH /* character height**<br>INTEGER TXP /* text path**<br>INTEGER TXALH /* text alignment horizontal**<br>INTEGER TXALV /* text alignment vertical**<br>INTEGER LSTR /* length of string (in characters)**<br>modified 2 April 1993
INQUIRE TEXT EXTENT calculates the extent (bounding rectangle) and concatenation offset for a specified ASCII text string and set of PHIGS text attributes.

**C Input Parameters**

- **wst** The workstation type to use to resolve the font.
- **font** The text font, specified as an index to the workstation’s non-writable table of available fonts. All the font indices have named constants defined in phigs.h. See INTRO INTERNATIONALIZATION (7P) for a list of valid font indices.
- **exp** A real value specifying the character expansion factor.
- **sp** The character spacing, specified as a real fraction of the font’s nominal character height.
- **ht** A real value specifying the character height.
- **path** The text path is one of the following enumerated values:

```c
typedef enum {
    PPATH_RIGHT, Right
    PPATH_LEFT, Left
    PPATH_UP, Up
    PPATH_DOWN, Down
} Ptext_path;
```

- **hor** The horizontal alignment. This is an enumerated value defined in phigs.h, and can be one of:

```c
typedef enum {
    PHOR_NORM, Normal
    PHOR_LEFT, Left
    PHOR_CTR, Center
    PHOR_RIGHT, Right
} Phor_text_align;
```

- **ver** The vertical alignment. This is an enumerated value defined in phigs.h, and can be one of:

```c
typedef enum {
    PVERT_NORM, Normal
    PVERT_TOP, Top
    PVERT_CAP, Cap
} Pvert_text_align;
```
INQUIRE TEXT EXTENT (3P)

PVERT_HALF, Half
PVERT_BASE, Base
PVERT_BOTTOM, Bottom

str A pointer to the text string.

C Output Parameters

error_ind A pointer to the location to store the error number of any error that this function detects.

rect A pointer to a Prect structure in which to return the text extent. Prect is defined in phigs.h as:

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

rect is defined in phigs.h as:

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

offset A pointer to a Ppoint structure in which to return the concatenation offset. See the Execution section below for its meaning.

FORTRAN Input Parameters

WKTYPE The workstation type to use to resolve the font.

FONT The font number. See INTRO INTERNATIONALIZATION for the valid fonts.

CHXP The character expansion factor. See SET CHARACTER EXPANSION FACTOR for the valid values.

CHSP The character spacing. See SET CHARACTER SPACING for the valid values.

CHH The character height. See SET CHARACTER HEIGHT for the valid values.

TXP The text path. See SET TEXT PATH for the valid values.

TXALH The horizontal alignment. See SET TEXT ALIGNMENT for the valid values.

TXALV The vertical alignment. See SET TEXT ALIGNMENT for the valid values.

LSTR The length of text string (in characters).

STR The text string.
FORTRAN Output Parameters

**ERRIND**
The error indicator. See the *Execution* section below for a description of its use. See the **ERRORS** section below for the possible values it may return.

**TXEXRX, TXEXRY**
The text extent. See the *Execution* section below for its description.

- **TXERX(1)** is the lower left x coordinate.
- **TXERY(1)** is the lower left y coordinate.
- **TXERX(2)** is the upper right x coordinate.
- **TXERY(2)** is the upper right y coordinate.

**COPX, COPY**
The concatenation offset. See the *Execution* section below for its description.

Execution

The extent of the specified character string in the local 2D text coordinate system is computed using the specified text attributes for the specified workstation type. *STROKE* precision is assumed. The text position is (0,0) in the text local coordinate system. See TEXT 3 for a description of the text local coordinate system and how it is defined. The text extent encloses the entire text string with the specified attributes applied.

The concatenation offset indicates the text position for the concatenation of a subsequent text output primitive in the local 2D text coordinate system. This includes, for TEXT PATHS RIGHT and LEFT, an adjustment to account for the intercharacter spacing of the last character as specified by the character spacing parameter. It will be necessary for the application to apply a suitable modelling transformation to account for the CHARACTER UP VECTOR if it is other than the default.

If an error is detected by this function then the *error indicator* will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the *error indicator* will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 051 Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type
- 052 Ignoring function, workstation type not recognized by the implementation
- 062 Ignoring function, this information is not available for this MO workstation type
- 106 Ignoring function, the specified font is not available for the requested text precision on the specified workstation

SEE ALSO

TEXT (3P)
ANNOTATION TEXT RELATIVE (3P)
ESCAPE -12 (3P)

modified 2 April 1993
GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
NAME

INQUIRE TEXT FACILITIES – obtain list of workstation text facilities

SYNOPSIS

C Syntax

```c
void
pinq_text_facs ( type, length, start, error_ind, facilities, total_length )
Pint type;                           workstation type
Pint length;                        length of application list
Pint start;                         starting position
Pint *error_ind;                    OUT error indicator
Ptext_facs *facilities;             OUT text facilities
Pint *total_length;                 OUT length of list in PHIGS
```

FORTRAN Syntax

```fortran
SUBROUTINE pqtsx ( WTYPE, N, ERRIND, NFPP, FONT, PREC, NCHH, MINCHH, MAXCHH, NCHX, MINCHX, MAXCHX, NPTXI )
INTEGER WTYPE                           workstation type
INTEGER N                               list element requested
INTEGER ERRIND                          OUT error indicator
INTEGER NFPP                            OUT number of text font and precision pairs
INTEGER FONT                            OUT Nth element of list of text fonts
INTEGER PREC                            OUT Nth element of list of text precisions
INTEGER NCHH                            OUT number of available character heights
REAL MINCHH                            OUT minimum character height (DC)
REAL MAXCHH                            OUT maximum character height (DC)
INTEGER NCHX                            OUT number of available character expansion factors
REAL MINCHX                            OUT minimum character expansion factor
REAL MAXCHX                            OUT maximum character expansion factor
INTEGER NPTXI                           OUT number of predefined text indices
```

Required PHIGS Operating States

( PHOP, *, *, * )

DESCRIPTION

Purpose

Use INQUIRE TEXT FACILITIES to obtain a list of the text facilities available on the specified workstation type.

C Input Parameters

- `type` Workstation type.
- `length` The `length` is the number of text font and precision `font_precs` pairs which the `facilities` output parameter can return in the allocated memory space. A length of zero may be specified in order to have `total_length` return the total number of elements in the workstation list of available font and precision pairs.
- `start` Starting position in the list of font and precision pairs in the workstation description table at which to begin the inquiry.

modified 2 April 1993
C Output Parameters

**error_ind**
A pointer to the location to store the error number of any error that this function detects.

**facilities**
A pointer to a Ptext_facs data structure in which the system returns a listing of the text facilities available on this type of workstation. Ptext_facs is defined in phigs.h as:

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

Ptext_font_prec is defined in phigs.h as follows:

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

Ptext_prec is an enumerated type with the following values:

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

**total_length**
An integer pointer in which the system returns the total number of items in the list of available font and precision pairs on the workstation description table. This is the value required for length if all the items in the workstation list are to be returned.
The application must allocate memory for the text facilities list of fonts and precisions.

**FORTRAN Input Parameters**

- **WTYPE** Get the polymarker facilities for this workstation type.
- **N** Get the Nth element from the list of font and precision pairs.

**FORTRAN Output Parameters**

- **ERRIND** The error number of any error that this function detects.
- **NFPP** The number of available font and precision pairs.
- **FONT** The Nth font from the list of available font and precision pairs.
- **PREC** The Nth precision from the list of available font and precision pairs.
- **NCHH** The number of available character heights.
- **MINCHH** The minimum character height, in Device Coordinates (DC).
- **MAXCHH** The maximum character height, in (DC).
- **NCHX** The number of available character expansion factors.
- **MINCHX** The minimum character expansion factor.
- **MAXCHX** The maximum character expansion factor.
- **NPTXI** The number of predefined text indices.

**ERRORS**

- **002** Ignoring function, function requires state (PHOP, *, *, *)
- **051** Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- **052** Ignoring function, workstation type not recognized by the implementation
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- **062** Ignoring function, this information is not available for this MO workstation type

**SEE ALSO**

INQUIRE PREDEFINED TEXT REPRESENTATION (3P)
**NAME**  
INQUIRE TEXT REPRESENTATION – obtain text representation on workstation

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pqtxr ( WKID, TXI, TYPE, ERRIND, FONT, PREC, CHXP, CHSP, COLI )
INTEGER WKID      // workstation identifier
INTEGER TXI       // text index
INTEGER TYPE      // type of returned values (PSET, PREALI)
INTEGER ERRIND    // OUT error indicator
INTEGER FONT      // OUT text font
INTEGER PREC      // OUT text precision (PSTRP, PCHARP, PSTRKP)
REAL CHXP         // OUT character expansion factor
REAL CHSP         // OUT character spacing
INTEGER COLI      // OUT text colour index
```

**Required PHIGS Operating States**  
(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE TEXT REPRESENTATION to determine the current attribute values for a specified entry in a specified workstation’s table of defined text representations.

See the description of the subroutine SET TEXT REPRESENTATION for information on the meaning of these attribute values.

**C Input Parameters**

- `ws`  
  Workstation identifier.

- `index`  
  Entry to be returned from the workstation’s table of text representations; if this entry is not present in the table and the `type` parameter is REALIZED, the representation for text index 1 is returned.

- `type`  
  An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

```c
typedef enum {
    PINQ_SET,
    // Return application-specified value
```
INQUIRE TEXT REPRESENTATION (3P)

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

rep
A pointer to a Ptext_bundle structure in which the system returns the text representation at index in the workstation’s table of text representations.

Ptext_bundle is defined in phigs.h as follows:

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

Ptext_prec is defined in phigs.h as:

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

FORTRAN Input Parameters

WKID    Workstation identifier.

TXI
Entry to be returned from the workstation’s table of text representations; if this entry is not present in the table and the type of returned value parameter is REALIZED, the representation for text index 1 is returned.

TYPE
An enumerated value specifying whether the inquired values are to be returned as the values originally specified by the application (SET), or as the values actually being used by the workstation if any of the application-specified values had to be mapped to ones available on the workstation (REALIZED). Valid values are defined in phigs.h as:

```c
PSET Return application-specified value
PREALI Return value available on the workstation
```

FORTRAN Output Parameters

ERRIND
The error number of any error that this function detects.

FONT
The text font at index TXI in the workstation’s table of text representations.

PREC
The text precision at index TXI in the workstation’s table of text representations.

CHXP
The character expansion factor at index TXI in the workstation’s table of text representations.

modified 2 April 1993
**CHSP**  The character spacing at index TXI in the workstation’s table of text representations.

**COLI**  The text colour index at index TXI in the workstation’s table of text representations.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
059  Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100  Ignoring function, the bundle index value is less than one
101  Ignoring function, the specified representation has not been defined
134  Ignoring function, the requested entry contains a general colour specification with *colour type* other than INDIRECT.

**SEE ALSO**

INQUIRE PREDEFINED TEXT REPRESENTATION (3P)
SET TEXT REPRESENTATION (3P)
INQUIRE LIST OF TEXT INDICES (3P)
INQUIRE TEXT REPRESENTATION PLUS (3PP)

---

modified 2 April 1993
NAME
INQUIRE VALUATOR DEVICE STATE—obtain state of valuator device

SYNOPSIS
C Syntax

void pinq_val_st ( ws, dev, store, err, op_mode, echo_switch, init_value, prompt_echo, echo_area, val_data )

Pint ws; workstation identifier
Pint dev; valuator device number
Pstore store; handle to store object
Pint *err; OUT error indicator
Pop_mode *op_mode; OUT operating mode
Pecho_switch *echo_switch; OUT echo switch
Pfloat *init_value; OUT initial value
Pint *prompt_echo; OUT prompt/echo type
Plimit *echo_area; OUT echo area
Pval_data **val_data; OUT data record

FORTRAN Syntax

SUBROUTINE pqvls ( WKID, VLDNR, MLDR, ERRIND, MODE, ESW, IVAL, PET, EAREA, LDR, DATREC )

INTEGER WKID workstation identifier
INTEGER VLDNR valuator device number
INTEGER MLDR dimension of data record array
INTEGER ERRIND OUT error indicator
INTEGER MODE OUT operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW OUT echo switch (PNECHO, PECHO)
REAL IVAL OUT initial value
INTEGER PET OUT prompt/echo type
REAL EAREA(4) OUT echo area in device coordinates
INTEGER LDR OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR) OUT data record

Required PHIGS Operating States
( PHOP, WSOP, *, * )

DESCRIPTION
Purpose
Use INQUIRE VALUATOR DEVICE STATE to determine the current state of the specified valuator device.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The valuator device data record

modified 2 April 1993
within the store buffer is accessed via the pointer pointed to by val_data.

\textit{ws}  
Workstation identifier. An integer specifying the workstation with which the specified valuator device is associated.

\textit{dev}  
The device number of the valuator device. See the \textit{AVAILABLE DEVICES} section of \textit{INITIALIZE VALUATOR} for a description of the available devices.

\textit{store}  
The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see \texttt{CREATE STORE (3P)}).

\textbf{C Output Parameters}

\textit{err}  
A pointer to the location to store the error number of any error that this function detects.

\textit{op\_mode}  
The operating mode. \texttt{Pop\_mode} is an enumerated type defined in \texttt{phigs.h} with the following values:

\begin{verbatim}
typedef enum {
    POP_REQ,    
    POP_SAMPLE, 
    POP_EVENT
} Pop_mode;
\end{verbatim}

\textit{echo\_switch}  
The echo state. \texttt{Pecho\_switch} is an enumerated type defined in \texttt{phigs.h} with the following values:

\begin{verbatim}
typedef enum {
    PSWITCH_NO_ECHO, 
    PSWITCH_ECHO
} Pecho_switch;
\end{verbatim}

\textit{init\_value}  
Initial value.

\textit{prompt\_echo}  
The prompt/echo type desired. See the \textit{AVAILABLE DEVICES} section of \textit{INITIALIZE VALUATOR} for a description of the available types.

\textit{echo\_area}  
A pointer to a variable of type \texttt{Plimit} that contains the echo area of the device. \texttt{Plimit} is defined in \texttt{phigs.h} as follows:

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

760 modified 2 April 1993
val_data

Pointer to a pointer that points to the valuator device state within store.

Pval_data is defined in phigs.h as follows:

```c
typedef struct {
    float low;    /* low range limit */
    float high;   /* high range limit */
    union {
        struct {
            int unused;
        } pet_r1;
        struct {
            char *label;
            char *format;
            char *low_label;
            char *high_label;
        } pet_u1;
    } pets;
} Pval_data;
```

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial value, the valuator range limits, the prompt/echo type, and the echo area.

Error 2001 is returned if MLDR is too small, but not if it is zero.

WKID  The workstation identifier of the workstation associated with the device.

STDNR  The device number of the VALUATOR device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR for a description of the available devices.

MLDR  The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it may return.

MODE  The operating mode.

ESW  The echo switch.

VAL  The initial value.

modified 2 April 1993 761
PET  The prompt/echo type.

EAREA  An array containing the limits of the echo area, XMIN, XMAX, YMIN, YMAX, in Device Coordinates.

LDR  The required dimension of the data record array, DATREC.

DATREC  The data record array.

Execution  INQUIRE VALUATOR DEVICE STATE returns the current state of the specified valuator device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial value, prompt/echo type, echo area, and data record. See SET VALUATOR MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE VALUATOR for a description of the initial valuator, prompt/echo type, echo area, and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function, the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, the error indicator will be set to zero and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS 003  Ignoring function, function requires state (PHOP, WSOP, *, *)

054  Ignoring function, the specified workstation is not open

061  Ignoring function, specified workstation is not category INPUT or OUTIN

250  Ignoring function, the specified device is not available on the specified workstation

SEE ALSO  INITIALIZE VALUATOR (3P)

SET VALUATOR MODE (3P)

INQUIRE VALUATOR DEVICE STATE 3 (3P)
NAME
INQUIRE VALUATOR DEVICE STATE 3 – obtain state of valuator device

SYNOPSIS
C Syntax
void
pinq_val_st3 ( ws, dev, store, err, op_mode, echo_switch, init_value, prompt_echo,
echo_vol, val_data )

Pint ws;  // workstation identifier
Pint dev;  // valuator device number
Pstore store;  // handle to store object
Pint *err;  // OUT error indicator
Pop_mode *op_mode;  // OUT operating mode
Pecho_switch *echo_switch;  // OUT echo switch
Pfloat *init_value;  // OUT initial value
Pint *prompt_echo;  // OUT prompt/echo type
Plimit3 *echo_vol;  // OUT echo area
Pval_data3 **val_data;  // OUT data record

FORTRAN Syntax
SUBROUTINE pqvls3 ( WKID, VLDNR, MLDR, ERRIND, MODE, ESW, IVAL,
PET, EVOL, LDR, DATREC )

INTEGER WKID  // workstation identifier
INTEGER VLDNR  // valuator device number
INTEGER MLDR  // dimension of data record array
INTEGER ERRIND  // OUT error indicator
INTEGER MODE  // OUT operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW  // OUT echo switch (PNECHO, PECHO)
REAL IVAL  // OUT initial value
INTEGER PET  // OUT prompt/echo type
REAL EVOL(6)  // OUT echo volume in device coordinates
INTEGER LDR  // OUT number of array elements used in data record
CHARACTER*80 DATREC(MLDR)  // OUT data record

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE VALUATOR DEVICE STATE 3 to determine the current state of the specified
valuator device.

C Input Parameters
Applications using the C binding must create a buffer to be used by this function as
memory space for storing data associated with the device state. This buffer is passed as the
store argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for
creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that
there is sufficient memory for the specific inquiry. The valuator device data record

modified 2 April 1993

763
within the store buffer is accessed via the pointer pointed to by \textit{val_data}.

\textbf{ws} Workstation identifier. An integer specifying the workstation with which the specified valuator device is associated.

\textbf{dev} The device number of the valuator device. See the \textit{AVAILABLE DEVICES} section of \texttt{INITIALIZE VALUATOR} for a description of the available devices.

\textbf{store} The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see \texttt{CREATE STORE (3P)}).

\textbf{C Output Parameters}

\textbf{err} A pointer to the location to store the error number of any error that this function detects.

\textbf{op\_mode} The operating mode. \texttt{Pop\_mode} is an enumerated type defined in \texttt{phigs.h} with the following values:

\begin{verbatim}
typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
} Pop_mode;
\end{verbatim}

\textbf{echo\_switch} The echo state. \texttt{Pecho\_switch} is an enumerated type defined in \texttt{phigs.h} with the following values:

\begin{verbatim}
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
\end{verbatim}

\textbf{init\_value} Initial value.

\textbf{prompt\_echo} The prompt/echo type desired. See the \textit{AVAILABLE DEVICES} section of \texttt{INITIALIZE VALUATOR} for a description of the available types.

\textbf{echo\_vol} A pointer to a \texttt{Plimit3} structure defining the \texttt{x}, \texttt{y}, and \texttt{z} components of the echo volume, in Device Coordinates. \texttt{Plimit3} is defined in \texttt{phigs.h} as follows:

\begin{verbatim}
typedef struct {
    Pfloat x_min; /* minimum x coordinate value */
    Pfloat x_max; /* maximum x coordinate value */
    Pfloat y_min; /* minimum y coordinate value */
    Pfloat y_max; /* maximum y coordinate value */
    Pfloat z_min; /* minimum z coordinate value */
    Pfloat z_max; /* maximum z coordinate value */
} Plimit3;
\end{verbatim}

\texttt{modi\_fied 2 April 1993}
val_data

Pointer to a pointer that points to the pick device state within store. Pval_data3 is defined in phigs.h as follows:

typedef struct {
    Pfloat low; /* low range limit */
    Pfloat high; /* high range limit */
    union {
        struct {
            Pint unused;
        } pet_r1;
        struct {
            char *label;
            char *format;
            char *low_label;
            char *high_label;
        } pet_u1;
    } pets;
} Pval_data3;

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which will be placed the contents of the device’s input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD. The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS will return the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values will be returned. These are LDR, the operating mode, the echo switch, the initial value, the valuator range limits, the prompt/echo type, and the echo volume.

Error 2001 is returned if MLDR is too small, but not if it is zero.

WKID The workstation identifier of the workstation associated with the device.

VTDNR The device number of the VALUATOR device. See the AVAILABLE DEVICES section of INITIALIZE VALUATOR 3 for a description of the available devices.

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

MODE The operating mode.

ESW The echo switch.
INQUIRE VALUATOR DEVICE STATE 3 (3P)  SunPHIGS Release 3.0

| VAL       | The initial value. |
| PET       | The prompt/echo type. |
| EVOL      | An array containing the limits of the echo volume, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX. |
| LDR       | The required dimension of the data record array, DATREC. |
| DATREC    | The data record array. |

**Execution**

INQUIRE VALUATOR DEVICE STATE 3 returns the current state of the specified valuator device, which is stored in the workstation state list of the workstation associated with the device. The current state includes the operating mode, echo switch, initial value, prompt/echo type, echo volume, and data record. See SET VALUATOR MODE for a description of the operating mode and the echo switch and how to set these values. See INITIALIZE VALUATOR 3 for a description of the initial valuator, prompt/echo type, echo volume, and data record contents and how to set these values.

Except in the cases mentioned in the C and FORTRAN Parameters sections above, if an error is detected by this function, then the error indicator will indicate the error number of the error detected and no other output data will be returned. If no error is detected, then the error indicator will be set to zero, and the inquired information will be available in the output parameters. Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, specified workstation is not of category INPUT or OUTIN
250  Ignoring function, the specified device is not available on the specified workstation

**SEE ALSO**

INITIALIZE VALUATOR 3 (3P)
SET VALUATOR MODE (3P)
INQUIRE VALUATOR DEVICE STATE (3P)
| NAME | INQUIRE VIEW FACILITIES – obtain view facilities of a specified workstation type |
| SYNOPSIS |  |
| C Syntax | void pinq_view_facs ( type, error_ind, num ) |
| | Pint type; workstation type |
| | Pint *error_ind; OUT error indicator |
| | Pint *num; OUT number of predefined view indices |
| FORTRAN Syntax | SUBROUTINE pqwf ( WTYPE, ERRIND, NPVWI ) |
| | INTEGER WTYPE workstation type |
| | INTEGER ERRIND OUT error indicator |
| | INTEGER NPVWI OUT number of predefined view indices |
| Required PHIGS Operating States | (PHOP, *, *, *) |
| DESCRIPTION | Use INQUIRE VIEW FACILITIES to determine the number of predefined view representations supported by a specified workstation type. |
| Purpose |  |
| C Input Parameter | type Workstation type being queried. |
| C Output Parameters | error_ind A pointer to the location to store the error number of any error that this function detects. |
| | num An pointer to a location in which the system returns the number of predefined view indices for the workstation. |
| FORTRAN Input Parameter | WTYPE Workstation type being queried. |
| FORTRAN Output Parameters | ERRIND The error number of any error that this function detects. |
| | NPVWI The number of predefined view indices for the workstation. |
| ERRORS |  |
| | 002 Ignoring function, function requires state (PHOP, *, *, *) |
| | 052 Ignoring function, workstation type not recognized by the implementation |
| | 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type |

modified 2 April 1993
057 Ignoring function, specified workstation is of category MI
062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

INQUIRE PREDEFINED VIEW REPRESENTATION (3P)
NAME

INQUIRE VIEW REPRESENTATION – obtain view representation on workstation

SYNOPSIS

C Syntax

void
pinq_view_rep ( ws, view_index, error_ind, update_state, cur_rep, req_rep )

Pint ws;  
workstation identifier
Pint view_index;  
view index
#define error_ind;  
OUT error indicator
Pupd_st update_state;  
OUT transformation update state
Pview_rep3 cur_rep;  
OUT current view representation
Pview_rep3 req_rep;  
OUT requested view representation

FORTRAN Syntax

SUBROUTINE pqvwr ( WKID, VIEWI, CURQ, ERRIND, VWUPD, VWORMT,
VWMPMT, VWCPLM, XYCLPI, BCLIPI, FCLIPI )

INTEGER WKID  
workstation identifier
INTEGER VIEWI  
view index requested
INTEGER CURQ  
whether current or requested values are to be returned
(PCURVL, PRQSVL)
INTEGER ERRIND  
OUT error indicator
INTEGER VWUPD  
OUT viewing transformation update state
REAL VWORMT(4, 4)  
OUT view orientation matrix
REAL VWMPMT(4, 4)  
OUT view mapping matrix
REAL VWCPLM(6)  
OUT view clipping limits (NPC)
INTEGER XYCLPI  
OUT x-y clipping indicator (PNCLIP, PCLIP)
INTEGER BCLIPI  
OUT back clipping indicator (PNCLIP, PCLIP)
INTEGER FCLIPI  
OUT front clipping indicator (PNCLIP, PCLIP)

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

Use INQUIRE VIEW REPRESENTATION to obtain the current and requested view representation in an entry of the specified workstation’s view table.

C Input Parameters

ws
The workstation identifier of the workstation whose state list is queried.

view_index
The view index whose representation is desired from the workstation’s table of defined view representations.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

update_state
A pointer to a location in which the system returns the view transformation update

modified 2 April 1993

769
state. Pupd_st is an enumerated type defined in phigs.h as follows: +1.75i
typedef enum {
  PUPD_NOT_PEND, Not pending
  PUPD_PEND, Pending
} Pupd_st;

cur_rep A pointer to a Pview_rep3 structure in which the system returns the definition of
the current view representation.

req_rep A pointer to a Pview_rep3 structure in which the system returns the definition of
the requested view representation.
Pview_rep3 is defined in phigs.h as follows:
typedef struct {
  Pmatrix3 ori_matrix; /* view orientation matrix */
  Pmatrix3 map_matrix; /* mapping matrix */
  Plimit3 clip_limit; /* clipping limits */
  Pclip_ind xy_clip; /* X-Y clipping indicator */
  Pclip_ind back_clip; /* back clipping indicator */
  Pclip_ind front_clip; /* front clipping indicator */
} Pview_rep3;
The Pmatrix3 type definition is a 4x4 matrix, defined in phigs.h as:
typedef Pfloat Pmatrix3[4][4];
The Plimit3 structure used to define the clipping limits is defined in phigs.h as follows:
typedef struct {
  Pfloat x_min; /* x min */
  Pfloat x_max; /* x max */
  Pfloat y_min; /* y min */
  Pfloat y_max; /* y max */
  Pfloat z_min; /* z min */
  Pfloat z_max; /* z max */
} Plimit3;
Pclip_ind is an enumerated type defined in phigs.h as follows:
typedef enum {
  PIND_CLIP, Clipping
  PIND_NO_CLIP, Not clipping
} Pclip_ind;
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th><strong>WKID</strong></th>
<th>The <em>workstation identifier</em> of the workstation whose state list is queried.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VIEWI</strong></td>
<td>The <em>view index</em> of the entry whose representation is desired.</td>
</tr>
</tbody>
</table>
| **CURQ** | An enumerated variable specifying whether the *current* or *requested* representation is desired:  
  - **PCURVL**: Return the current value of the representation  
  - **PRQSVL**: Return the requested value of the representation  
  The requested values can differ from the current values if the update state is *Pending*. |

### FORTRAN Output Parameters

| **ERRIND** | The error number of any error that this function detects. |
| **VWUPD** | The *view transformation update state*. Possible values are: **PNPEND** (*Not Pending*) or **PPEND** (*Pending*). |
| **VWORMT** | A 4×4 array in which the view orientation matrix is returned. |
| **VWMPMT** | A 4×4 array in which the view mapping matrix is returned. |
| **VWCPLM** | An array in which the view clipping limits are returned; the first two elements of this array give the minimum and maximum clipping values in NPC for x, the next two for y, and the last two for z. |
| **XYCLPI** | The X-Y clipping indicator, one of: **PNCLIP** (*Not Clipping*) or **PCLIP** (*Clipping*). |
| **BCLIPI** | The back clipping indicator. |
| **FCLIPI** | The front clipping indicator. |

### ERRORS

| **003** | Ignoring function, function requires state (PHOP, WSOP, *, *) |
| **054** | Ignoring function, the specified workstation is not open |
| **057** | Ignoring function, specified workstation is of category MI |
| **114** | Ignoring function, the view index value is less than zero |
| **101** | Ignoring function, the specified representation has not been defined |

### SEE ALSO

- **INQUIRE PREDEFINED VIEW REPRESENTATION** (3P)
- **SET VIEW INDEX** (3P)
- **SET VIEW REPRESENTATION 3** (3P)
- **INQUIRE VIEW FACILITIES** (3P)
**NAME**

INQUIRE WORKSTATION CATEGORY – obtain category of specified workstation type

**SYNOPSIS**

**C Syntax**

```c
void pinq_ws_cat ( type, error_ind, category )
Pint type;  // workstation type
Pint *error_ind;  // OUT error indicator
Pws_cat *category;  // OUT workstation category
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pqwkca ( WTYPE, ERRIND, WKCAT )
INTEGER WTYPE  // workstation type
INTEGER ERRIND  // OUT error indicator
INTEGER WKCAT  // OUT workstation category
```

**Required PHIGS Operating States**

(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**

Use INQUIRE WORKSTATION CATEGORY to determine the category of a specified workstation type.

**C Input Parameters**

- `type`  Type of workstation.

**C Output Parameters**

- `error_ind`  A pointer to the location to store the error number of any error that this function detects.
- `category`  A pointer to a Pws_cat enumerated variable, in which the workstation category is returned. Values for Pws_cat are defined in phigs.h as follows:

```c
typedef enum {
    PCAT_OUT,  // Output only
    PCAT_IN,  // Input only
    PCAT_OUTIN,  // Output and input
    PCAT_MO,  // Metafile output
    PCAT_MI  // Metafile input
} Pws_cat;
```

**FORTRAN Input Parameters**

- `WTYPE`  Type of workstation.
FORTRAN Output Parameters

**ERRIND**
The error number of any error that this function detects.

**WKCAT**
The workstation category, defined in phigs77.h as follows:

- **POUTPT**  Output Only
- **PINPUT**  Input Only
- **POUTIN**  Output and Input
- **PMO**  Metafile Output
- **PMI**  Metafile Input

**ERRORS**

- **002**  Ignoring function, function requires state (PHOP, *, *, *)
- **052**  Ignoring function, workstation type not recognized by the implementation
- **051**  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

**SEE ALSO**

- PHIGS WORKSTATION DESCRIPTION TABLE (7P)
- OPEN WORKSTATION (3P)
- INQUIRE WORKSTATION CLASSIFICATION (3P)
NAME
INQUIRE WORKSTATION CLASSIFICATION – obtain classification of specified workstation type

SYNOPSIS
C Syntax
void
pinq_ws_class ( type, error_ind, class )

Pint type;  // workstation type
Pint *error_ind;  // OUT error indicator
Pws_class *class;  // OUT workstation class

FORTRAN Syntax
SUBROUTINE pqwkcl ( WTYPE, ERRIND, VRTYPE )
INTEGER WTYPE  // workstation type
INTEGER ERRIND  // OUT error indicator
INTEGER VRTYPE  // OUT vector/raster/other type

Required PHIGS Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE WORKSTATION CLASSIFICATION to determine the classification of a specified workstation type.

C Input Parameters
 type  // Type of workstation.

C Output Parameters
 error_ind  // A pointer to the location to store the error number of any error that this function detects.

class  // A pointer to a Pws_class enumerated variable in which the system returns the workstation class. Values for Pws_class are defined in phigs.h as follows:

typedef enum {
PCLASS_VEC,  // Vector device
PCLASS_RASTER,  // Raster device
PCLASS_OTHER  // Other device
} Pws_class;

FORTRAN Input Parameters
 WTYPE  // Type of workstation.

FORTRAN Output Parameters
 ERRIND  // The error number of any error that this function detects.

VRTYPE  // The workstation class, defined in phigs77.h as follows:

 PVECTR  // Vector device

modified 2 April 1993
PRASTR  Raster device
POTHWK  Other device

ERRORS
002  Ignoring function, function requires state (PHOP, *, *, *)
052  Ignoring function, workstation type not recognized by the implementation
051  Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
059  Ignoring function, the specified not does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062  Ignoring function, this information is not available for this MO workstation type

SEE ALSO
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
INQUIRE WORKSTATION CATEGORY (3P)
INQUIRE DISPLAY SPACE SIZE (3P)
NAME INQUIRE WORKSTATION CONNECTION AND TYPE—obtain connection identifier and type of specified workstation

SYNOPSIS
C Syntax

```c
void
pqinc_ws_conn_type ( ws, store, error_ind, conn_id, ws_type )
```

<table>
<thead>
<tr>
<th>C Input Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ws</code></td>
<td>The workstation identifier of the workstation in question.</td>
</tr>
<tr>
<td><code>store</code></td>
<td>The memory buffer PHIGS is to use for storing the connection identifier information. This buffer must exist prior to calling this function (see CREATE STORE (3P)).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C Output Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>error_ind</code></td>
<td>A pointer to the location to store the error number of any error that this function detects.</td>
</tr>
<tr>
<td><code>conn_id</code></td>
<td>A pointer to a pointer to the connection identifier.</td>
</tr>
<tr>
<td><code>ws_type</code></td>
<td>A pointer to a Pint variable in which to return the workstation type.</td>
</tr>
</tbody>
</table>

FORTRAN Syntax

```fortran
SUBROUTINE pqwkc ( WKID, ERRIND, CONID, WTYPE )
```

<table>
<thead>
<tr>
<th>Required PHIGS Operating States</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(PHOP, WSOP, *, *)</code></td>
<td>Use INQUIRE WORKSTATION CONNECTION AND TYPE to inquire about the specific workstation type and connection identifier associated with a given workstation. The connection identifier and workstation type are associated with a workstation when it is opened. At that time a specific workstation type is created, based on the generic type supplied by the caller, and the specific workstation type and the user-supplied connection identifier are stored in the workstation’s state list. This function returns that specific workstation type and the connection identifier. See OPEN WORKSTATION for descriptions of the possible connection identifiers and generic workstation types.</td>
</tr>
</tbody>
</table>

| `ws`    | The workstation identifier of the workstation in question. |
| `store` | The memory buffer PHIGS is to use for storing the connection identifier information. This buffer must exist prior to calling this function (see CREATE STORE (3P)). |
| `error_ind` | A pointer to the location to store the error number of any error that this function detects. |
| `conn_id` | A pointer to a pointer to the connection identifier. |
| `ws_type` | A pointer to a Pint variable in which to return the workstation type. |
FORTRAN Input Parameters

WKID  The workstation identifier of the workstation in question.

FORTRAN Output Parameters

ERRIND  The error indicator. See the Execution section below for a description of its use. See the ERRORS section below for the possible values it can return.

CONID  The connection id of the workstation.

WTYPE  The specific workstation type associated with the workstation.

Execution

INQUIRE WORKSTATION CONNECTION AND TYPE returns the specific workstation type and connection identifier associated with the specified workstation. See OPEN WORKSTATION for a description of the possible connection identifiers and workstation types.

ERRORS

003  Ignoring function, function requires state (PHOP, WSOP, *, *)

054  Ignoring function, the specified workstation is not open

SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES (3P)
OPEN WORKSTATION (3P)
WORKSTATION TYPE CREATE (3P)
WORKSTATION TYPE GET (3P)
WORKSTATION TYPE SET (3P)
**NAME**
INQUIRE WORKSTATION STATE TABLE LENGTHS – obtain lengths of workstation state tables for specified workstation type

**SYNOPSIS**

### C Syntax

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

### FORTRAN Syntax

```fortran
SUBROUTINE pqwksl ( WTYPE, ERRIND, MPLBTE, MPMBTE, MTXBTE, MINBTE, MEDBTE, MPAI, MCOLI, VWTBI )
INTEGER WTYPE  /* workstation type */
INTEGER ERRIND  /* OUT error indicator */
INTEGER MPLBTE  /* OUT maximum number of polyline bundle table entries */
INTEGER MPMBTE  /* OUT maximum number of polymarker bundle table entries */
INTEGER MTXBTE  /* OUT maximum number of text bundle table entries */
INTEGER MINBTE  /* OUT maximum number of interior bundle table entries */
INTEGER MEDBTE  /* OUT maximum number of edge bundle table entries */
INTEGER MPAI  /* OUT maximum number of pattern indices */
INTEGER MCOLI  /* OUT maximum number of colour indices */
INTEGER VWTBI  /* OUT maximum number of view table indices */
```

**Required PHIGS Operating States**
(PHOP, *, *, *)

**DESCRIPTION**

### Purpose
Use INQUIRE WORKSTATION STATE TABLE LENGTHS to determine the lengths of the state tables of a specified workstation type.

### C Input Parameters
- **type** Type of workstation.

### C Output Parameters
- **error_ind** A pointer to the location to store the error number of any error that this function detects.
- **lengths** A pointer to a Pws_st_tables data structure, which returns the length of the workstation state tables. Pws_st_tables is defined in phigs.h as:

```c
typedef struct {
    Pint line_bundles;  /* polyline tables */
    Pint mark_bundles;  /* polymarker tables */
    Pint text_bundles;  /* text tables */
    Pint int_bundles;  /* interior tables */
    Pint edge_bundles;  /* edge tables */
} Pws_st_tables;
```
Pint pat_reps; /* pattern tables */
Pint colr_reps; /* colour tables */
Pint view_reps; /* view tables */

} Pws_st_tables;

FORTRAN Input Parameters

WTYPE Get the workstation state table lengths for this workstation type.

FORTRAN Output Parameters

ERRIND The error number of any error that this function detects.

MPLBTE The maximum number of polyline bundle table entries.

MPMBTE The maximum number of polymarker bundle table entries.

MTXBTE The maximum number of text bundle table entries.

MINBTE The maximum number of interior bundle table entries.

MEDBTE The maximum number of edge bundle table entries.

MPAI The maximum number of pattern indices.

MCOLI The maximum number of colour indices.

VWTBI The maximum number of view table indices.

ERRORS

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

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
INQUIRE WORKSTATION STATE TABLE LENGTHS PLUS (3PP)
NAME  
INQUIRE WORKSTATION STATE VALUE – obtain workstation state value

SYNOPSIS
C Syntax
void
pinq_ws_st ( ws_state )
Pws_st *ws_state;  OUT workstation state

FORTRAN Syntax
SUBROUTINE pqwkst ( WKSTA )
INTEGER  WKSTA  OUT workstation state value (PWSCL, PWSOP)

Required PHIGS
Operating States
(*, *, *, *)

DESCRIPTION
Purpose
Use INQUIRE WORKSTATION STATE VALUE to determine the current state of the workstation.
The state value may be either workstation open (PWSOP) or workstation closed (PWSCL).
This function may be used to verify that OPEN WORKSTATION was successful before
performing additional PHIGS operations on the workstation.

C Output Parameters
ws_state
A pointer to an enumerated variable in which the current state of the workstation
is returned. Pws_st is defined in phigs.h as follows:

typedef enum {
    PWS_ST_WSCL,  Workstation closed
    PWS_ST_WSOP  Workstation open
} Pws_st;

FORTRAN Output
Parameters
WKSTA The current state of the workstation, one of the enumerated values:

    PWSCL  Workstation closed
    PWSOP  Workstation open

ERRORS
No Error

SEE ALSO
INQUIRE SET OF OPEN WORKSTATIONS (3P)
OPEN WORKSTATION (3P)
CLOSE WORKSTATION (3P)
INQUIRE SYSTEM STATE VALUE (3P)

780 modified 2 April 1993
NAME
INQUIRE WORKSTATION TRANSFORMATION—obtain 2D workstation transformation

SYNOPSIS
C Syntax
void
pinq_ws_tran ( ws, error_ind, upd_st, req_win_lim, cur_win_lim, req_vp_lim,
cur_vp_lim )

Pint ws;
Pint *error_ind;
Pupd_st *upd_st;
Plimit *req_win_lim;
Plimit *cur_win_lim;
Plimit *req_vp_lim;
Plimit *cur_vp_lim;

FORTRAN Syntax
SUBROUTINE pqwkt ( WKID, ERRIND, TUS, RWINDO, CWINDO,
RVIEWP, CVIEWP )
INTEGER WKID workstation identifier
INTEGER ERRIND OUT error indicator
INTEGER TUS OUT workstation transformation update state
REAL RWINDO(4) OUT requested workstation window in NPC
REAL CWINDO(4) OUT current workstation window in NPC
REAL RVIEWP(4) OUT requested workstation viewport in DC
REAL CVIEWP(4) OUT current workstation viewport in DC

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE WORKSTATION TRANSFORMATION to determine the current and requested 2D
workstation transformation from a specified workstation’s state list.

C Input Parameter
ws The workstation identifier of the workstation whose state list is queried.

C Output Parameters
error_ind
A pointer to the location to store the error number of any error that this function
detects.

upd_st The current update state. Pupd_st is defined in phigs.h as follows:
typedef enum {
    PUPD_NOT_PEND
    PUPD_PEND
} Pupd_st;

req_win_lim
The requested workstation window. Plimit is defined in phigs.h as follows:

mod 2 April 1993
typedef struct {
    float x_min;
    float x_max;
    float y_min;
    float y_max;
} Plimit;

**cur_win_lim**
The current workstation window. Plimit is defined above.

**req_vp_lim**
The requested workstation viewport. Plimit is defined above.

**cur_vp_lim**
The current workstation viewport. Plimit is defined above.

### FORTRAN Input Parameter

**WKID**

The *workstation identifier* of the workstation whose state list is queried.

### FORTRAN Output Parameters

**ERRIND**

The error number of any error that this function detects.

**TUS**

The workstation transformation update state, one of the values: PNPEND (*Not Pending*) or PPEND (*Pending*). 

**RWINDO**

An array where the four bounds of the requested workstation window are returned; the first two elements of this array give the minimum and maximum bounds in NPC for $x$, and the next two for $y$.

**CWINDO**

An array where the four bounds of the current workstation window are returned; the first two elements of this array give the minimum and maximum bounds in NPC for $x$, and the next two for $y$.

**RVIEWP**

An array where the four bounds of the requested workstation viewport are returned; the first two elements of this array give the minimum and maximum bounds in DC for $x$, and the next two for $y$.

**CVIEWP**

An array where the four bounds of the current workstation viewport are returned; the first two elements of this array give the minimum and maximum bounds in DC for $x$, and the next two for $y$.

### Execution

The requested and current entries may differ if a workstation transformation change has been requested, but has not yet been provided. The update state will be *Pending* in this case, and *Not Pending* otherwise.
ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
057 Ignoring function, specified workstation is of category MI

SEE ALSO

SET WORKSTATION VIEWPORT (3P)
SET WORKSTATION WINDOW (3P)
INQUIRE WORKSTATION TRANSFORMATION 3 (3P)
NAME
INQUIRE WORKSTATION TRANSFORMATION 3 – obtain 3D workstation transformation

SYNOPSIS
C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pqwkt3 ( WKID, ERRIND, TUS, RWINDO, CWINDO, RVIEWP, CVIEWP )
INTEGER WKID       /* workstation identifier */
INTEGER ERRIND    /* OUT error indicator */
INTEGER TUS       /* OUT workstation transformation update state */
REAL RWINDO(6)    /* OUT requested workstation window in NPC */
REAL CWINDO(6)    /* OUT current workstation window in NPC */
REAL RVIEWP(6)    /* OUT requested workstation viewport in DC */
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use INQUIRE WORKSTATION TRANSFORMATION 3 to determine the current and requested 3D transformation from a specified workstation’s state list.

C Input Parameter
```
ws    The workstation identifier of the workstation whose state list is queried.
```

C Output Parameters
```
error_ind
A pointer to the location to store the error number of any error that this function detects.

upd_st The current update state. Pupd_st is defined in phigs.h as follows:
```
typedef enum {
    PUPD_NOT_PEND,
    PUPD_PEND
} Pupd_st;
```

req_win_lim
The requested workstation window. Plimit3 is defined in phigs.h as follows:
```
typedef struct {
    Pfloat x_min;
    Pfloat x_max;
    Pfloat y_min;
    Pfloat y_max;
    Pfloat z_min;
    Pfloat z_max;
} Plimit3;

cur_win_lim
The current workstation window. Plimit3 is defined above.

req_vp_lim
The requested workstation viewport. Plimit3 is defined above.

cur_vp_lim
The current workstation viewport. Plimit3 is defined above.

FORTRAN Input
Parameter

WKID  The workstation identifier of the workstation whose state list is queried.

FORTRAN Output
Parameters

ERRIND  The error number of any error that this function detects.

TUS  The workstation transformation update state, one of the values: PNPEND (Not Pending) or PPEND (Pending).

RWINDO
An array where the six bounds of the requested workstation window are returned; the first two elements of this array give the minimum and maximum bounds in NPC for x, the next two for y, and the last two for z.

CWINDO
An array where the six bounds of the current workstation window are returned; the first two elements of this array give the minimum and maximum bounds in NPC for x, the next two for y, and the last two for z.

RVIEWP
An array where the six bounds of the requested workstation viewport are returned; the first two elements of this array give the minimum and maximum bounds in DC for x, the next two for y, and the last two for z.

CVIEWP
An array where the six bounds of the current workstation viewport are returned; the first two elements of this array give the minimum and maximum bounds in DC for x, the next two for y, and the last two for z.

Execution
The requested and current entries may differ if a workstation transformation change has been requested, but has not yet been provided. The update state will be Pending in this case, and Not Pending otherwise.
### Errors

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>057</td>
<td>Ignoring function, specified workstation is of category MI</td>
</tr>
</tbody>
</table>

### See Also

- SET WORKSTATION VIEWPORT 3 (3P)
- SET WORKSTATION WINDOW 3 (3P)
- INQUIRE WORKSTATION TRANSFORMATION (3P)
NAME

SET ANNOTATION STYLE – create structure element to set current *annotation style* attribute

SYNOPSIS

C Syntax

```c
void pset_anno_style ( style )
Pint style;  *annotation style*
```

FORTRAN Syntax

```fortran
SUBROUTINE psans ( ASTYLE )
INTEGER ASTYLE  *annotation style*
```

Required PHIGS Operating States

( PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET ANNOTATION STYLE creates a structure element containing a value for the current annotation style attribute, which defines the height of a capital letter, in Text Local Coordinates. This attribute applies to the output primitives

```
ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3
```

If the current edit mode is INSERT, a SET ANNOTATION STYLE element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET ANNOTATION STYLE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

*style*  An integer value specifying one of the following annotation styles:

1  *PANNO_STYLE_UNCONNECTED*

2  *PANNO_STYLE_LEAD_LINE*

FORTRAN Input Parameter

*ASTYLE*  The *annotation style* specifies one of the following:

1  *PUNCON*

2  *PLDLN*

Execution

When the SET ANNOTATION STYLE element is traversed, the current annotation style entry in the PHIGS traversal state list is set to *annotation style*. Characters in text output primitives which follow in the structure network are drawn with the current annotation style.

The *Lead line* value means that the annotation text is connected to its reference point by a lead line drawn using the current polyline attributes.

modified 2 April 1993 787
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>005</th>
<th>Ignoring function, function requires state (PHOP, *, STOP, *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEE ALSO</td>
<td></td>
<td>ANNOTATION TEXT RELATIVE 3 (3P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENERALIZED DRAWING PRIMITIVE -18 (3P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE ANNOTATION FACILITIES (3P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHIGS WORKSTATION DESCRIPTION TABLE (7P)</td>
</tr>
</tbody>
</table>
NAME

SET ANNOTATION TEXT ALIGNMENT – create structure element to set current text alignment attribute

SYNOPSIS

C Syntax

```c
void
pset_anno_align ( text_align )
Ptext_align  *text_align;  text alignment
```

FORTRAN Syntax

```fortran
SUBROUTINE psatal ( ATALH, ATALV )
INTEGER ATALH  annotation text alignment horizontal
INTEGER ATALV  annotation text alignment vertical
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET ANNOTATION TEXT ALIGNMENT creates a structure element containing a value for the current annotation text alignment attribute, which positions the annotation text string in relation to the annotation text position. This attribute applies to the output primitives:

ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3

If the current edit mode is INSERT, a SET ANNOTATION TEXT ALIGNMENT element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET ANNOTATION TEXT ALIGNMENT element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

text_align

A pointer to the text alignment structure, defined as follows:

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

`text_align->hor` is the horizontal component. This is an enumerated value, and may be one of:

- PHOR_NORM Normal
- PHOR_LEFT Left
- PHOR_CTR Center
- PHOR_RIGHT Right

`text_align->vert` is the vertical component. This is an enumerated value, and may be one of:

- PVERT_NORM Normal
**FORTRAN Input Parameters**

**ATALH** The text alignment horizontal component. This is an enumerated value, and may be one of:

- **PAHNOR** Normal
- **PALEFT** Left
- **PACENT** Center
- **PARITE** Right

**ATALV**

The ext alignment vertical component. This is an enumerated value, and may be one of:

- **PAVNOR** Normal
- **PATOP** Top
- **PACAP** Cap
- **PAHALF** Half
- **PABASE** Base
- **PABOTT** Bottom

**Execution**

When the SET ANNOTATION TEXT ALIGNMENT element is traversed, the current annotation text alignment entry in the PHIGS traversal state list is set to annotation text alignment. This attribute is used to position text strings from annotation text output primitives that follow in the structure network, in relation to the text position provided with each text output primitive. The horizontal component has four values; the vertical component, six. The two components of the alignment can be considered individually.

Imagine first rendering the text string using all other text attributes, and then moving the entire text string to place the text extent parallelogram that surrounds the character bodies into the correct position in relation to the text position. (The size and shape of the text is entirely specified by the other attributes.) This movement is oriented by the annotation text character up vector and the annotation text character base vector. (Consider the direction of the character up vector to be vertical and that of the character base vector to be horizontal.

The horizontal alignment of LEFT or RIGHT requires the corresponding side of the parallelogram to pass through the text position. The horizontal alignment of CENTER causes the text position to lie midway between the left and right sides of the parallelogram.

The vertical alignment corresponds to one of the five horizontal lines through the definition of a character. (These lines are in the same location for every character in a single font.) The vertical alignment of TOP or BOTTOM requires the corresponding side of the parallelogram to pass through the text position. The vertical alignment of CAP causes
the text position to lie on the capline of the string (when the annotation text path is LEFT or RIGHT) or on the capline of the topmost character in the string (when the annotation text path is UP or DOWN). The vertical alignment of BASE causes the text position to lie on the baseline of the entire string (when the annotation text path is LEFT or RIGHT) or on the baseline of the bottom character in the string (when annotation text path is UP or DOWN). The vertical alignment of HALF causes the text position to lie on the halfline of the entire string (when the annotation text path is LEFT or RIGHT) or on a line midway between the halflines of the top and bottom characters (when annotation text path is UP or DOWN).

The NORMAL value of either annotation text alignment component causes an effect equivalent to one of the other values of the same component. PHIGS defines which other value is used as the natural alignment for the annotation text path value used:

<table>
<thead>
<tr>
<th>Annotation Text Path Value</th>
<th>Equivalent (Horizontal, Vertical) Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>Left, Base</td>
</tr>
<tr>
<td>LEFT</td>
<td>Right, Base</td>
</tr>
<tr>
<td>UP</td>
<td>Center, Base</td>
</tr>
<tr>
<td>DOWN</td>
<td>Center, Top</td>
</tr>
</tbody>
</table>

The default annotation text alignment is (NORMAL, NORMAL); the default annotation text path is RIGHT.

Example

(CENTER, TOP) annotation text alignment might be used with annotation text path RIGHT to center a chart’s x-axis label under the x-axis, without calculating the combined size of the characters in the string. (RIGHT, CENTER) might be used with annotation text path DOWN to center the right edge of a chart’s y-axis label along the y-axis. Each character faces normally, but the characters in the string would proceed down the display, to the left of the y-axis.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ANNOTATION TEXT RELATIVE 3 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
SET TEXT ALIGNMENT (3P)
NAME
SET ANNOTATION TEXT CHARACTER HEIGHT – create structure element to set current annotation text character height attribute

SYNOPSIS
C Syntax
void
pset_anno_char_ht ( height )
float height; character height

FORTRAN Syntax
SUBROUTINE psatch ( ATCHH )
REAL ATCHH annotation text character height (TLC)

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET ANNOTATION TEXT CHARACTER HEIGHT creates a structure element containing a value for the current annotation text character height attribute, which defines the height of a capital letter, in Text Local Coordinates. This attribute applies to the output primitives:

ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3

If the current edit mode is INSERT, then a SET ANNOTATION TEXT CHARACTER HEIGHT element is inserted in the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET ANNOTATION TEXT CHARACTER HEIGHT element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
height A real value specifying the annotation text character height.

FORTRAN Input Parameter
ATCHH A real value specifying the annotation text character height.

Execution
When the SET ANNOTATION TEXT CHARACTER HEIGHT element is traversed, the current annotation text character height entry in the PHIGS traversal state list is set to SET ANNOTATION TEXT CHARACTER HEIGHT. Characters in text output primitives that follow in the structure network are drawn with the current annotation text character height.

The annotation text character height is specified in Text Local Coordinates (TLC) and determines the height of a capital letter, measured from the character base line parallel to the character up vector. The character base line and character up vector are relative to the TLC system created by the reference point and annotation offset specified in the annotation text relative output primitive.

The default annotation text character height is 0.01.
Errors
005 Ignoring function, function requires state (PHOP, *, STOP, *)

See Also
ANNOTATION TEXT RELATIVE 3 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
SET CHARACTER HEIGHT (3P)
NAME

SET ANNOTATION TEXT CHARACTER UP VECTOR – create structure element to set current annotation text character up vector attribute

SYNOPSIS

C Syntax

```c
void pset_anno_char_up_vec ( up_vect )
```

```c
Pvec *up_vect; character up vector
```

FORTRAN Syntax

```fortran
SUBROUTINE psetcu ( ATCHUX, ATCHUY )
```

```fortran
REAL ATCHUX, ATCHUY text character up vector (TLC)
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET ANNOTATION TEXT CHARACTER UP VECTOR creates a structure element containing a value for the current annotation text character up vector attribute that determines the orientation of individual characters. This attribute applies to the output primitives:

- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3

If the current edit mode is INSERT, then a SET ANNOTATION TEXT CHARACTER UP VECTOR element is inserted in the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET ANNOTATION TEXT CHARACTER UP VECTOR element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

**up_vect** A pointer to the annotation text character up vector containing TLC values, defined as follows:

```c
typedef struct {
    Pfloat delta_x; x coordinate
    Pfloat delta_y; y coordinate
} Pvec;
```

FORTRAN Input Parameters

**ATCHUX** The annotation text character up vector x component.

**ATCHUY** The annotation text character up vector y component.

Execution

When the SET ANNOTATION TEXT CHARACTER UP VECTOR element is traversed, the current annotation text character up vector entry in the PHIGS traversal state list is set to annotation text character up vector. Characters in text output primitives that follow in the structure network are oriented by the annotation text character up vector. The *annotation text character base vector* is defined as a vector of arbitrary length set at a right angle to the
character up vector in the clockwise direction. The default annotation text character up vector is (0,1); the default annotation text character base vector is (1,0). If the annotation text character up vector is determined to be degenerate (that is, has length 0), then the defaults are used.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ANNOTATION TEXT RELATIVE 3 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
SET CHARACTER UP VECTOR (3P)
NAME

SET ANNOTATION TEXT PATH – create structure element to set current text path attribute

SYNOPSIS

C Syntax

```c
void pset_anno_path ( path )
Text_path path; text path
```

FORTRAN Syntax

```fortran
SUBROUTINE psatp ( ATP )
INTEGER ATP @annotation text path
(PRIGHT, PLEFT, PUP, PDOWN)
```

Required PHIGS Operating States

(HPH, *, STOP, *)

DESCRIPTION

Purpose

SET ANNOTATION TEXT PATH creates a structure element containing a value for the text path attribute that controls the direction in which the string is written relative to the character up and base vectors. This attribute applies to the following output primitives:

- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3

which follow in the structure network.

If the current edit mode is INSERT, then a SET ANNOTATION TEXT PATH element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET ANNOTATION TEXT PATH element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

`path` The annotation text path is one of the following enumerated values:

- PPATH_RIGHT Right
- PPATH_LEFT Left
- PPATH_UP Up
- PPATH_DOWN Down

FORTRAN Input Parameter

`ATP` The annotation text path is one of the following enumerated values:

- PRIGHT Right
- PLEFT Left
- PUP Up
- PDOWN Down

Execution

When the SET ANNOTATION TEXT PATH element is traversed, the current annotation text path entry in the PHIGS traversal state list is set to annotation text path. The annotation text path determines the direction of displacement between one character and the next in

796 modified 2 April 1993
a string, defined in relation to the annotation text character base and up vectors.
If annotation text path is set to Right, then the text string is written along a baseline in the
direction of the ANNOTATION TEXT CHARACTER BASE VECTOR. If it is set to Left, then the text
string is written along a baseline in the direction opposite to the ANNOTATION TEXT
CHARACTER BASE VECTOR. If it is set to Up, then the text string is written in the direction of
the ANNOTATION TEXT CHARACTER UP VECTOR. If it is set to Down, then the text string is
written in the direction opposite to the ANNOTATION TEXT CHARACTER UP VECTOR.
The ANNOTATION TEXT CHARACTER BASE VECTOR attribute is an implicit attribute derived
from the ANNOTATION TEXT CHARACTER UP VECTOR.

Note: annotation text path controls only the direction in which the string is written. The
position of the string in relation to the text position point is controlled by the annotation
text alignment.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

ANNOTATION TEXT RELATIVE 3 (3P)
GENERALIZED DRAWING PRIMITIVE -18 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
SET TEXT PATH (3P)
NAME

SET CHARACTER EXPANSION FACTOR – create structure element to set current character expansion factor attribute

C Syntax

void
pset_char_expan ( exp_factor )
float exp_factor; character expansion factor

FORTRAN Syntax

SUBROUTINE pschxp ( CHXP )
REAL CHXP character expansion factor

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET CHARACTER EXPANSION FACTOR creates a structure element containing a value for the current character expansion attribute, which changes the characters’ width-to-height ratio from the ratio with which the font was designed.

When the character expansion factor Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute applies to the following output primitives:

TEXT
TEXT 3
ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3

C Input Parameter

exp_factor
A real value specifying the character expansion factor.

FORTRAN Input Parameter

CHXP A real value specifying the character expansion factor.

Execution

If the current edit mode is INSERT, a SET CHARACTER EXPANSION FACTOR element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET CHARACTER EXPANSION FACTOR element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element. When the SET CHARACTER EXPANSION FACTOR element is traversed, the current character expansion factor entry in the PHIGS traversal state list is set to exp_factor.

When the current character expansion factor ASF is set to INDIVIDUAL, characters in text output primitives that follow in the structure network are modified by the absolute value of the expansion factor specified.

The default character expansion factor is 1.0, which results in no change in the width-to-height ratio of characters from their definition by the font designer. A character expansion factor of less than 1.0 produces narrower characters, and a character expansion...
factor greater than 1.0 produces wider characters.
When the character expansion factor ASF is set to BUNDLED, the character expansion factor is taken from the workstation’s representation indicated by the current text index. In this case, the expansion factor set with \texttt{SET CHARACTER EXPANSION FACTOR} has no effect.

\begin{enumerate}
  \item \textbf{ERRORS} 005 Ignoring function, function requires state (PHOP, \textasciitilde, STOP, \textasciitilde)
  \item \textbf{SEE ALSO}
    \begin{itemize}
      \item GENERALIZED DRAWING PRIMITIVE -17 (3P)
      \item GENERALIZED DRAWING PRIMITIVE -18 (3P)
      \item GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
      \item GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
      \item SET CHARACTER HEIGHT (3P)
      \item SET CHARACTER SPACING (3P)
      \item SET INDIVIDUAL ASF (3P)
      \item SET TEXT REPRESENTATION (3P)
    \end{itemize}
\end{enumerate}
NAME
SET CHARACTER HEIGHT – create structure element to set current character height attribute

SYNOPSIS
C Syntax
void
pset_char_ht ( height )
float height; character height

FORTRAN Syntax
SUBROUTINE pschh ( CHH )
REAL CHH character height (TLC)

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET CHARACTER HEIGHT creates a structure element containing a value for the current character height attribute, which defines the height of a capital letter, in Text Local Coordinates. This attribute applies to the output primitives:

TEXT
TEXT 3

If the current edit mode is INSERT, then a SET CHARACTER HEIGHT element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET CHARACTER HEIGHT element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
height A real value specifying the character height.

FORTRAN Input Parameter
CHH A real value specifying the character height.

Execution
When the SET CHARACTER HEIGHT element is traversed, the current character height entry in the PHIGS traversal state list is set to character height. Characters in text output primitives that follow in the structure network are drawn with the current character height.

The character height is specified in Text Local Coordinates (TLC) and determines the height of a capital letter, measured from the character base line parallel to the character up vector. The character base line and character up vector are relative to the TLC system created by the text position and text vectors specified in the text output primitive.

The default character height is 0.01.

800 modified 2 April 1993
ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
SET ANNOTATION TEXT CHARACTER HEIGHT (3P)
SET CHARACTER EXPANSION FACTOR (3P)
SET CHARACTER SPACING (3P)
SET TEXT REPRESENTATION (3P)
NAME

SET CHARACTER SPACING – create structure element to set current character spacing attribute

SYNOPSIS

C Syntax

void
pset_char_space ( spacing )

float spacing; character spacing

FORTRAN Syntax

SUBROUTINE pset_char_space ( CHSP )

REAL CHSP character spacing

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET CHARACTER SPACING creates a structure element containing a value for the current character spacing attribute, which controls how much space is inserted between characters in a text string.

When the character spacing Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute applies to the following output primitives:

- TEXT
- TEXT 3
- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3

C Input Parameter

spacing The character spacing, specified as a real fraction of the font’s nominal character height.

FORTRAN Input Parameter

CHSP The character spacing, specified as a real fraction of the font’s nominal character height.

Execution

If the current edit mode is INSERT, then a SET CHARACTER SPACING element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET CHARACTER SPACING element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET CHARACTER SPACING element is traversed, the current character spacing entry in the PHIGS traversal state list is set to character spacing. When the current character spacing ASF is set to INDIVIDUAL, the character spacing is applied to text strings in output primitives that follow in the structure network.

The character spacing attribute specifies the amount of space to be inserted between the character bodies in the text primitives. A positive value inserts additional space between characters, and a negative value causes the character bodies to overlap. The default value
is 0.0: the character bodies are placed adjacent to one another, with no additional space beyond that in the font design. (A character’s body is the rectangle that encloses the limits of the character.)

When the character spacing ASF is set to BUNDLED, the character spacing will be taken from the workstation’s representation indicated by the current text index. In this case, the spacing set with SET CHARACTER SPACING has no effect.

**ERRORS**

| 005 | Ignoring function, function requires state (PHOP, *, STOP, *) |

**SEE ALSO**

- GENERALIZED DRAWING PRIMITIVE -17 (3P)
- GENERALIZED DRAWING PRIMITIVE -18 (3P)
- GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
- GENERALIZED DRAWING PRIMITIVE 3 -18 (3P)
- SET CHARACTER EXPANSION FACTOR (3P)
- SET CHARACTER HEIGHT (3P)
- SET INDIVIDUAL ASF (3P)
- SET TEXT REPRESENTATION (3P)
### NAME
SET CHARACTER UP VECTOR – create structure element to set current *character up vector* attribute

### SYNOPSIS
**C Syntax**
```c
void pset_char_up_vec ( up_vect )
    Pvec *up_vect;  // character up vector
```

**FORTRAN Syntax**
```fortran
SUBROUTINE pschup ( CHUX, CHUY )
    REAL    CHUX, CHUY    // character up vector (TLC)
```

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION
**Purpose**
SET CHARACTER UP VECTOR creates a structure element containing a value for the current character up vector attribute, which determines the orientation of individual characters. This attribute applies to the following output primitives:

- TEXT
- TEXT 3

If the current edit mode is INSERT, a SET CHARACTER UP VECTOR element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET CHARACTER UP VECTOR element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

**C Input Parameter**
- `up_vect` A pointer to the character up vector containing TLC values, defined as follows:
  ```c
typedef struct {
    Pfloat delta_x;  // x coordinate
    Pfloat delta_y;  // y coordinate
  } Pvec;
```

**FORTRAN Parameters**
- `CHUX` The character up vector x component.
- `CHUY` The character up vector y component.

**Execution**
When the SET CHARACTER UP VECTOR element is traversed, the current character up vector entry in the PHIGS traversal state list is set to character up vector. Characters in text output primitives that follow in the structure network will be oriented by the character up vector. The character base vector is defined as a vector of arbitrary length set at a right angle to the character up vector in the clockwise direction.

The default character up vector is (0,1); the default character base vector is (1,0). If the character up vector is determined to be degenerate (that is, has length 0), then the defaults are used.

---

804 modified 2 April 1993
Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

GENERALIZED DRAWING PRIMITIVE -17 (3P)
GENERALIZED DRAWING PRIMITIVE 3 -17 (3P)
SET ANNOTATION TEXT CHARACTER UP VECTOR (3P)
NAME
SET CHOICE MODE – set choice device operating mode and echoing state

SYNOPSIS
C Syntax

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

FORTRAN Syntax

```fortran
SUBROUTINE pschm ( WKID, CHDNR, MODE, ESW )
INTEGER WKID       /* workstation identifier */
INTEGER CHDNR      /* choice device number */
INTEGER MODE       /* operating mode (PREQU, PSAMPL, PEVENT) */
INTEGER ESW        /* echo switch (PNECHO, PECHO) */
```

Required PHIGS Operating States

( PHOP, WSOP, *, * )

DESCRIPTION
Purpose

Use the SET CHOICE MODE subroutine to set the operating mode (REQUEST, SAMPLE, or EVENT) and the echo switch (ECHO, NOECHO) for a specified choice device on a specified workstation.

C Input Parameters

ws The workstation identifier of the workstation associated with the device.

dev The device number of the choice device to be set. See the AVAILABLE DEVICES section in INITIALIZE CHOICE 3 for a description of the available devices.

mode Specifies the operating mode for the specified choice device. Pop_mode is an enumeration defined in phigs.h as follows:

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

echo The echo switch value for the specified choice device. Pecho_switch is an enumeration defined in phigs.h as follows:

```c
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
```
FORTRAN Input

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>CHDNR</td>
<td>The device number of the choice device to be set. See the AVAILABLE DEVICES section in INITIALIZE CHOICE 3 for a description of the available devices.</td>
</tr>
<tr>
<td>MODE</td>
<td>The desired mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PREQU Request</td>
</tr>
<tr>
<td></td>
<td>PSAMPL Sample</td>
</tr>
<tr>
<td></td>
<td>PEVENT Event</td>
</tr>
<tr>
<td>ESW</td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PNECHO No echo</td>
</tr>
<tr>
<td></td>
<td>PECHO Echo</td>
</tr>
</tbody>
</table>

Execution

The SET CHOICE MODE sets the operating mode of the specified choice device to REQUEST, SAMPLE, or EVENT, and the echo switch to ECHO or NOECHO. The default operating mode is REQUEST. The default echo switch is ECHO.

The operating mode controls how the input from the device is obtained.

- If the operating mode is REQUEST, the subroutine REQUEST CHOICE or REQUEST CHOICE 3 can be used to add the specified device number to the device trigger’s list of recipients, and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the REQUEST CHOICE subroutine returns the current input value of the device’s measure process, and the status OK. If a break occurs, the status NONE is returned.

- If the operating mode is SAMPLE, the SAMPLE CHOICE or SAMPLE CHOICE 3 subroutine can be used to return the current input value of the device without waiting for the trigger to fire.

- If the operating mode is EVENT, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines AWAIT EVENT and GET CHOICE can then be used to read event reports from the queue.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process.

ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or of category OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
</tbody>
</table>

modified 2 April 1993
SEE ALSO

INITIALIZE CHOICE 3 (3P)
REQUEST CHOICE (3P)
GET CHOICE (3P)
SAMPLE CHOICE (3P)
INQUIRE CHOICE DEVICE STATE (3P)
### NAME
SET COLOUR MODEL – select colour model for workstation colour table

### SYNOPSIS
**C Syntax**

void

cset_colr_model ( ws, model )

Pint ws;  /* workstation identifier */
Pint model;  /* colour model */

**FORTRAN Syntax**

SUBROUTINE pscmd ( WKID, CMODEL )

INTEGER WKID  /* workstation identifier */
INTEGER CMODEL  /* colour model */

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

**Purpose**

SET COLOUR MODEL selects the colour model used to represent colours on the workstation table of defined colour representations, which is used to specify a primitive’s colour attribute.

**Note:** SunPHIGS presently supports only the RGB colour model.

### C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| ws        | Set the colour model for the workstation with identifier *ws*.
| model     | Use this colour model to interpret the colour values in the workstation colour table. Values defined in *phigs.h* are: |
|           | 1 PMODEL_RGB  *Red, Green, Blue*  |
|           | 2 PMODEL_CIELUV  *CIE universal colour definition system*  |
|           | 3 PMODEL_HSV  *Hue, saturation, value*  |
|           | 4 PMODEL_HLS  *Hue, lightness, saturation*  |

### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| WKID      | Set the colour model for the workstation with identifier *WKID*.
| CMODEL    | Use this colour model to interpret the colour values in the workstation colour table. Valid values are defined in *phigs77.h*: |
|           | PRGB  *Red, Green, Blue*  |
|           | PCIE  *CIE universal colour definition system*  |
|           | PHSV  *Hue, saturation, value*  |
|           | PHLS  *Hue, lightness, saturation*  |

### Execution

When SET COLOUR MODEL is called, the current colour model entry in the workstation state list is set to colour model, which selects a method of interpreting the colour bundles in the workstation table of defined colour representations. Each entry on the colour table contains three colour values. When the current colour model is set to *Red, Green, Blue*,...
these values are interpreted as the red, green and blue components, respectively, of the colour in RGB colour space.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>110</td>
<td>Ignoring function, the specified colour model is not available on the workstation.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

INQUIRE COLOUR MODEL FACILITIES (3P)
INQUIRE COLOUR MODEL (3P)
**NAME**

SET COLOUR REPRESENTATION – define colour representation entry in workstation’s colour table

**SYNOPSIS**

**C Syntax**

```c
void pset_colr_rep ( ws, index, rep )
```

- `Pint ws;` workstation identifier
- `Pint index;` colour bundle index
- `Pcolr_rep *rep;` colour representation pointer

**FORTRAN Syntax**

```fortran
SUBROUTINE pscr ( WKID, CI, NCCS, CSPEC )
```

- `INTEGER WKID` workstation identifier
- `INTEGER CI` colour index
- `INTEGER NCCS` number of components of colour specification
- `REAL CSPEC` colour specification

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

SET COLOUR REPRESENTATION defines an entry in the workstation’s colour table. Each entry in this table contains three component values, defining a colour in the current colour model. The colour representations apply to all primitives.

**C Input Parameters**

- `ws` The identifier of the workstation for which the colour representation is being defined.
- `index` The colour index of the entry being defined.
- `rep` A pointer to a union containing the three colour components defining the colour representation. `Pcolr_rep` is defined in phigs.h as follows:

```c
typedef union {
  Prgb rgb; /* Red Green Blue colour specification */
  Pcieluv cieluv; /* CIE L*U*V* colour specification */
  Phls hls; /* Hue Lightness Saturation colour specification */
  Phsv hsv; /* Hue Saturation Value colour specification */
  Pdata unsupp; /* Colour in unsupported colour model */
} Pcolr_rep;
```

`Prgb` is defined in phigs.h as follows:

```c
typedef struct {
  /* Red Green Blue colour specification */
  /* CIE L*U*V* colour specification */
  /* Hue Lightness Saturation colour specification */
  /* Hue Saturation Value colour specification */
  /* Colour in unsupported colour model */
}```

**modified 2 April 1993**
Pfloat red; /* red, hue, and so on */
Pfloat green; /* green, saturation, lightness, and so on */
Pfloat blue; /* blue, value, saturation, and so on */

} Prgb;

Pcieluv is defined in phigs.h as follows:
typedef struct {
    Pfloat cieluv_x; /* x coefficient */
Pfloat cieluv_y; /* y coefficient */
Pfloat cieluv_y_lum; /* y luminance */
} Pcieluv;

Phls is defined in phigs.h as follows:
typedef struct {
    Pfloat hue; /* hue */
Pfloat lightness; /* lightness */
Pfloat satur; /* saturation */
} Phls;

Phsv is defined in phigs.h as follows:
typedef struct {
    Pfloat hue; /* hue */
Pfloat satur; /* saturation */
Pfloat value; /* value */
} Phsv;

Pdata is defined in phigs.h as follows:
typedef struct {
    size_t size; /* size of data */
    char *data /* pointer to data */
} Pdata;

FORTRAN Input Parameters

WKID The identifier of the workstation for which the colour representation is being defined.

CI The colour index of the entry being defined.

NCCS The number of components of the colour specification.

CSSPEC(*) The colour specification.

Execution When SET COLOUR REPRESENTATION is called, the colour index entry in the table of defined colour representations on the workstation is set to the colour components specified.

812 modified 2 April 1993
For the RGB colour model, the only one implemented in this release, the range for these
colour components is 0 to 1: 0 sets the colour component off; 1 sets the colour component
at full intensity.
The colour table is numbered from 0, the background colour, to a workstation-dependent
maximum.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that
is, the workstation category is not OUTPUT, OUTIN, or MO)
113 Ignoring function, the colour index value is less than zero
103 Ignoring function, setting this bundle table entry would exceed the maximum
number of entries allowed in the workstation bundle table
118 Ignoring function, one of the components of the colour specification is out of
range. The valid range is dependent upon the current colour model

**SEE ALSO**

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
WORKSTATION TYPE SET (3P)
SET POLYLINE COLOUR INDEX (3P)
SET POLYMARKER COLOUR INDEX (3P)
SET TEXT COLOUR INDEX (3P)
SET INTERIOR COLOUR INDEX (3P)
SET EDGE COLOUR INDEX (3P)
SET COLOUR MODEL (3P)
INQUIRE COLOUR REPRESENTATION (3P)
INQUIRE PREDEFINED COLOUR REPRESENTATION (3P)
NAME
SET CONFLICT RESOLUTION – set conflict resolution flags

SYNOPSIS
C Syntax
void
pset_conf_res ( archival_resolution, retrieval_resolution )
Pconf_res archival_resolution;@archival
conflict resolution
Pconf_res retrieval_resolution;@retrieval
conflict resolution

FORTRAN Syntax
SUBROUTINE pscnrs ( ARCCR, RETCR )
INTEGER ARCCR archival conflict resolution
INTEGER RETCR retrieval conflict resolution

Required PHIGS
Operating States
(PHOP, *, *, *)

DESCRIPTION
Purpose
The SET CONFLICT RESOLUTION subroutine sets the conflict resolution flags controlling how
structure archival and retrieval conflicts are resolved. The flag value can be MAINTAIN,
ABANDON or UPDATE.

The default value of the archival conflict resolution flag is UPDATE. The default value of
the retrieval conflict resolution flag is ABANDON.

C Input Parameters

archival_resolution
An enumerated variable specifying the value of the archival conflict resolution
flag.

retrieval_resolution
An enumerated variable specifying the value of the retrieval conflict resolution
flag.

Valid values (defined in phigs.h) for these two flags are:

PRES_MAINTAIN
PRES_ABANDON
PRES_UPD

FORTRAN Input
Parameters

ARCCR An enumerated variable specifying the value of the archival conflict resolution
flag.

RETCR An enumerated variable specifying the value of the retrieval conflict resolution
flag.

Valid values (defined in phigs77.h) for these two flags are:

PCRMNT Maintain
PCRABA Abandon

modified 2 April 1993
The SET CONFLICT RESOLUTION subroutine sets the conflict resolution flags in the PHIGS state list. The archival conflict resolution flag controls what happens when an attempt is made to archive structures whose identifiers already exist in the archive file. The retrieval conflict resolution flag controls what happens when an attempt is made to retrieve structures whose identifiers already exist in the Central Structure Store.

The flag values can be Maintain, Abandon, or Update. A value of Maintain prevents conflicting structures from being overwritten; a value of Update allows conflicting structures to be overwritten; and a value of Abandon indicates that the entire archival or retrieval operation should be abandoned, with no changes made to the archive or the Central Structure Store if there are any conflicts.

ERRORS

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

SEE ALSO

ARCHIVE STRUCTURES (3P)
INQUIRE ALL CONFLICTING STRUCTURES (3P)
INQUIRE CONFLICTING STRUCTURES IN NETWORK (3P)
INQUIRE CONFLICT RESOLUTION (3P)
RETRIEVE STRUCTURES (3P)
NAME

SET DISPLAY UPDATE STATE – set workstation deferral and modification modes

SYNOPSIS

C Syntax

void

psetDisp_upd_st ( ws, def_mode, mod_mode )

Pint ws;      // workstation identifier
Pdef_mode def_mode;  // deferral mode
Pmod_mode mod_mode;  // modification mode

FORTRAN Syntax

SUBROUTINE psdus ( WKID, DEFMOD, MODMOD )

INTEGER WKID    // workstation identifier
INTEGER DEFMOD  // deferral mode
INTEGER MODMOD  // modification mode

DESCRIPTION

Purpose

SET DISPLAY UPDATE STATE sets the deferral and modification modes in the specified workstation’s state list. These modes provide control over when and how changes to the structure store and workstation tables actually appear in the display. These values have a great effect on both visual results and performance.

The deferral mode controls when the display is updated (that is, made entirely correct). This allows you to delay time-consuming or visually-distracting regeneration until a set of related changes are completed or until a correct display is absolutely necessary, as before an operator input interaction can proceed.

The modification mode controls the visual effects that take place while the workstation is postponing display updates. This allows you to selectively change the display, but to avoid regenerating the entire picture.

C Input Parameters

ws      // Identifier of the workstation whose update state to set.

def_mode

An enumerated value specifying the deferral mode to be set for this workstation. Valid values are defined in phigs.h:

PDEFER_ASP  // Make the display visually correct As Soon As Possible
PDEFER_BNIG  // Make the display visually correct Before the Next Interaction
        // Globally
PDEFER_BNIL  // Make the display visually correct Before the Next Interaction
        // Locally
PDEFER_ASTI  // Make the display visually correct At Some Time
PDEFER_WAIT  // Make the display visually correct When the Application
        // Requests It

mod_mode

816

modified 2 April 1993
An enumerated value specifying the *modification mode* to be set for this workstation. Valid values are defined in phigs.h:

- PMODE_NIVE: No immediate visual effects
- PMODE_UWOR: Update without regeneration
- PMODE_UQUM: Use quick update methods

**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>WKID</th>
<th>Identifier of the workstation whose update state to set.</th>
</tr>
</thead>
</table>

**DEFMOD**

An enumerated value specifying the *deferral mode* to be set for this workstation. Valid values are defined in phigs77.h:

- PASAP: Make the display visually correct As Soon As Possible
- PBNIG: Make the display visually correct Before the Next Interaction Globally
- PBNIL: Make the display visually correct Before the Next Interaction Locally
- PASTI: Make the display visually correct At Some Time
- PWAITD: Make the display visually correct When the Application Requests It

**MODMOD**

An enumerated value specifying the *modification mode* to be set for this workstation. Valid values are defined in phigs77.h:

- PNIVE: No immediate visual effects
- PUWOR: Update without regeneration
- PUQUM: Use quick update methods

**Execution**

The SET DISPLAY UPDATE STATE sets the *deferral mode* and *modification mode* entries in the specified workstation’s state list. These modes control the degree to which the display must reflect the state of the central structure store and the workstation tables.

**Deferral Modes**

The deferral mode (*As Soon As Possible (ASAP)*) demands that the display be consistent with the structure store and the workstation state list at all times. This typically causes a regeneration (that is, clearing of the display and traversal of all posted structure networks) for every change to a structure appearing on the workstation, and for every change to the workstation’s state list. The modification mode is unused when the deferral mode is ASAP.

The deferral mode (*Before the Next Interaction Globally*) behaves just like ASAP when any input device is active on *any* workstation. The deferral mode (*Before the Next Interaction Locally*) behaves just like ASAP when any input device is active on *this* workstation. Otherwise, they behave exactly like *At Some Time*, as described below. An input device is considered *active* the entire time it is in EVENT or SAMPLE mode, or while a REQUEST is pending for the device (although nothing but window redisplay can occur while a REQUEST is pending, because it necessarily blocks).

The deferral mode (*At Some Time*) updates the display only at certain times, which are at the discretion of the PHIGS implementation and are workstation-dependent. For SunPHIGS, these times are when the window system gives the workstation a REDRAW request in
response to window damage, and when the open structure is closed.

The deferral mode (*When the Application Requests It*) is abbreviated *WAIT*; the display will not be updated implicitly. Explicit updates are requested by using *REDRAW ALL STRUCTURES* or *UPDATE WORKSTATION* with *PERFORM*.

**Modification Modes**

During modification mode (*No Immediate Visual Effects*), the only changes to the display are those that would be done in accordance with the deferral mode.

**Note:**

*When the Application Requests It* and *No Immediate Visual Effects* used together prohibit any change to the display in response to changes in the central structure store or the workstation state lists.

The modification mode (*Update Without Regeneration*) performs all updates that can be realized immediately without regenerating the entire display. This includes the actions whose *dynamic modification accepted* entries in the workstation description table are set to *Immediately*.

The modification mode (*Use Quick Update Methods*) allows use of workstation dependent simulations of changes that cannot be performed immediately unless the display is regenerated. These simulations are described in the section *Available Quick Update Methods*. Actions for which a *quick update method* is available have *dynamic modification accepted* entries in the workstation description table set to *Can Be Simulated*. As in *Update Without Regeneration* mode, actions whose *dynamic modification accepted* entries in the workstation description table are set to *Immediately* are performed immediately. Unless an action can be simulated or the update performed immediately, the update is deferred, as if the modification mode were *No Immediate Visual Effects*.

**State of Visual Representation**

The workstation’s *state of visual representation* indicates whether the display is *Correct*, *Simulated* (but no updates have been deferred), or *Deferred*. This state list value may be obtained using *INQUIRE DISPLAY UPDATE STATE(3P)*.

Changing the *deferral mode* causes a regeneration, if the display is not correct and the new *deferral mode* requires it. Changing the *modification mode* has no retroactive effect on previous changes that have been simulated or deferred.

**Available Quick Update Methods**

SunPHIGS provides the same simulations for X Tool and X Drawable workstations when *modification mode* is *Use Quick Update Methods* – all variations on a theme. A portion of the image is undrawn, by drawing in the background color (index 0), then the revised portion is drawn, using the colors specified by PHIGS attributes. If the portion being drawn includes *EXECUTE STRUCTURE* elements, then the execution is carried out. Note that the simulations disregard the structure network’s posting priority; therefore, undrawing can leave holes in other primitives that overlap it, and drawing can place on top a structure that should be under another. These problems are the reason the term *simulation* is used; SunPHIGS approximates the correct picture, but sometimes this approximation is
inadequate.

No simulations are supported for CGM Output workstations. Simulations for X Tool and X Drawable workstations are described below.

Structure Content Modification

If a range of primitives and EXECUTE STRUCTURE elements is deleted (or if EMPTY STRUCTURE is called), then the contents are undrawn. If an output primitive or EXECUTE STRUCTURE element is deleted, then it is undrawn; if such an element is inserted, then it is drawn. If such an element is replaced using Replace edit mode (see SET EDIT MODE(3P)), then the delete simulation is followed by the insert simulation. Replacing a local or global modelling transformation element with another is simulated by undrawing the rest of the structure and then redrawing those elements using the new transformation. If the structure appears more than once on the workstation, the simulation occurs for all appearances.

There are presently no simulations for attribute changes, but the effect of updating a primitive’s attributes can be achieved by deleting a primitive, changing the attributes that precede it, and inserting the primitive again.

Structure Posting

When POST STRUCTURE is called, if the structure is not already posted on the workstation, the simulation is to draw the structure network on top of all posted structures (ignoring priority). If the structure is already posted on the workstation, calling POST STRUCTURE may change the structure’s priority relative to other structures posted to the same workstation. If its relative priority increases, then the structure network is drawn on top of all posted structures. The simulation for UNPOST STRUCTURE is to undraw the structure network.

Because of previously deferred changes to attributes, the undraw of a primitive may be imperfect. UNPOST ALL STRUCTURES is simulated by clearing the workstation’s display surface.

Structure Manipulation

When a structure is deleted by either DELETE STRUCTURE or DELETE STRUCTURE NETWORK, all appearances of the structure are undrawn. DELETE ALL STRUCTURES is simulated by clearing the workstation’s display surface.

For large amounts of editing, it can be more efficient or more visually pleasing to disable screen update by setting the deferral mode to When the Application Requests It (WAIT) and the modification mode to No Immediate Visual Effects. Then any amount of editing may be performed without any visual change. When desired, UPDATE WORKSTATION (or setting deferral mode to ASAP) correctly displays the modified data. Double buffering (enabled by WORKSTATION TYPE SET (3P)) allows the corrected display to be prepared before it is displayed.

Actions that are DEFERRED by SunPHIGS in Use Quick Update Methods mode, such as changing a workstation representation, may degrade the quality of the QUM simulation. An example is changing an interior representation and then deleting a fill area that uses bundled attributes from that representation: the quick update method will undraw the
primitive using the attributes of the current bundle, and not the bundle values with which the primitive was initially drawn.

**Interactions between Deferral and Modification Modes and The Window System**

**X Tool**  SunPHIGS regenerates the image from the CSS when the window is damaged by window system events. This can occur even if the deferral mode is *When the Application Requests It* (WAIT) and the modification mode is *No Immediate Visual Effects*, which used together prohibit implicit changes to the display. However, only the damaged portion of the SunPHIGS canvas is repainted. Hence, the regeneration may use newer information than that which generated the older, out-of-date, but not damaged portion, which is also visible and may not match the newly-drawn portion. To remove any visible mismatch, the entire window can be updated by selecting Refresh from the X Tool’s Frame menu.

**X Drawable**

An application using the X Drawable workstation type must update the workstation explicitly (using UPDATE WORKSTATION or REDRAW ALL STRUCTURES) when it receives a window damage event from the window system. An X Drawable workstation type does not know about window system events unless the application notifies it.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

UPDATE WORKSTATION (3P)
REDRAW ALL STRUCTURES (3P)
INQUIRE DISPLAY UPDATE STATE (3P)
INQUIRE DEFAULT DISPLAY UPDATE STATE (3P)
NAME
SET EDGE COLOUR INDEX – create structure element to set current edge colour index attribute

SYNOPSIS
C Syntax
void
pset_edge_colr_ind ( index )
Pint index; edge colour index

FORTRAN Syntax
SUBROUTINE psedci ( COLI )
INTEGER COLI edge colour index

Required PHIGS Operating States
( PHOP, *, STOP, * )

DESCRIPTION
Purpose
SET EDGE COLOUR INDEX creates a structure element containing a value for the current edge colour index attribute. When the current edge colour index Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute indexes the colour representation that applies to the output primitives:

- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- GENERALIZED DRAWING PRIMITIVE (Fill Area Primitives)
- GENERALIZED DRAWING PRIMITIVE 3 (Fill Area Primitives)
- NON-UNIFORM B-SPLINE SURFACE
- SET OF FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

C Input Parameter
index  An integer colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameter
COLI  An integer colour index, which selects a colour value from the workstation’s colour table.

Execution
If the current edit mode is INSERT, a SET EDGE COLOUR INDEX element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET EDGE COLOUR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET EDGE COLOUR INDEX element is traversed, the current edge colour index entry in the PHIGS traversal state list is set to the colour index.

modified 2 April 1993
When the current edge colour index ASF is set to INDIVIDUAL, the fill area output primitives that follow in the structure network are filled with the colour representation selected by the current edge colour index from the workstation’s colour table. If the colour index specified is not available on the workstation, then colour index 1 is used.

When the current edge colour index ASF is set to BUNDLED, the edge colour index is taken from the workstation’s representation indicated by the current edge index. In this case, the colour index set with SET EDGE COLOUR INDEX has no effect.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)
113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

SET COLOUR REPRESENTATION (3P)
SET EDGE FLAG (3P)
SET INDIVIDUAL ASF (3P)
SET EDGE REPRESENTATION (3P)
NAME

SET EDGE FLAG – create structure element to set current edge flag attribute

SYNOPSIS

C Syntax

```c
void pset_edge_flag ( edge_flag )
edge_flag edge_flag; edge_flag
```

FORTRAN Syntax

```fortran
SUBROUTINE psedfg ( EDFLAG )
INTEGER EDFLAG edge_flag (POFF, PON)
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET EDGE FLAG creates a structure element containing a value for the current edge flag attribute. When the current edge flag Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute defines whether or not edges are drawn on the following output primitives:

- Fill Area Set
- Fill Area Set 3
- Fill Area Set 3 with Data
- Generalized Drawing Primitive (Fill Area Primitives)
- Generalized Drawing Primitive 3 (Fill Area Primitives)
- Non-Uniform B-Spline Surface
- Set of Fill Area Set 3 with Data
- Quadrilateral Mesh 3 with Data
- Triangle Strip 3 with Data

C Input Parameter

`edge_flag`

The edge flag is an enumerated type defined in phigs.h to have the following values:

- PEDGE_OFF Off
- PEDGE_ON On

FORTRAN Input Parameter

`EDFLAG`

The edge flag is an enumerated value, from:

- POFF Off
- PON On

Execution

If the current edit mode is INSERT when SET EDGE FLAG is called, a SET EDGE FLAG element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET EDGE FLAG element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

modified 2 April 1993
When the SET EDGE FLAG element is traversed, the current edge flag entry in the PHIGS traversal state list is set to edge flag. When the edge flag ASF is set to INDIVIDUAL the edges of the fill area set output primitives that follow in the structure network are drawn when edge flag is On, and are not drawn (the default) when edge flag is Off.

When the current edge flag ASF is set to BUNDLED, the edge flag setting is taken from the workstation’s representation indicated by the current edge index. In this case, the edge flag set with SET EDGE FLAG has no effect.

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO SET INDIVIDUAL ASF (3P)
SET EDGE REPRESENTATION (3P)
FILL AREA SET (3P)
### NAME

SET EDGE INDEX – create structure element containing *edge representation index* attribute

### SYNOPSIS

**C Syntax**

```c
void pset_edge_ind ( index )
Pint index;  edge index
```

**FORTRAN Syntax**

```fortran
SUBROUTINE psedi ( EDI )
INTEGER EDI  edge representation index
```

**Required PHIGS Operating States**

`(PHOP, *, STOP, *)`

### DESCRIPTION

**Purpose**

SET EDGE INDEX creates a structure element containing an edge index representation number that selects an entry from the workstation’s edge bundle table. This attribute applies to the output primitives:

- `FILL AREA SET`
- `FILL AREA SET 3`
- `FILL AREA SET 3 WITH DATA`
- `GENERALIZED DRAWING PRIMITIIVE`
- `GENERALIZED DRAWING PRIMITIIVE 3`
- `NON-UNIFORM B-SPLINE SURFACE`
- `SET OF FILL AREA SET 3 WITH DATA`
- `QUADRILATERAL MESH 3 WITH DATA`
- `TRIANGLE STRIP 3 WITH DATA`

If the current edit mode is INSERT, a SET EDGE INDEX element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET EDGE INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

**C Input Parameter**

`index` An edge index for edge representation on the workstation.

**FORTRAN Input Parameter**

`EDI` An *edge index* for a edge representation on the workstation.

**Execution**

When the SET EDGE INDEX element is traversed, the current edge index value is set to the edge index that specifies an entry from the workstation’s edge bundle table. The default edge index is 1, and if the edge index specified is not available on the workstation, then 1 is used.

modified 2 April 1993
An edge representation contains values for the following attributes:

- edge flag
- edgetype
- edgewidth scale factor
- edge colour index

When the ASFs are INDIVIDUAL, the attributes come from the appropriate SET attribute elements.

Edge representations are defined with SET EDGE REPRESENTATION.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)
100 Ignoring function, the bundle index value is less than one

**SEE ALSO**

- SET EDGE FLAG (3P)
- SET INDIVIDUAL ASF (3P)
- SET EDGE REPRESENTATION (3P)
- SET EDGE REPRESENTATION PLUS (3PP)
## NAME

SET EDGE REPRESENTATION – define edge attribute bundle on workstation

## SYNOPSIS

### C Syntax

```c
void pset_edge_rep ( ws, index, rep )
Pint ws;    /* workstation identifier */
Pint index; /* edge bundle index */
Pedge_bundle *rep; /* edge representation pointer */
```

### FORTRAN Syntax

```fortran
SUBROUTINE psedr ( WKID, EDI, EDFLAG, EDTYPE, EWIDTH, COLI )
INTEGER WKID    /* workstation identifier */
INTEGER EDI    /* edge index */
INTEGER EDFLAG /* edge flag (POFF, PON) */
INTEGER EDTYPE /* edgetype */
REAL EWIDTH    /* edgewidth scale factor */
INTEGER COLI   /* edge colour index */
```

## Required PHIGS Operating States

(PHOP, WSOP, *, *)

## DESCRIPTION

### Purpose

SET EDGE REPRESENTATION defines a bundle of edge attributes for a specified entry in the workstation’s edge bundle table. Depending on the ASF for each of the edge attributes, the bundled attributes may apply to the following primitives:

- **FILL AREA SET**
- **FILL AREA SET 3**
- **GENERALIZED DRAWING PRIMITIVE (Fill Area Primitives)**
- **GENERALIZED DRAWING PRIMITIVE 3 (Fill Area Primitives)**
- **NON-UNIFORM B-SPLINE SURFACE**
- **SET OF FILL AREA SET 3 WITH DATA**
- **QUADRILATERAL MESH 3 WITH DATA**
- **TRIANGLE STRIP 3 WITH DATA**

### C Input Parameters

- **ws** The identifier of the workstation for which the edge representation is being defined.
- **index** The *edge index* of the entry being defined.
- **rep** A pointer to a structure containing the attribute values defining the edge representation, defined as follows:

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

modified 2 April 1993
rep->flag is an enumerated value, specifying whether edges are to be displayed or not. Valid values are:

- PEDGE_OFF     Off
- PEDGE_ON      On

rep->type is an enumerated value, specifying one of the following:

1. PLINE_SOLID    Solid
2. PLINE_DASH     Dashed
3. PLINE_DOT      Dotted
4. PLINE_DASH_DOT Dot-dashed
0. PLINE_LONG_DASH Long-dashed
−1. PLINE_DOT_DASH_DOT Dot-dashed-dotted
−2. PLINE_CENTER Center (long-short dashed)
−3. PLINE_PHANTOM Phantom (long-short-short dashed)

Support for edgetypes is workstation-dependent.

rep->width is an edgewidth scale factor value.

rep->colr_ind is the edge colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameters

WKID    The identifier of the workstation for which the edge representation is being defined.
EDI    The edge index of the entry being defined.
EDFLAG   The edge flag is an enumerated value, specifying whether edges are to be displayed or not. Valid values are:

- POFF     Off
- PON      On

EDTYPE   The edgetype specifies one of the following:

1. PLSOLI     Solid
2. PLDASH     Dashed
3. PLDOT      Dotted
4. PLDASD     Dot-dashed
0. PLNLONGDASH Long-dashed
−1. PLNDOTDASHDOT Dot-dashed-dotted
−2. PLNCENTER Center (long-short dashed)
−3. PLNPHANTOM Phantom (long-short-short dashed)

Support for edgetypes is workstation-dependent. See SET EDGETYPE(3P) for caveats regarding the Center and Phantom edgetypes.

EWIDTH   The edgewidth scale factor value.
COLI The edge colour index, which selects a colour value from the workstation’s colour table.

Execution When SET EDGE REPRESENTATION is called, the edge index entry in the table of defined edge representations on the workstation is set to the edge flag, edgetype, edgewidth scale factor, and edge colour index values.

When area-defining output primitives are displayed, the edge representation specified by the current edge index entry in the PHIGS traversal state list provides the edge attributes for which the Aspect Source Flag (ASF) is BUNDLED. For example, when the current edgetype ASF is set to BUNDLED, the effective edgetype is the edgetype attribute in the edge representation selected by the current edge index. The current edge index is set by SET EDGE INDEX elements.

The edge bundle table is numbered from 1.

See GENERALIZED DRAWING PRIMITIVE(3P) and GENERALIZED DRAWING PRIMITIVE 3(3P) to determine which of the generalized primitives use the fill area set attributes.

ERRORS 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100 Ignoring function, the bundle index value is less than one
103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
113 Ignoring function, the colour index value is less than zero
107 Ignoring function, the specified edge type is not available on the specified workstation

SEE ALSO SET EDGE INDEX (3P)
SET EDGE FLAG (3P)
SET EDGE COLOUR INDEX (3P)
SET INDIVIDUAL ASF (3P)
INQUIRE EDGE REPRESENTATION (3P)
SET EDGE REPRESENTATION PLUS (3PP)
**NAME**

SET EDGETYPE – create structure element to set current `edgetype` attribute

**SYNOPSIS**

**C Syntax**

```c
void pset_edgetype ( edgetype )
Pint edgetype; edgetype
```

**FORTRAN Syntax**

```fortran
SUBROUTINE psedt ( EDTYPE )
INTEGER EDTYPE edgetype
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

SET EDGETYPE creates a structure element structure containing a value for the current edgetype attribute.

When the current edgetype Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute defines the edgetype to be applied to the following output primitives:

- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- GENERALIZED DRAWING PRIMITIVE
- GENERALIZED DRAWING PRIMITIVE 3
- NON-UNIFORM B-SPLINE SURFACE
- SET OF FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

**C Input Parameter**

`edgetype`

An integer value specifying one of the following edge types:

1. PLINE_SOLID `Solid`
2. PLINE_DASH `Dashed`
3. PLINE_DOT `Dotted`
4. PLINE_DASH_DOT `Dot-dashed`
0. PLINE_LONG_DASH `Long-dashed`
-1. PLINE_DOT_DASH_DOT `Dot-dashed-dotted`
-2. PLINE_CENTER `Center (long-short dashed)`
-3. PLINE_PHANTOM `Phantom (long-short-short dashed)`

Support for edgetypes is workstation dependent.

**FORTRAN Input Parameter**

`EDTYPE`

The edgetype specifies one of the following:

830 modified 2 April 1993
<table>
<thead>
<tr>
<th>Edgetype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PLSOLI</td>
<td>Solid</td>
</tr>
<tr>
<td>2 PLDASH</td>
<td>Dashed</td>
</tr>
<tr>
<td>3 PLDOT</td>
<td>Dotted</td>
</tr>
<tr>
<td>4 PLDASD</td>
<td>Dot-dashed</td>
</tr>
<tr>
<td>0 PLNLONGDASH</td>
<td>Long-dashed</td>
</tr>
<tr>
<td>−1 PLNDOTDASHDOT</td>
<td>Dot-dashed-dot-dotted</td>
</tr>
<tr>
<td>−2 PLNCENTER</td>
<td>Center (Long-short Dashed)</td>
</tr>
<tr>
<td>−3 PLNPHANTOM</td>
<td>Phantom (Long-short-short Dashed)</td>
</tr>
</tbody>
</table>

Support for edgetypes is workstation dependent.

**Execution**

If the current edit mode is INSERT, a SET EDGETYPE element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET EDGETYPE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET EDGETYPE element is traversed, the current edgetype entry in the PHIGS traversal state list is set to edgetype. When the edge flag attribute is set to ON and the current edgetype ASF is set to INDIVIDUAL, the edges of the fill area set output primitives that follow in the structure network are drawn with the edgetype specified. See GENERALIZED DRAWING PRIMITIVE and GENERALIZED DRAWING PRIMITIVE 3 to determine which of the generalized primitives use the fill area set attributes.

If the edge flag is set to OFF, then these primitives are drawn without edges, and only the interior fill is displayed. If the edge flag is set to OFF and the interior style is Empty, then the primitive will not be visible.

When the edge flag is ON and the current edgetype ASF is set to BUNDLED, the edgetype is taken from the workstation’s representation indicated by the current edge index. In this case, the edgetype set with SET EDGETYPE has no effect.

If the edgetype specified is not available on the workstation to which the structure is posted, then edgetype 1 (Solid) is used.

The Center and Phantom edgetypes are intended to support Line Conventions and Lettering standards ANSI Y14.2M - 1979 and ISO DIS 128. (Center lines are used to represent axes of symmetrical parts and features, bold circles, and paths of motion. Phantom lines are used to indicate alternate positions of moving parts, adjacent positions of related parts, and repeated detail.) The length of the long dashes are intended to vary in length, depending on the length of the line. Both center lines and phantom lines should start and end with long dashes. Very short center lines may be unbroken. Even when these edgetypes are supported by a workstation, adjustment of the long dash lengths in this way is workstation dependent. No workstation type presently varies the long dash lengths.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)
SEE ALSO

SET EDGE FLAG (3P)
SET INDIVIDUAL ASF (3P)
SET EDGE REPRESENTATION (3P)
SET REFLECTANCE PROPERTIES (3PP)
GENERALIZED DRAWING PRIMITIVE (3P)
GENERALIZED DRAWING PRIMITIVE 3 (3P)
NAME

SET EDGEWIDTH SCALE FACTOR – create structure element to set current edgewidth scale factor attribute

SYNOPSIS

C Syntax

void pset_edgewidth ( scale )
Pfloat scale; edgewidth scale factor

FORTRAN Syntax

SUBROUTINE psewsc ( EWIDTH )
REAL EWIDTH edgewidth scale factor

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET EDGEWIDTH SCALE FACTOR creates a structure element containing a value for the current edgewidth scale factor attribute. When the current edgewidth scale factor Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute defines the current edgewidth applied to the output primitives:

- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- GENERALIZED DRAWING PRIMITIVE (Fill Area Primitives)
- GENERALIZED DRAWING PRIMITIVE 3 (Fill Area Primitives)
- NON-UNIFORM B-SPLINE SURFACE
- SET OF FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

C Input Parameter

scale A real value specifying the edgewidth scale factor.

FORTRAN Input Parameter

EWIDTH A real value specifying the edgewidth scale factor.

Execution

If the current edit mode is INSERT when SET EDGEWIDTH SCALE FACTOR is called, then a SET EDGEWIDTH SCALE FACTOR element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET EDGEWIDTH SCALE FACTOR element is traversed, the current edgewidth scale factor entry in the PHIGS traversal state list is set to edgewidth scale factor.

When the edge flag is set to ON and the current edgewidth scale factor ASF is set to INDIVIDUAL, the edges of the fill area set output primitives that follow in the structure network are drawn with the edgewidth specified.

modified 2 April 1993
If the edge flag is OFF, then these primitives are drawn without edges and only the interior fill is displayed. When the edge flag is ON and the edgewidth scale factor ASF is set to BUNDLED, the edgewidth scale factor is taken from the workstation’s representation indicated by the current edge index. In this case, the edgewidth value set with SET EDGEWIDTH SCALE FACTOR has no effect.

The width of the edge drawn is determined by applying the current edgewidth scale factor to the nominal edgewidth, and this result is then mapped to the nearest edgewidth supported on the workstation.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

SET EDGE FLAG (3P)
SET INDIVIDUAL ASF (3P)
SET EDGE REPRESENTATION (3P)
NAME

SET EDIT MODE – set edit mode to control addition of new structure elements to open structure

SYNOPSIS

C Syntax

void

pset_edit_mode ( mode )

Pedit_mode mode; edit mode

FORTRAN Syntax

SUBROUTINE psedm ( EDITMO )

INTEGER EDITMO edit mode (PINSRT, PREPLC)

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

The SET EDIT MODE subroutine sets the edit mode controlling how PHIGS subroutines that create new structure elements will add the new element to the open structure. The mode may be Insert or Replace.

Insert is the default.

C Input Parameter

mode An enumerated variable specifying the edit mode to be used to add new elements to the open structure. Valid values (defined in phigs.h) are:

PEDIT_INSERT
PEDIT_REPLACE

FORTRAN Input Parameter

EDITMO An enumerated variable specifying the edit mode to be used to add new elements to the open structure. Valid values (defined in phigs77.h) are:

PINSRT Insert
PREPLC Replace

Execution

The SET EDIT MODE subroutine sets the edit mode in the PHIGS state list to Insert or Replace. The value in this entry controls how PHIGS subroutines that create new structure elements add the new element to the currently open structure.

While the edit mode is Insert, new structure elements are inserted into the open structure after the element pointed to by the current element pointer.

While the edit mode is Replace, new structure elements replace the element in the open structure pointed to by the current element pointer.

modified 2 April 1993
ERRORS 002 Ignoring function, function requires state (PHOP, *, *, *)

SEE ALSO

INQUIRE EDIT MODE (3P)
DELETE ELEMENT (3P)
<table>
<thead>
<tr>
<th>NAME</th>
<th>SET ELEMENT POINTER – set element pointer to specified pointer value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td></td>
</tr>
<tr>
<td>C Syntax</td>
<td><code>void pset_elem_ptr ( ep_value )</code></td>
</tr>
<tr>
<td></td>
<td><code>Pint ep_value; element pointer value</code></td>
</tr>
<tr>
<td>FORTRAN Syntax</td>
<td><code>SUBROUTINE psep ( EP )</code></td>
</tr>
<tr>
<td></td>
<td><code>INTEGER EP element position</code></td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Use SET ELEMENT POINTER to set the element pointer in the currently open structure to point to the element numbered <code>element pointer value</code>. Editing of the structure contents, such as adding or deleting elements, is done in relation to the current element pointer position.</td>
</tr>
<tr>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>C Input Parameter</td>
<td><code>ep_value</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the number of the structure element in the currently-open structure to which the element pointer should be set.</td>
</tr>
<tr>
<td>FORTRAN Input Parameter</td>
<td><code>EP</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the number of the structure element in the currently-open structure to which the element pointer should be set.</td>
</tr>
<tr>
<td>Execution</td>
<td>The SET ELEMENT POINTER subroutine sets the current element pointer to <code>element pointer value</code>. The element pointer value is the number of the element in the currently-open structure at which operations on the structure contents begin. For example, subroutines which add new structure elements put the new element into the currently-open structure in relation to the element pointer; the new element is inserted into the structure after the element pointed to by the element pointer or replaces the element pointed to by the element pointer. If the pointer value is specified as less than 0, the element pointer is set to 0. If <code>element pointer value</code> is greater than the number of elements in the open structure, the element pointer will be set to the last element in the structure.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>005 Ignoring function, function requires state (PHOP, *, STOP, *)</td>
</tr>
</tbody>
</table>

modified 2 April 1993
SEE ALSO

OFFSET ELEMENT POINTER (3P)
INQUIRE ELEMENT POINTER (3P)
INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
SET ELEMENT POINTER AT LABEL (3P)
SET ELEMENT POINTER AT PICK IDENTIFIER (3PP)
ELEMENT SEARCH (3P)
NAME

SET ELEMENT POINTER AT LABEL – set element pointer to next occurrence of specified label

SYNOPSIS

C Syntax

void
pset_elem_ptr_label ( label_id )

Pint  label_id;  label identifier

FORTRAN Syntax

SUBROUTINE pseplb ( LABEL )

INTEGER LABEL  label identifier

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET ELEMENT POINTER AT LABEL sets the element pointer of the currently-open structure to the next occurrence in the structure of an element containing the specified label identifier. Label structure elements are created by the LABEL subroutine.

C Input Parameter

label_id Specifies the label identifier in the open structure at which to set the element pointer.

FORTRAN Input Parameter

LABEL Specifies the label identifier in the open structure at which to set the element pointer.

Execution

SET ELEMENT POINTER AT LABEL searches for the label identifier from the structure element following the current element pointer to the end of the open structure. If an occurrence of label identifier is not found between the current element pointer and the end of the structure, the subroutine does not move the element pointer and generates an error.

ERRORS

005  Ignoring function, function requires state (PHOP, *, STOP, *)

205  Ignoring function, the label does not exist in the open structure between the element pointer and the end of the structure

SEE ALSO

OFFSET ELEMENT POINTER (3P)
INQUIRE ELEMENT POINTER (3P)
INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
SET ELEMENT POINTER (3P)
SET ELEMENT POINTER AT PICK IDENTIFIER (3PP)
ELEMENT SEARCH (3P)
NAME
SET ERROR HANDLING – Set the error handling function – C binding only

SYNOPSIS
C Syntax

void
pset_err_hand ( new_err_hand, old_err_hand )
void (∗new_err_hand) (); → application’s error handling function
void (∗∗old_err_hand) (); → OUT old error handling function

Required PHIGS Operating States
(*, *, *, *)

DESCRIPTION
SET ERROR HANDLER sets the error handling function that will be called when PHIGS reports an error. The default function is perr_hand. See the ERROR HANDLING man page for a description of the parameters passed to the error handling function.

Purpose
Use SET ERROR HANDLER is used to set the error handling function to an application’s own error handling function.

When the error handling mode is ON, ERROR HANDLING is called by PHIGS functions when they detect an error.

An application may replace the default ERROR HANDLING function, using this function, in order to modify the default response to errors, which is simply to call ERROR LOGGING. Any such replacement must have the name above and accept the same parameters.

C Input Parameters
new_err_hand
A function that is to be the new error handling function.

old_err_hand
A pointer to a function pointer where the current error handling function is returned.

Execution
SET ERROR HANDLER sets the error handling function that will be called when PHIGS reports an error. PHIGS calls the error handling function when an error is detected in a non-inquiry function and the error handling mode is ON.

See OPEN PHIGS for a description of the PHIGS interaction with the error file.

ERRORS
No Errors

SEE ALSO
ERROR HANDLING (3P)
ERROR LOGGING (3P)
SET ERROR HANDLING MODE (3P)
**NAME**
SET ERROR HANDLING MODE – set current error handling mode

**SYNOPSIS**

**C Syntax**
```c
void
pset_err_hand_mode ( mode )
Perr_mode mode;  error handling mode
```

**FORTRAN Syntax**
```fortran
SUBROUTINE pserhm ( ERHM )
INTEGER ERHM  error handling mode (POFF, PON)
```

**Required PHIGS Operating States**
(PHOP, *, *, *)

**DESCRIPTION**

**Purpose**
Use SET ERROR HANDLING MODE to enable (ON) or disable (OFF) the PHIGS ERROR HANDLING routine.

The default error handling mode is ON.

**C Input Parameter**
`mode` An enumerated value specifying whether PHIGS error reporting should be enabled or disabled. Valid values (defined in phigs.h) for this parameter are:
- PERR_OFF
- PERR_ON

**FORTRAN Input Parameter**
`ERHM` An enumerated variable specifying whether the PHIGS error handling system is on or off. Valid values (defined in phigs77.h) for this parameter are:
- POFF
- PON

**Execution**
If the error handling mode is on and an error is detected by a PHIGS subroutine, the subroutine calls the ERROR HANDLING subroutine and performs a built in error reaction (this generally consists of any cleanup operations that are possible).

If the error handling mode is off, the PHIGS subroutine which detects an error performs its built-in error reaction, but does not call ERROR HANDLING.

When the ERROR HANDLING subroutine is called, it accepts the identification of the error condition, the identification of the subroutine calling it, and the error file from the calling subroutine. The default PHIGS ERROR HANDLING routine simply calls the ERROR LOGGING procedure to write this information to the error file.

**ERRORS**
002 Ignoring function, function requires state (PHOP, *, *, *)
SEE ALSO

ERROR HANDLING (3P)
INQUIRE ERROR HANDLING MODE (3P)
ESCAPE -1 (3P)
OPEN PHIGS (3P)
NAME

SET GLOBAL TRANSFORMATION – create structure element containing 2D global modelling transformation matrix

SYNOPSIS

C Syntax

void pset_global_tran ( xform )

Pmatrix xform;  // transformation matrix

FORTRAN Syntax

SUBROUTINE psgmt ( XFRMT )

REAL XFRMT(3, 3)  // transformation matrix

Required PHIGS Operating States

( PHOP, *, STOP, * )

DESCRIPTION

Purpose

SET GLOBAL TRANSFORMATION creates a structure element containing a 2D global modelling transformation matrix, which during traversal will replace the current global modelling transformation. Then the Local Modelling Transformation is composed with the new Global Modelling Transformation to create a new Composite Modelling Transformation, which maps the Modelling Coordinates (MC) used to define individual output primitives to a unified World Coordinate (WC) Space.

If the current edit mode is INSERT, the SET GLOBAL TRANSFORMATION element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the SET GLOBAL TRANSFORMATION element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

xform  // The 3 x 3 homogeneous transformation matrix, of type:

typedef Pfloat Pmatrix[3][3];

FORTRAN Input Parameter

XFRMT  // An array containing the 3 x 3 homogeneous transformation matrix.

Execution

When traversal of a structure begins, the initial current local modelling transformation (L) and the current global modelling transformation (G) are both the 3D, 4 x 4 identity matrix. The ‘Composite modelling transformation’ (C) within a structure traversal is formed by the matrix multiplication of the ‘current local modelling transformation’ (L) and the ‘current global modelling transformation’ (G) as follows:

C ← G × L

When an EXECUTE STRUCTURE element is encountered, one step in the invocation of the referenced structure is to save the current modelling transformations (L, G, and C). The child structure inherits the parent’s composite modelling transformation (C) as its current global modelling transformation (G), and an identity matrix as its initial current local...
modelling transformation \((L)\). The parent and child have equal composite modelling transformations \((C)\), because \(L\) is the identity. After traversal of the child structure network, the saved transformations are restored so that the parent is unaffected by the execution of a child.

When the `SET GLOBAL TRANSFORMATION` element is traversed, first the element’s `transformation matrix` is expanded to the 3D equivalent `transformation matrix` \((T)\) and it replaces the current global modelling transformation \((G)\):

\[
G' \leftarrow T
\]

The current local modelling transformation \((L)\), is then composited with the new current global modelling transformation \((G)\) to calculate the new composite modelling transformation \((C)\):

\[
C \leftarrow G' \times L
\]

The current Composite Modelling Transformation maps the Modelling Coordinates, used to define individual output primitives, to World Coordinates. Mapping the primitives to the World Coordinate Space establishes the relation between different objects of the image by redefining the parts in terms of a unified coordinate space. This allows the application to define different parts of the image in different local Modelling Coordinates convenient to the objects being defined, and then to apply transformations that will map the local coordinate systems of each part to a single World Coordinate \((WC)\) Space. Finally, the viewing mechanism maps \(WC\) to Device Coordinates on the workstation’s physical display surface.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

- `SET LOCAL TRANSFORMATION` (3P)
- `BUILD TRANSFORMATION MATRIX` (3P)
- `TRANSFORM POINT` (3P)
- `SET VIEW REPRESENTATION` (3P)
- `SET GLOBAL TRANSFORMATION` 3 (3P)

modified 2 April 1993
### NAME
SET GLOBAL TRANSFORMATION 3 – create structure element containing 3D global modelling transformation matrix

### SYNOPSIS

#### C Syntax
```c
void pset_global_tran3 ( xform )
Pmatrix3 xform;  // transformation matrix
```

#### FORTRAN Syntax
```fortran
SUBROUTINE psgmt3 ( XFRMT )
REAL XFRMT(4, 4)  // transformation matrix
```

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION

#### Purpose
SET GLOBAL TRANSFORMATION 3 creates a structure element containing a 3D global modelling transformation matrix, which during traversal will replace the current global modelling transformation. Then the Local Modelling Transformation is composed with the new Global Modelling Transformation to create a new Composite Modelling Transformation, which maps the Modelling Coordinates (MC) used to define individual output primitives to a unified World Coordinate (WC) Space.

If the current edit mode is INSERT, the SET GLOBAL TRANSFORMATION 3 element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the SET GLOBAL TRANSFORMATION 3 element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

#### C Input Parameter
- `xform` The 4 × 4 homogeneous transformation matrix (T), of type:
  ```c
typedef Pfloat Pmatrix3[4][4];
```

#### FORTRAN Input Parameter
- `XFRMT` An array containing the 4 × 4 homogeneous transformation matrix (T).

#### Execution
When traversal of a structure begins, the initial current local modelling transformation (L) and the current global modelling transformation (G) are both the 3D, 4 × 4 identity matrix. The composite modelling transformation (C) within a structure traversal is formed by the matrix multiplication of the current local modelling transformation (L) and the current global modelling transformation (G) as follows:

\[ C \leftarrow G \times L \]

When an EXECUTE STRUCTURE element is encountered, one step in the invocation of the referenced structure is to save the current modelling transformations (L, G, and C). The child structure inherits the parent’s composite modelling transformation (C) as its current global modelling transformation (G), and an identity matrix as its initial current local
modelling transformation ($L$). The parent and child have equal composite modelling transformations ($C$), because $L$ is the identity. After traversal of the child structure network, the saved transformations are restored so that the parent is unaffected by the execution of a child.

When the `SET GLOBAL TRANSFORMATION 3` element is traversed, the element’s *transformation matrix* ($T$) replaces the current global modelling transformation ($G$):

$$G' \leftarrow T$$

The current local modelling transformation ($L$), is then composited with the new current global modelling transformation ($G'$) to calculate the new composite modelling transformation ($C$).

$$C \leftarrow G' \times L$$

The current Composite Modelling Transformation maps the Modelling Coordinates, which are used to define individual output primitives, to World Coordinates. Mapping the primitives to the World Coordinate Space establishes the relation between different objects of the image by redefining the parts in terms of a unified coordinate space. This allows the application to define different parts of the image in different local Modelling Coordinates convenient to the objects being defined, and then to apply transformations that will map the local coordinate systems of each part to a single World Coordinate (WC) Space.

Finally, the viewing mechanism maps WC to Device Coordinates on the workstation’s physical display surface.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

- `SET LOCAL TRANSFORMATION 3 (3P)`
- `BUILD TRANSFORMATION MATRIX 3 (3P)`
- `TRANSFORM POINT 3 (3P)`
- `SET VIEW REPRESENTATION 3 (3P)`
- `SET GLOBAL TRANSFORMATION (3P)`
NAME  SET HIGHLIGHTING FILTER – set workstation’s name set filter to determine highlighted primitives

SYNOPSIS

C Syntax

void pset_highl_filter ( ws, filter )

Pint ws;  workstation identifier
Pfilter *filter;  highlighting filter

FORTRAN Syntax

SUBROUTINE pshlft ( WKID, ISN, IS, ESN, ES )

INTEGER WKID  workstation identifier
INTEGER ISN  number of names in the inclusion set
INTEGER IS(ISN)  inclusion set
INTEGER ESN  number of names in the exclusion set
INTEGER ES(ESN)  exclusion set

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET HIGHLIGHTING FILTER sets the workstation’s highlighting filter, which is compared to the traversal-time current name set of each primitive to determine if the primitive is highlighted.

The filter contains an inclusion set and an exclusion set of names. During traversal, a primitive is eligible for highlighting if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. Each name in the inclusion name set and exclusion name set is a small positive integer.

C Input Parameters

ws  The identifier of the workstation whose highlighting filter is to be set.
filter  A pointer to a Pfilter structure containing the inclusion set and exclusion set of names. Pfilter is defined in phigs.h as follows:

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

The Pint_list structure is defined in phigs.h as follows:

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

modified 2 April 1993
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WKID</strong></td>
<td>The identifier of the workstation whose highlighting filter is to be set.</td>
</tr>
<tr>
<td><strong>ISN</strong></td>
<td>The number of names for the inclusion set.</td>
</tr>
<tr>
<td><strong>IS</strong></td>
<td>An array containing the set of ISN names for the inclusion set.</td>
</tr>
<tr>
<td><strong>ESN</strong></td>
<td>The number of names for the exclusion set.</td>
</tr>
<tr>
<td><strong>ES</strong></td>
<td>An array containing the set of ESN names for the exclusion set.</td>
</tr>
</tbody>
</table>

### Execution

SET HIGHLIGHTING FILTER sets the workstation’s highlighting filter, which contains an inclusion set and an exclusion set of names, both empty by default. A primitive is eligible for highlighting if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. This means the exclusion set has precedence over the inclusion set. If the workstation’s highlighting filter inclusion set is empty, then no primitives are eligible for highlighting.

If the current name set is empty, then subsequent primitives are not eligible. When traversal of a posted structure network starts, the current name set is empty. During traversal, the member names specified by the ADD NAMES TO SET element are added to the current name set by the union operation on the sets. REMOVE NAMES FROM SET elements remove names from the current name set.

The actual appearance of highlighting is workstation-dependent. The SunPHIGS technique for highlighting is to override the primitive’s intrinsic colour with the highlighting colour, which is, by default, the maximum colour index in the workstation colour table. The highlighting colour may be changed using the Generalized Structure Element, Set Highlight Color Index.

### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
</tbody>
</table>

### SEE ALSO

- ADD NAMES TO SET (3P)
- REMOVE NAMES FROM SET (3P)
- GENERALIZED STRUCTURE ELEMENT (3P)
- INQUIRE HIGHLIGHTING FILTER (3P)
NAME

SET HLHSR IDENTIFIER – create structure element to set current hidden line and hidden surface removal (HLHSR) identifier attribute

SYNOPSIS

C Syntax

void

pset_hlhsr_id ( id )

Pint id; HLHSR identifier

FORTRAN Syntax

SUBROUTINE pshrid ( HRID )

INTEGER HRID HLHSR identifier

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

When the SET HLHSR IDENTIFIER element is traversed, the current HLHSR identifier entry in the PHIGS traversal state list is set to the value of the HLHSR identifier. The current HLHSR identifier is applied to output primitives that follow in the structure network. If the HLHSR mode is PHIGS_HLHSR_MODE_ZBUFF, then the current HLHSR identifier traversal state from the structure network must be on before enabling HLHSR for the workstation. If the HLHSR mode is PHIGS_HLHSR_MODE_ZBUFF_NO_ID, then HLHSR is performed regardless of the current value of the HLHSR identifier. When the workstation’s HLHSR mode is PHIGS_HLHSR_MODE_NONE, no HLHSR is performed on the workstation.

The HLHSR identifier in the structure network is used in conjunction with the HLHSR mode on the workstation during traversal. Presently, both must be one to enable Hidden Surface Removal.

If the current edit mode is INSERT, then a SET HLHSR IDENTIFIER element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET HLHSR IDENTIFIER element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

id The HLHSR identifier value. Presently supported values are:

0 PHIGS_HLHSR_ID_OFF Remove nothing
1 PHIGS_HLHSR_ID_ON Remove hidden surfaces and hidden lines using Z-buffer

FORTRAN Input Parameter

HRID The HLHSR identifier value. Presently supported values are:

0 PHIGSHLHSRIDOFF Remove nothing
1 PHIGSHLHSRIDON Remove hidden surfaces and hidden lines using Z-buffer

modified 2 April 1993 849
Execution
When the `SET HLHSR IDENTIFIER` element is traversed, the current HLHSR identifier entry in the PHIGS traversal state list is set to HLHSR identifier. The current HLHSR identifier is applied to output primitives which follow in the structure network. The current HLHSR identifier traversal state from the structure network is compared to the current HLHSR mode on the workstation. When the workstation’s HLHSR mode is `OFF`, no HLHSR is performed, regardless of the current HLHSR identifier.

HLHSR is performed using a Z-buffer algorithm.

Z-buffer processing is not applied to primitives filled with hatches.

The HLHSR mode for a workstation is set with the `SET HLHSR MODE` function. The supported HLHSR modes are workstation-dependent.

For systems without Z-buffered hardware, HLHSR is done in software, but with a significant loss of performance.

Some view mapping matrices may cause problems with rendering. See `EVALUATE VIEW MAPPING MATRIX 3 (3P)` for more detailed information.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
- `SET HLHSR MODE (3P)`
- `INQUIRE HLHSR IDENTIFIER FACILITIES (3P)`
- `INQUIRE HLHSR MODE FACILITIES (3P)`
NAME

SET HLHSR MODE – enable or disable hidden line and hidden surface removal (HLHSR) on workstation

SYNOPSIS

C Syntax

void

pset_hlhsr_mode ( ws, mode )

Pint  ws;

Pint  mode;  \textit{HLHSR mode}

FORTRAN Syntax

SUBROUTINE pshrm ( WKID, HRM )

INTEGER  WKID  \textit{workstation identifier}

INTEGER  HRM  \textit{HLHSR mode}

Required PHIGS Operating States

\textit{(PHOP, WSOP, *, *)}

DESCRIPTION

Purpose

The \textit{SET HLHSR MODE} element requests a certain Hidden Line and Hidden Surface Removal (HLHSR) mode for the workstation. The workstation’s current mode enables or disables HLHSR for a workstation. If the HLHSR mode is \texttt{PHIGS_HLHSR_MODE_ZBUFF}, then the current HLHSR identifier traversal state from the structure network must be on before enabling HLHSR for the workstation. If the HLHSR mode is \texttt{PHIGS_HLHSR_MODE_ZBUFF_NO_ID}, HLHSR is performed regardless of the current value of the HLHSR identifier. When the workstation’s HLHSR mode is \texttt{PHIGS_HLHSR_MODE_NONE}, no HLHSR is performed on the workstation.

C Input Parameters

\begin{itemize}
  \item \textit{ws}  The identifier of the workstation whose HLHSR mode is being set.
  \item \textit{mode}  The HLHSR mode value. Presently defined values are:
    \begin{itemize}
      \item 0  \texttt{PHIGS_HLHSR_MODE_NONE}  \texttt{Disable Z-buffering}
      \item 1  \texttt{PHIGS_HLHSR_MODE_ZBUFF}  \texttt{Enable Z-buffering}
      \item 2  \texttt{PHIGS_HLHSR_MODE_PAINTERS}
      \item 3  \texttt{PHIGS_HLHSR_MODE_SCANLINE}
      \item 4  \texttt{PHIGS_HLHSR_MODE_LINEONLY}
      \item 5  \texttt{PHIGS_HLHSR_MODE_ZBUFF_NO_ID}  \texttt{Enable Z-buffering without HLHSR IDs}
    \end{itemize}
\end{itemize}

FORTRAN Input Parameters

\begin{itemize}
  \item \textit{WKID}  The identifier of the workstation whose HLHSR mode is being set.
  \item \textit{HRM}  The HLHSR mode value. Presently defined values are:
    \begin{itemize}
      \item 0  \texttt{PHIGS_HLHSRMDNONE}  \texttt{Disable Z-buffering}
      \item 1  \texttt{PHIGS_HLHSRMDZBUFF}  \texttt{Enable Z-buffering}
      \item 2  \texttt{PHIGS_HLHSRMDPNTR}
      \item 3  \texttt{PHIGS_HLHSRMDSCNL}
      \item 4  \texttt{PHIGS_HLHSRMDLNON}
    \end{itemize}
\end{itemize}

modified 2 April 1993
If the requested HLHSR mode value is supported on the specified workstation, then SET HLHSR MODE immediately sets the requested HLHSR mode entry in the workstation’s state list to the specified mode. The effect of calling SET HLHSR MODE is not visible until the requested HLHSR mode replaces the current HLHSR mode. The time at which this occurs depends on the workstation’s display update state.

This assignment is performed immediately and the HLHSR update state is set to Not Pending if any one of the following is true:

1. The workstation display update state allows update.
2. The workstation modification mode is any value other than No Immediate Visual Effect, and the dynamic modification accepted for HLHSR mode entry in the workstation description table is set to Immediate.
3. The display space empty status in the workstation state list is EMPTY.

Otherwise, the HLHSR update state is set to Pending and the requested HLHSR mode will not replace the current HLHSR mode until the next time the workstation is updated. The HLHSR update state will be set to Not Pending at that time.

During traversal, if the HLHSR mode is PHIGS_HLHSR_MODE_ZBUFF, then the current HLHSR identifier traversal state from the structure network must be on for HLHSR to be performed using a Z-buffer algorithm. If the HLHSR mode is PHIGS_HLHSR_MODE_ZBUFF_NO_ID, then HLHSR is performed regardless of the current value of the HLHSR identifier. When the workstation’s HLHSR mode is PHIGS_HLHSR_MODE_NONE, no HLHSR is performed on the workstation.

For systems without Z-buffered hardware, HLHSR is done in software, but with a significant loss of performance.

Some view mapping matrices may cause problems with rendering. See EVALUATE VIEW MAPPING MATRIX 3 (3P) for more detailed information.

### ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

111 Ignoring function, the specified HLHSR mode is not available on the specified workstation

### SEE ALSO

SET HLHSR IDENTIFIER (3P)

INQUIRE HLHSR IDENTIFIER FACILITIES (3P)

INQUIRE HLHSR MODE FACILITIES (3P)

INQUIRE HLHSR MODE (3P)
NAME

SET INDIVIDUAL ASF – create structure element to set current Aspect Source Flag (ASF) for attribute

SYNOPSIS

C Syntax

```c
void pset_indiv_asf ( att_id, att_source )
  Paspect att_id;  /* attribute identifier */
  Pasf att_source; /* attribute source */
```

FORTRAN Syntax

```fortran
SUBROUTINE psiasf ( ASPCID, ASFVAL )
  INTEGER ASPCID  /* aspect identifier */
  INTEGER ASFVAL  /* aspect source flag value (PBUNDL, PINDIV) */
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET INDIVIDUAL ASF creates a structure element containing the Aspect Source Flag (ASF) value for a specified output primitive attribute. The ASF setting for each attribute determines if the value used for that attribute is taken from a defined attribute bundle (for example, from the polymarker representation selected by the current polymarker index) or from an attribute value defined individually (for example, from the current marker type set by SET MARKER TYPE (3P)).

If the current edit mode is INSERT, then the structure element created is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET INDIVIDUAL ASF element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameters

`att_id` An enumerated attribute identifier value specifying the attribute for which the ASF is to be set. Possible values are:

```c
typedef enum {
  PASPECT_LINETYPE,
  PASPECT_LINEWIDTH,
  PASPECT_LINE_COLR_IND,
  PASPECT_MARKER_TYPE,
  PASPECT_MARKER_SIZE,
  PASPECT_MARKER_COLR_IND,
  PASPECT_TEXT_FONT,
  PASPECT_TEXT_PREC,
  PASPECT_CHAR_EXPAN,
  PASPECT_CHAR_SPACE,
  PASPECT_TEXT_COLR_IND,
};
```
PASPECT_INT_STYLE,
PASPECT_INT_STYLE_IND,
PASPECT_INT_COLR_IND,
PASPECT_EDGE_FLAG,
PASPECT_EDGETYPE,
PASPECT_EDGECOLOR,
PASPECT_EDGECOLOR_IND,
PASPECT_LINE_SHAD_METH,
PASPECT_INT_SHAD_METH,
PASPECT_REFL_PROPS,
PASPECT_INT_REFL_EQN,
PASPECT_BACK_INT_STYLE,
PASPECT_BACK_INT_STYLE_IND,
PASPECT_BACK_INT_COLR,
PASPECT_BACK_INT_SHAD_METH,
PASPECT_BACK_REFL_PROPS,
PASPECT_BACK_INT_REFL_EQN,
PASPECT_CURVE_APPROX_CRIT,
PASPECT_SURF_APPROX_CRIT

) Paspect;

att_source
An enumerated ASF value specifying the source that should be used for the
attribute identifier or att_id. Values defined are:

typedef enum {
    PASF_BUNDLED,
    PASF_INDIV
} Pasf;

FORTRAN Input
Parameters
All of the following data types are predefined in phigs77.h.

ASPCID
An enumerated attribute identifier value specifying the attribute for which the
ASF is to be set. Valid values defined are:

0 PLN Linetype
1 PLWSC Linewidth scale factor
2 PPLCI Polyline colour index
3 PMK Markertype
4 PMKSC Markersize scale factor
5 PMPCI Polymarker colour index
6 PTXFN Text font
7 PTXPR Text precision
8 PCHXP Character expansion factor
9 PCHSP Character spacing
10 PTXCI Text colour index

modified 2 April 1993
The current setting of the ASF for an attribute determines whether the current value of that attribute is taken from an attribute bundle or from an individually specified value set by the appropriate SET INDIVIDUAL ASF attribute function.

For example, when the current text font ASF is set to INDIVIDUAL, the value of the current text font (as set by SET TEXT FONT) is used for text output primitives. When the current text font ASF is set to BUNDLED, the text font from the workstation’s text representation indicated by the current text index is used instead.
SET EDGE REPRESENTATION (3P)
SET EDGE REPRESENTATION PLUS (3PP)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET POLYLINE REPRESENTATION PLUS (3PP)
SET POLYMARKER REPRESENTATION PLUS (3PP)
SET TEXT REPRESENTATION PLUS (3PP)
NAME

SET INTERIOR COLOUR INDEX – create structure element to set current interior colour index attribute

SYNOPSIS

C Syntax

void
pset_int_colr_ind ( index )

Pint index;  interior colour index

FORTRAN Syntax

SUBROUTINE psici ( COLI )

INTEGER COLI  interior colour index

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET INTERIOR COLOUR INDEX creates a structure element containing a value for the current interior colour index attribute.

When the current interior colour index Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute indexes the current colour representation which applies to the following output primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- SET OF FILL AREA SET 3 WITH DATA
- NON-UNIFORM B-SPLINE SURFACE
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

C Input Parameter

index  An integer colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameter

COLI  An integer colour index, which selects a colour value from the workstation’s colour table.

Execution

If the current edit mode is INSERT, a SET INTERIOR COLOUR INDEX element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET INTERIOR COLOUR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

modified 2 April 1993

857
When the `SET INTERIOR COLOUR INDEX` element is traversed, the current interior colour index entry in the PHIGS traversal state list is set to the colour index.

When the interior colour index ASF is set to `INDIVIDUAL`, the fill area output primitives which follow in the structure network are filled with the colour representation selected by the current interior colour index from the workstation’s colour table.

If the colour index specified is not available on the workstation, colour index 1 will be used.

When the current interior colour index ASF is set to `BUNDLED`, the interior colour index is taken from the workstation’s representation indicated by the current interior index. In this case, the colour index set with `SET INTERIOR COLOUR INDEX` has no effect.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

- `SET COLOUR REPRESENTATION (3P)`
- `SET INTERIOR STYLE (3P)`
- `SET INDIVIDUAL ASF (3P)`
- `SET INTERIOR REPRESENTATION (3P)`
- `SET INTERIOR COLOUR (3PP)`
<table>
<thead>
<tr>
<th>NAME</th>
<th>SET INTERIOR INDEX – create structure element containing <em>interior representation index</em> attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td><strong>C Syntax</strong></td>
</tr>
<tr>
<td></td>
<td><code>void pset_int_ind ( index )</code></td>
</tr>
<tr>
<td></td>
<td><code>Pint index; *interior index</code></td>
</tr>
<tr>
<td></td>
<td><strong>FORTRAN Syntax</strong></td>
</tr>
<tr>
<td></td>
<td><code>SUBROUTINE psii ( II )</code></td>
</tr>
<tr>
<td></td>
<td><code>INTEGER II *interior index</code></td>
</tr>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td></td>
<td>SET INTERIOR INDEX creates a structure element containing an interior index number, which selects an entry from the workstation’s interior bundle table. This attribute applies to the output primitives</td>
</tr>
<tr>
<td></td>
<td>Fill Area</td>
</tr>
<tr>
<td></td>
<td>Fill Area 3</td>
</tr>
<tr>
<td></td>
<td>Fill Area Set</td>
</tr>
<tr>
<td></td>
<td>Fill Area Set 3 with Data</td>
</tr>
<tr>
<td></td>
<td>Set of Fill Area Set 3 with Data</td>
</tr>
<tr>
<td></td>
<td>Non-uniform B-spline Surface</td>
</tr>
<tr>
<td></td>
<td>Quadrilateral Mesh 3 with Data</td>
</tr>
<tr>
<td></td>
<td>Triangle Strip 3 with Data</td>
</tr>
<tr>
<td>C Input Parameter</td>
<td>index  An <em>interior index</em> for an interior representation on the workstation.</td>
</tr>
<tr>
<td>FORTRAN Input Parameter</td>
<td>II  An <em>interior index</em> for an interior representation on the workstation.</td>
</tr>
<tr>
<td>Execution</td>
<td>If the current edit mode is INSERT, a SET INTERIOR INDEX element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET INTERIOR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.</td>
</tr>
<tr>
<td></td>
<td>When the SET INTERIOR INDEX element is traversed, the current interior index value is set to the <em>interior index</em>, which specifies an entry from the workstation’s interior bundle table. The default <em>interior index</em> is 1, and if the <em>interior index</em> specified is not available on the workstation, 1 will be used.</td>
</tr>
</tbody>
</table>

modified 2 April 1993
An interior representation contains values for the following attributes:
- interior style
- interior style index
- interior colour index

Area-defining output primitives which follow in the structure network will use the values from the specified representation for those interior attributes whose Aspect Source Flag (ASF) is set to BUNDLED. When the ASFs are INDIVIDUAL, the attributes come from the appropriate SET attribute elements.

Interior representations are defined with SET INTERIOR REPRESENTATION.

**ERRORS**
005 Ignoring function, function requires state (PHOP, *, STOP, *)
100 Ignoring function, the bundle index value is less than one

**SEE ALSO**
- SET INDIVIDUAL ASF (3P)
- SET INTERIOR REPRESENTATION (3P)
- SET INTERIOR REPRESENTATION PLUS (3PP)

modified 2 April 1993
NAME
SET INTERIOR REPRESENTATION – define interior attribute bundle on workstation

SYNOPSIS
C Syntax
void
pset_int_rep ( ws, index, rep )
Pint ws;  // workstation identifier
Pint index;  // interior bundle index
Pint_bundle *rep;  // interior representation pointer

FORTRAN Syntax
SUBROUTINE psir ( WKID, II, INTS, STYLI, COLI )
INTEGER WKID  // workstation identifier
INTEGER II  // interior index
INTEGER INTS  // interior style (PHOLLO, PSOLID, PPATTR, PHATCH, PISEMP)
INTEGER STYLI  // style index
INTEGER COLI  // colour index

Required PHIGS Operating States
( PHOP, WSOP, *, * )

DESCRIPTION
Purpose
SET INTERIOR REPRESENTATION defines a bundle of interior attributes for a specified entry in the workstation’s interior bundle table.

Depending on the Aspect Source Flag (ASF) for each of the interior attributes, the bundled attributes may apply to the following primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- SET OF FILL AREA SET 3 WITH DATA
- NON-UNIFORM B-SPLINE SURFACE
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

C Input Parameters
ws  // The identifier of the workstation for which the interior representation is being defined.
index  // The interior index of the entry being defined.
rep  // A pointer to a Pint_bundle structure containing the interior representation attributes. Pint_bundle is defined as:

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

} Pint_bundle;

style is an enumerated value specifying one of the following interior styles. Pint_style is defined as:

typedef enum {
    PSTYLE_HOLLOW, Hollow
    PSTYLE_SOLID, Solid
    PSTYLE_PAT, Patterned
    PSTYLE_HATCH, Hatched
    PSTYLE_EMPTY, Empty
} Pint_style;

style_ind is the interior style index, which selects from the workstation’s PATTERN or HATCH table.

colr_ind is the interior colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The identifier of the workstation for which the interior representation is being defined.</td>
</tr>
<tr>
<td>II</td>
<td>The interior index of the entry being defined.</td>
</tr>
<tr>
<td>INTS</td>
<td>An enumerated interior style value, specifying one of the following:</td>
</tr>
<tr>
<td></td>
<td>0 PHOLLO Hollow</td>
</tr>
<tr>
<td></td>
<td>1 PSOLID Solid</td>
</tr>
<tr>
<td></td>
<td>2 PPATTR Patterned</td>
</tr>
<tr>
<td></td>
<td>3 PHATCH Hatched</td>
</tr>
<tr>
<td></td>
<td>4 PISEMP Empty</td>
</tr>
<tr>
<td>STYLI</td>
<td>The interior style index, which selects from the workstation’s PATTERN or HATCH table.</td>
</tr>
<tr>
<td>COLI</td>
<td>The interior colour index, which selects a colour value from the workstation’s colour table.</td>
</tr>
</tbody>
</table>

Execution

When SET INTERIOR REPRESENTATION is called, the interior index in the table of defined interior representations on the workstation is set to the interior style, interior style index, and interior colour index values.

When area-defining output primitives are displayed, the interior representation specified by the current interior index entry in the PHIGS traversal state list provides the interior attributes for which the ASF is BUNDLED. For example, when the current interior style ASF is set to BUNDLED, the effective interior style is the interior style attribute in the interior representation selected by the current interior index. The current interior index is set by SET INTERIOR INDEX elements.
The interior bundle table is numbered from 1.

<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>059</td>
<td>Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)</td>
</tr>
<tr>
<td>100</td>
<td>Ignoring function, the bundle index value is less than one</td>
</tr>
<tr>
<td>103</td>
<td>Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table</td>
</tr>
<tr>
<td>108</td>
<td>Ignoring function, the specified interior style is not available on the workstation</td>
</tr>
<tr>
<td>112</td>
<td>Ignoring function, the pattern index value is less than one</td>
</tr>
<tr>
<td>113</td>
<td>Ignoring function, the colour index value is less than zero</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET INTERIOR INDEX (3P)
- SET INTERIOR STYLE (3P)
- SET INDIVIDUAL ASF (3P)
- INQUIRE INTERIOR REPRESENTATION (3P)
- SET INTERIOR REPRESENTATION PLUS (3PP)
NAME
SET INTERIOR STYLE – create structure element to set current interior style attribute

SYNOPSIS
C Syntax
void pset_int_style ( style )
    Pint_style style;  interior style

FORTRAN Syntax
SUBROUTINE psis ( INTS )
    INTEGER INTS    interior style

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET INTERIOR STYLE creates a structure element containing a value for the current interior style attribute.

When the current interior style Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the interior style to be applied to the following output primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- NON-UNIFORM B-SPLINE SURFACE
- FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA
- SET OF FILL AREA SET 3 WITH DATA

C Input Parameters
All of the following data types are predefined in phigs.h.

style  An enumerated value specifying one of the following interior styles:

```c
typedef enum {
    PSTYLE_HOLLOW,  Hollow
    PSTYLE_SOLID,   Solid
    PSTYLE_PAT,     Patterned
    PSTYLE_HATCH,   Hatched
    PSTYLE_EMPTY,   Empty
} Pint_style;
```

FORTRAN Input Parameters
All of the following data types are predefined in phigs77.h.

INTS  An enumerated value specifying one of the following interior styles:

```fortran
0  PHOLLO    Hollow
```
Execution
If the current edit mode is INSERT, a SET INTERIOR STYLE element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET INTERIOR STYLE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET INTERIOR STYLE element is encountered during traversal, the current interior style entry in the PHIGS traversal state list is set to the specified interior style.

When the current interior style ASF is set to INDIVIDUAL, area-defining output primitives which follow in the structure network are filled using the current interior style. The interior styles have the following meanings:

- **Hollow**: No filling, but draw the bounding polyline using the colour specified by the interior colour index. This is the default style and is also used if an interior style is specified that is not available on the workstation.
- **Solid**: Fill the interior using the colour specified by the interior colour index.
- **Patterned**: Fill the interior using the interior style index as an index into the workstation pattern table. See SET PATTERN REPRESENTATION.
- **Hatched**: Fill the interior using the colour specified by the interior colour index and the interior style index as an index into the workstation table of available hatches. Hatches are predefined; they cannot be defined by the application.
- **Empty**: No filling, but draw the edges in accordance with the edge flag. FILL AREA and FILL AREA 3 primitives will be invisible, but could still be pickable. See SET EDGE FLAG for more information.

When the current interior style ASF is set to BUNDLED, the interior style is taken from the workstation representation indicated by the current interior index. In this case, the interior style set with SET INTERIOR STYLE has no effect.

**ERRORS**

- 005 Ignoring function, function requires state (PHOP, *, STOP, *)

modified 2 April 1993
SEE ALSO

SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION (3P)
SET EDGE FLAG (3P)
SET INTERIOR STYLE INDEX (3P)
SET PATTERN REPRESENTATION (3P)
NAME
SET INTERIOR STYLE INDEX – create structure element to set current interior style index attribute

SYNOPSIS
C Syntax
void
pset_int_style_ind ( index )
Pint index; interior style index

FORTRAN Syntax
SUBROUTINE psisi ( ISTYLI )
INTEGER ISTYLI interior style index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET INTERIOR STYLE INDEX creates a structure element containing a value for the current interior style index attribute.

When the current interior style index ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- FILL AREA SET 3 WITH DATA
- NON-UNIFORM B-SPLINE SURFACE
- POLYHEDRON 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

If the current edit mode is INSERT, a SET INTERIOR STYLE INDEX element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET INTERIOR STYLE INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
index The interior style index, which selects from the workstation’s PATTERN or HATCH.

FORTRAN Input Parameter
ISTYLI The interior style index, which selects from the workstation’s PATTERN or HATCH table.

Execution
When the SET INTERIOR STYLE INDEX element is traversed, the current interior style index entry in the PHIGS traversal state list is set to the interior style index. The default interior style index is 1.

modified 2 April 1993
When the current interior style index \textit{ASF} is set to \textit{INDIVIDUAL}, filled area output primitives which follow in the structure network are filled using the interior style index specified. If the interior style index specified is not available on the workstation, then \texttt{HATCH -1} is used.

When the interior style (either individual or bundled, depending on the interior style \textit{ASF}) is \textit{Hollow}, \textit{Solid}, or \textit{Empty}, the interior style index does not apply. When the interior style is \textit{Patterned}, the interior style index determines which of the workstation's pattern table entries is used; the interior style index may be called the pattern index in this case. When the interior style is \textit{Hatched}, the interior style index determines which of the workstation's predefined hatches is used; the interior style index may be called the hatch index in this case.

When the current interior style index \textit{ASF} is set to \textit{BUNDLED}, the interior style index is taken from the workstation representation indicated by the current interior index. In this case, the interior style index set with \texttt{SET INTERIOR STYLE INDEX} has no effect.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

\texttt{SET INTERIOR STYLE (3P)}
\texttt{SET INTERIOR REPRESENTATION PLUS (3PP)}
\texttt{SET PATTERN REPRESENTATION (3P)}
\texttt{INQUIRE INTERIOR FACILITIES (3P)}
\texttt{SET INDIVIDUAL ASF (3P)}
\texttt{SET INTERIOR REPRESENTATION (3P)}
NAME

SET INVISIBILITY FILTER – set workstation’s name set filter to determine invisible primitives

SYNOPSIS

C Syntax

void
pset_invis_filter ( ws, filter )

Pint  ws;  workstation identifier
Pfilter  *filter;  highlighting filter

FORTRAN Syntax

SUBROUTINE psvft ( WKID, ISN, IS, ESN, ES )

INTEGER WKID  workstation identifier
INTEGER ISN  number of names in the inclusion set
INTEGER IS(ISN)  inclusion set
INTEGER ESN  number of names in the exclusion set
INTEGER ES(ESN)  exclusion set

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET INVISIBILITY FILTER sets the workstation’s invisibility filter, which is compared to the traversal-time current name set of each primitive to determine if the primitive is invisible. The filter contains an inclusion set and an exclusion set of names. During traversal, a primitive is invisible if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. This means the exclusion set has precedence over the inclusion set. Each name in the current name set, inclusion set, and exclusion set is a small positive integer.

C Input Parameters

ws  The identifier of the workstation whose invisibility filter is to be set.

filter  A pointer to a Pfilter structure containing the inclusion set and exclusion set of names. Pfilter is defined in phigs.h as follows:
typedef struct {
    Pint_list  incl_set;  /* inclusion set */
    Pint_list  excl_set;  /* exclusion set */
}  Pfilter;
The Pint_list structure is defined in phigs.h as follows:
typedef struct {
    Pint  num_ints;  /* number of Pints in list */
    Pint  *ints;  /* list of integers */
}  Pint_list;

modified 2 April 1993
**FORTRAN Input Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WKID</strong></td>
<td>The identifier of the workstation whose invisibility filter is to be set.</td>
</tr>
<tr>
<td><strong>ISN</strong></td>
<td>The number of names for the inclusion set.</td>
</tr>
<tr>
<td><strong>IS</strong></td>
<td>An array containing the set of <code>ISN</code> names for the inclusion set.</td>
</tr>
<tr>
<td><strong>ESN</strong></td>
<td>The number of names for the exclusion set.</td>
</tr>
<tr>
<td><strong>ES</strong></td>
<td>An array containing the set of <code>ESN</code> names for the exclusion set.</td>
</tr>
</tbody>
</table>

**Execution**

`SET INVISIBILITY FILTER` sets the workstation’s invisibility filter, which contains an inclusion set and an exclusion set of names, both empty by default. A primitive is invisible if at least one name in the current name set is in the inclusion set and no name in the current name set is in the exclusion set. If the workstation’s invisibility filter inclusion set is empty, then no primitives are invisible.

If the current name set is empty, then subsequent primitives are not invisible. When traversal of a posted structure network starts, the current name set is empty. During traversal, the member names specified by the `ADD NAMES TO SET` element are added to the current name set by the union operation on the sets. `REMOVE NAMES FROM SET` elements remove names from the current name set.

**ERRORS**

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **059** Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

**SEE ALSO**

- `ADD NAMES TO SET (3P)`
- `REMOVE NAMES FROM SET (3P)`
- `INQUIRE INVISIBILITY FILTER (3P)`

---

modified 2 April 1993
NAME
SET LINETYPE – create structure element to set current linetype attribute

SYNOPSIS
C Syntax

void
pset_linetype ( linetype )

Pint linetype; linetype

FORTRAN Syntax

SUBROUTINE psln ( LTYPE )
INTEGER LTYPE linetype

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION
Purpose

SET LINETYPE create a structure element containing a value for the current linetype attribute. When the current linetype ASF (Aspect Source Flag) is set to INDIVIDUAL, this attribute defines the linetype to be applied to the output primitives:

POLYLINE
POLYLINE 3
GDP (line primitives)
GDP 3 (line primitives)

If the current edit mode is INSERT, then a SET LINETYPE element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET LINETYPE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
linetype An integer value specifying a linetype; the following line types are defined:

1 PLINE_SOLID Solid
2 PLINE_DASH Dashed
3 PLINE_DOT Dotted
4 PLINE_DASH_DOT Dot-dashed
0 PLINE_LONG_DASH Long-dashed
−1 PLINE_DOT_DASH_DOT Dot-dashed-dotted
−2 PLINE_CENTER Center (long-short dashed)
−3 PLINE_PHANTOM Phantom (long-short-short dashed)

Support for linetypes is workstation dependent.

FORTRAN Input Parameter
LTYPE An integer value specifying a linetype; the following line types are defined:

1 PLSOLI Solid
2 PLDASH Dashed
3 PLDOT Dotted
4 PLDASD Dot-dashed

modified 2 April 1993
Support for linetypes is workstation-dependent.

**Execution**

When the SET LINETYPE element is traversed, the current linetype entry in the PHIGS traversal state list is set to `linetype`. When the current linetype ASF is set to INDIVIDUAL, line output primitives which follow in the structure network will be drawn with the `linetype` specified. See GENERALIZED DRAWING PRIMITIVE (3P) and GENERALIZED DRAWING PRIMITIVE 3 (3P) to determine which of the generalized primitives use the polyline attributes.

When the current linetype ASF is set to BUNDLED, the effective linetype is the linetype attribute in the polyline representation selected by the current polyline index. In this case, the `linetype` set with SET LINETYPE has no effect.

If the `linetype` specified is not available on the workstation, linetype 1 (`Solid`) is used.

The Center and Phantom linetypes are intended to support Line Conventions and Lettering standards ANSI Y14.2M - 1979 and ISO DIS 128. (Center lines are used to represent axes of symmetrical parts and features, bold circles, and paths of motion. Phantom lines are used to indicate alternate positions of moving parts, adjacent positions of related parts, and repeated detail.) The length of the long dashes are intended to vary in length, depending on the length of the line. Both center lines and phantom lines should start and end with long dashes. Very short center lines may be unbroken. Even when these linetypes are supported by a workstation, it is workstation-dependent whether the long dash lengths are adjusted in this way. No workstation type presently varies the long dash lengths.

**Errors**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**See Also**

SET EDGETYPE (3P)

SET INDIVIDUAL ASF (3P)

SET POLYLINE REPRESENTATION (3P)
### NAME
SET LINEWIDTH SCALE FACTOR – create structure element to set current *linewidth scale factor* attribute

### SYNOPSIS
#### C Syntax

```c
void pset_linewidth ( width )
Float width; linewidth scale factor
```

#### FORTRAN Syntax

```fortran
SUBROUTINE pslwsc ( LWIDTH )
REAL LWIDTH linewidth scale factor
```

#### Required PHIGS Operating States

( PHOP, *, STOP, * )

### DESCRIPTION

**Purpose**

SET LINEWIDTH SCALE FACTOR create a structure element containing a value for the *current linewidth scale factor* attribute. When the *linewidth scale factor ASF* is set to INDIVIDUAL, this attribute defines the current linewidth to be applied to the output primitives:

- POLYLINE
- POLYLINE 3
- GDP (line primitives)
- GDP 3 (line primitives)

If the current edit mode is INSERT, a SET LINEWIDTH SCALE FACTOR element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET LINEWIDTH SCALE FACTOR element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

#### C Input Parameter

- `width` A real value specifying the *linewidth scale factor.*

#### FORTRAN Input Parameter

- `LWIDTH` A real value specifying the *linewidth scale factor.*

**Execution**

When the SET LINEWIDTH SCALE FACTOR element is traversed, the *current linewidth scale factor* entry in the PHIGS traversal state list is set to *linewidth scale factor.* When the *linewidth scale factor ASF* (Aspect Source Flag) is set to INDIVIDUAL, line output primitives that follow in the structure network are drawn using the *linewidth scale factor* specified.

The width of the line drawn is determined by applying the current linewidth scale factor to the *nominal linewidth* as defined in the workstation description table and this result is then mapped to the nearest linewidth supported on the workstation. Presently, the nominal line width is 1 pixel, and supported linewidths vary by 1-pixel units.

---

modified 2 April 1993
When the *current linewidth scale factor* ASF is set to BUNDLED, the effective linewidth scale factor is the linewidth attribute in the polyline representation selected by the *current polyline index*. In this case, the *linewidth scale factor* value set with SET LINEWIDTH SCALE FACTOR has no effect.

**ERRORS**  
005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**  
SET EDGEWIDTH SCALE FACTOR (3P)  
SET INDIVIDUAL ASF (3P)  
SET POLYLINE REPRESENTATION (3P)
NAME
SET LOCAL TRANSFORMATION – create structure element containing 2D Local Modelling Transformation matrix

SYNOPSIS
C Syntax

```c
void
pset_local_tran ( xform, compose_type )
```

```c
Pmatrix xform;
transformation matrix

Pcompose_type compose_type;
composition type
```

FORTRAN Syntax

```fortran
SUBROUTINE pslmt ( XFRMT, CTYPE )
```

```fortran
REAL XFRMT(3, 3)  transformation matrix

INTEGER CTYPE  composition type (PCPRE, PCPOST, PCREPL)
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION
Purpose

SET LOCAL TRANSFORMATION creates a structure element containing a 2D Local Modelling Transformation matrix and a composition type, allowing various objects in the image to be defined in various convenient Modelling Coordinate systems. During traversal the element will modify the current local modelling transformation according to the composition type. Then the modified Local Modelling Transformation is composed with the current Global Modelling Transformation to create a new Composite Modelling Transformation. The Composite Modelling Transformation then maps the Modelling Coordinates (MC) used to define individual output primitives to a unified World Coordinate (WC) Space.

If the current edit mode is INSERT, the SET LOCAL TRANSFORMATION element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the SET LOCAL TRANSFORMATION element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameters

```
xform  The 3 × 3 homogeneous transformation matrix, of type:
```

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

```
compose_type
```

The composition type is an enumerated value, one of:

```
PTYPE_PRECONCAT  Preconcatenate
PTYPE_POSTCONCAT  Postconcatenate
PTYPE_REPLACE  Replace
```

FORTRAN Input Parameters

```
XFRMT An array containing the 3 × 3 homogeneous transformation matrix.
```

```
CTYPE  The composition type is an enumerated value, one of:
```

modified 2 April 1993
When traversal of a structure begins, the initial current Local Modelling Transformation (L) and the current Global Modelling Transformation (G) are both the 3D, $4 \times 4$ identity matrix. The Composite Modelling Transformation (C) within a structure traversal is formed by the matrix multiplication of the current Local Modelling Transformation (L) and the current Global Modelling Transformation (G) as follows:

\[ C \leftarrow G \times L \]

When an EXECUTE STRUCTURE element is encountered, one step in the invocation of the referenced structure is to save the current modelling transformations (L, G, and C). The child structure inherits the parent’s Composite Modelling Transformation (C) as its current Global Modelling Transformation (G), and an identity matrix as its initial current Local Modelling Transformation (L). The parent and child have equal Composite Modelling Transformations (C), because L is the identity. After traversal of the child structure network, the saved transformations are restored so that the parent is unaffected by the execution of a child.

When the SET LOCAL TRANSFORMATION element is traversed, first the element’s transformation matrix is expanded to the 3D equivalent transformation matrix (T). Then the compose type specifies the role of the current Local Modelling Transformation (L) in composing the new value for the current Local Modelling Transformation (L'). which is then combined with the current Global Modelling Transformation (G) to calculate the new Composite Modelling Transformation (C).

**Preconcatenate**

The transformation matrix (T) is multiplied by the current Local Modelling Transformation (L):

\[ L' \leftarrow L \times T \]

\[ C \leftarrow G \times L' \]

**Postconcatenate**

The current Local Modelling Transformation (L) is multiplied by the transformation matrix (T):

\[ L' \leftarrow T \times L \]

\[ C \leftarrow G \times L' \]

**Replace**

The transformation matrix (T) replaces the value of current Local Modelling Transformation (L).

\[ L' \leftarrow T \]

\[ C \leftarrow G \times L' \]

The current Composite Modelling Transformation maps the Modelling Coordinates, used to define individual output primitives, to World Coordinates. Mapping the primitives to the World Coordinate Space establishes the relation between different objects of the
image by redefining the parts in terms of a unified coordinate space. This allows the application to define different parts of the image in different local Modelling Coordinates convenient to the objects being defined, and then to apply transformations that will map the local coordinate systems of each part to a single World Coordinate (WC) Space. Finally, the viewing mechanism maps WC to Device Coordinates on the workstation’s physical display surface.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

- SET GLOBAL TRANSFORMATION (3P)
- SET VIEW REPRESENTATION (3P)
- BUILD TRANSFORMATION MATRIX (3P)
- TRANSFORM POINT (3P)
- SET LOCAL TRANSFORMATION 3 (3P)
NAME  SET LOCAL TRANSFORMATION 3 – create structure element containing 3D local modelling transformation matrix

SYNOPSIS

C Syntax

```c
void pset_local_tran3 ( xform, compose_type )
Pmatrix3 xform;  // transformation matrix
Pcompose_type compose_type;  // composition type
```

FORTRAN Syntax

```fortran
SUBROUTINE pslmt3 ( XFRMT, CTYPE )
REAL XFRMT(4, 4)  // transformation matrix
INTEGER CTYPE  // type (PCPRE, PCPOST, PCREPL)
```

Required PHIGS Operating States

( PHOP, *, STOP, * )

DESCRIPTION

Purpose

SET LOCAL TRANSFORMATION 3 creates a structure element containing a 3D local modelling transformation matrix and a composition type, allowing various objects in the image to be defined in various convenient Modelling Coordinate systems.

During traversal the element will modify the current local modelling transformation according to the composition type. Then the modified Local Modelling Transformation is composed with the current Global Modelling Transformation to create a new Composite Modelling Transformation. The Composite Modelling Transformation then maps the Modelling Coordinates (MC) used to define individual output primitives to a unified World Coordinate (WC) Space.

If the current edit mode is INSERT, the SET LOCAL TRANSFORMATION 3 element is inserted into the open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the SET LOCAL TRANSFORMATION 3 element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameters

- `xform` The 4 × 4 homogeneous transformation matrix, of type:

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

- `compose_type`

  The composition type is an enumerated value, one of:

  ```c
  PTYPE_PRECONCAT Preconcatenate
  PTYPE_POSTCONCAT Postconcatenate
  PTYPE_REPLACE Replace
  ```

FORTRAN Input Parameters

- `XFRMT` An array containing the 4 × 4 homogeneous transformation matrix.

- `CTYPE` The composition type is an enumerated value, one of:

  ```fortran
  type (PCPRE, PCPOST, PCREPL)
  ```

modified 2 April 1993
When traversal of a structure begins, the initial current local modelling transformation \( L \) and the current global modelling transformation \( G \) are both the \( 3 \times 4 \) identity matrix. The Composite modelling transformation \( C \) within a structure traversal is formed by the matrix multiplication of the current local modelling transformation \( L \) and the current global modelling transformation \( G \) as follows:

\[
C \leftarrow G \times L
\]

(Despite American convention, the PHIGS standard follows the European convention, which multiplies a transformation matrix by a point represented as a column vector on the right.)

When an EXECUTE STRUCTURE element is encountered, one step in the invocation of the referenced structure is to save the current modelling transformations \( (L, G, \text{ and } C) \). The child structure inherits the parent’s composite modelling transformation \( (C) \) as its current global modelling transformation \( (G) \), and an identity matrix as its initial current local modelling transformation \( (L) \). The parent and child have equal composite modelling transformations \( (C) \), because \( L \) is the identity. After traversal of the child structure network, the saved transformations are restored so that the parent is unaffected by the execution of a child.

When the SET LOCAL TRANSFORMATION 3 element is traversed, the compose type specifies the role of the current local modelling transformation \( (L) \) in composing the new value for the current local modelling transformation \( (L') \), which is then combined with the current global modelling transformation \( (G) \) to calculate the new composite modelling transformation \( (C) \).

**Preconcatenate**

The transformation matrix \( (T) \) is multiplied by the current local modelling transformation \( (L) \):

\[
L' \leftarrow L \times T \\
C \leftarrow G \times L'
\]

**Postconcatenate**

The current local modelling transformation \( (L) \) is multiplied by the transformation matrix \( (T) \):

\[
L' \leftarrow T \times L \\
C \leftarrow G \times L'
\]

**Replace**

The transformation matrix \( (T) \) replaces the value of current local modelling transformation \( (L) \):

\[
L' \leftarrow T \\
C \leftarrow G \times L'
\]
The current Composite Modelling Transformation maps the Modelling Coordinates, used to define individual output primitives, to World Coordinates. Mapping the primitives to the World Coordinate Space establishes the relation between different objects of the image by redefining the parts in terms of a unified coordinate space. This allows the application to define different parts of the image in different local Modelling Coordinates convenient to the objects being defined, and then to apply transformations that will map the local coordinate systems of each part to a single World Coordinate (WC) Space. Finally, the viewing mechanism maps WC to Device Coordinates on the workstation's physical display surface.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

SET GLOBAL TRANSFORMATION 3 (3P)
SET VIEW REPRESENTATION 3 (3P)
BUILD TRANSFORMATION MATRIX 3 (3P)
TRANSFORM POINT 3 (3P)
SET LOCAL TRANSFORMATION (3P)
### NAME
SET LOCATOR MODE – set locator device operating mode and echoing state

### SYNOPSIS

#### C Syntax

```c
void pset_loc_mode ( ws, dev, mode, echo )
```

- `ws`: workstation identifier
- `dev`: locator device number
- `mode`: operating mode
- `echo`: echo switch

#### FORTRAN Syntax

```fortran
SUBROUTINE pslcm ( WKID, LCDNR, MODE, ESW )
```

- `WKID`: workstation identifier
- `LCDNR`: locator device number
- `MODE`: operating mode (PREQU, PSAMPL, PEVENT)
- `ESW`: echo switch (PNECHO, PECHO)

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

#### Purpose
Use the SET LOCATOR MODE subroutine to set the operating mode (Request, Sample, or Event) and the echo switch (Echo, No Echo) for a specified locator device on a specified workstation.

#### C Input Parameters

- `ws`: The workstation identifier of the workstation associated with the device.
- `dev`: The device number of the locator device to be set. See the `AVAILABLE DEVICES` section in INITIALIZE LOCATOR 3 for a description of the available devices.
- `mode`: Specifies the operating mode for the specified locator device. `Pop_mode` is an enumeration defined in `phigs.h` as follows:
  ```c
typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
  } Pop_mode;
  ```
- `echo`: The echo switch value for the specified locator device. `Pecho_switch` is an enumeration defined in `phigs.h` as follows:
  ```c
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
  } Pecho_switch;
  ```

modified 2 April 1993

881
SET LOCATOR MODE (3P)

FORTRAN Input

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>LCDNR</td>
<td>The device number of the locator device to be set. See the AVAILABLE DEVICES section in INITIALIZE LOCATOR 3 for a description of the available devices.</td>
</tr>
<tr>
<td>MODE</td>
<td>The desired operating mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PREQU Request</td>
</tr>
<tr>
<td></td>
<td>PSAMPL Sample</td>
</tr>
<tr>
<td></td>
<td>PEVENT Event</td>
</tr>
<tr>
<td>ESW</td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PNECHO No echo</td>
</tr>
<tr>
<td></td>
<td>PECHO Echo</td>
</tr>
</tbody>
</table>

Execution

The SET LOCATOR MODE sets the operating mode of the specified locator device to Request, Sample, or Event, and the echo switch to Echo or No Echo. The default operating mode is Request. The default echo switch is Echo.

The operating mode controls how the input from the device is obtained.

- If the operating mode is Request, the subroutine REQUEST LOCATOR, or REQUEST LOCATOR 3 may be used to add the specified device number to the device trigger’s list of recipients and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the REQUEST LOCATOR subroutine returns the input value (the current coordinates of the device) and the status OK. If a break occurs, the status NONE is returned.

- If the operating mode is Sample, the SAMPLE LOCATOR or SAMPLE LOCATOR 3 subroutine may be used to return the current input value of the device without waiting for the trigger to fire.

- If the operating mode is Event, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines AWAIT EVENT and/or GET LOCATOR may then be used to read event reports from the queue.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process. The locator echo updates the cursor position.

ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation category is not INPUT or OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
</tbody>
</table>

modified 2 April 1993
SEE ALSO

REQUEST LOCATOR 3 (3P)
GET LOCATOR 3 (3P)
INQUIRE LOCATOR DEVICE STATE 3 (3P)
INITIALIZE LOCATOR 3 (3P)
NAME

SET MARKER SIZE SCALE FACTOR – create structure element to set current marker size scale factor attribute

SYNOPSIS

C Syntax

void
pset_marker_size ( size )
Pfloat size; marker size scale factor

FORTRAN Syntax

SUBROUTINE psmksc ( MSZSF )
REAL MSZSF marker size scale factor

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET MARKER SIZE SCALE FACTOR creates a structure element containing a value for the current marker size scale factor attribute.

When the current marker size scale factor Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the marker size scale factor to be applied to the output primitives:

POLYMARKER
POLYMARKER 3

C Input Parameter

size A real value specifying the marker size scale factor.

FORTRAN Input Parameter

MSZSF A real value specifying the marker size scale factor.

Execution

If the current edit mode is INSERT, a SET MARKER SIZE SCALE FACTOR element is inserted after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET MARKER SIZE SCALE FACTOR element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET MARKER SIZE SCALE FACTOR element is traversed, the current marker size scale factor entry in the PHIGS traversal state list is set to marker size scale factor.

When the current marker size scale factor ASF is INDIVIDUAL, POLYMARKER and POLYMARKER 3 output primitives which follow in the structure network are drawn using the marker size scale factor specified. This scaling factor is multiplied by the nominal marker size defined in the workstation description table, and the nearest available size on the workstation is used. Presently, the nominal marker size is 11 pixels and supported sizes increase in 2 pixel increments.
When the current marker size scale factor ASF is BUNDLED, the effective marker size scale factor is the marker size scale factor attribute in the polymarker representation selected by the current polymarker index. In this case, the marker size scale factor set with SET MARKER SIZE SCALE FACTOR has no effect.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

SET INDIVIDUAL ASF (3P)
SET POLYMARKER REPRESENTATION (3P)

modified 2 April 1993
NAME
SET MARKER TYPE – create structure element to set current marker type attribute

SYNOPSIS
C Syntax

void
pset_marker_type ( markertype )

Pint markertype; marker type

FORTRAN Syntax

SUBROUTINE psmk ( MTYPE )

INTEGER MTYPE marker type

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION
Purpose

SET MARKER TYPE creates a structure element containing a value for the current marker
type attribute.

When the current marker type Aspect Source Flag (ASF) is set to INDIVIDUAL, the current
value of this attribute defines the marker type to be applied to the output primitives:

POLYMARKER
POLYMARKER 3

C Input Parameter

markertype

An integer value specifying a marker type; the following marker types are
defined:

1 PMARKER_DOT Point (.)
2 PMARKER_PLUS Plus sign (+)
3 PMARKER_ASTERISK Asterisk (*)
4 PMARKER_CIRCLE Circle (\(\bigcirc\))
5 PMARKER_CROSS X-mark (\(\times\))
0 PMARKER_SQUARE Square (\(\square\))
−1 PMARKER_BOWTIE_NE Bowtie −1 (\(\bigtriangledown\))
−2 PMARKER_BOWTIE_NW Bowtie −2 (\(\bigtriangleup\))

Support for marker types is workstation dependent.

FORTRAN Input Parameter

MTYPE An integer value specifying a marker type; the following marker types are
defined:

1 PPOINT Point (.)
2 PPLUS Plus sign (+)
3 PAST Asterisk (*)
4 POMARK Circle (\(\bigcirc\))
5 PXMARK X-mark (\(\times\))
0 PSQUARE Square (\(\square\))
Support for marker types is workstation-dependent.

**Execution**

If the current edit mode is INSERT, a SET MARKER TYPE element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET MARKER TYPE element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET MARKER TYPE element is traversed, the current marker type entry in the PHIGS traversal state list is set to marker type.

When the current marker type ASF is set to INDIVIDUAL, POLYMARKER and POLYMARKER 3 output primitives which follow in the structure network are then drawn using the marker type specified.

The default marker type is 3, *Asterisk* (∗). If the marker type specified is not available on the workstation, the default will be used.

When the current marker type ASF is set to BUNDLED, the effective marker type is the marker type attribute in the polymarker representation selected by the current polymarker index. In this case, the marker type set with SET MARKER TYPE has no effect.

**ERRORS**

005 Ignoring function, function requires state (PHOP, ∗, STOP, ∗)

**SEE ALSO**

- SET INDIVIDUAL ASF (3P)
- SET POLYMARKER REPRESENTATION (3P)
NAME
SET MODELLING CLIPPING INDICATOR – create structure element to set current modelling clipping indicator attribute

SYNOPSIS
C Syntax
void
pset_model_clip_ind ( ind )
Pclip_ind ind; clipping indicator

FORTRAN Syntax
SUBROUTINE psmcli ( MCLIPI )
INTEGER MCLIPI modelling clipping indicator (PNCLIP, PCLIP)

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET MODELLING CLIPPING INDICATOR creates a structure element containing a value for the current modelling clipping indicator attribute, which affects all output primitives.

C Input Parameter
ind The clipping indicator is an enumerated value, from:
    PIND_NO_CLIP Do not clip
    PIND_CLIP Clip

FORTRAN Input Parameter
MCLIPI The clipping indicator is an enumerated value, from:
    PNCLIP Do not clip
    PCLIP Clip

Execution
If the current edit mode is INSERT, a SET MODELLING CLIPPING INDICATOR element is inserted into the currently open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

During traversal, this attribute enables or disables modelling clipping to the volume controlled by SET MODELLING CLIPPING VOLUME, SET MODELLING CLIPPING VOLUME 3, and RESTORE MODELLING CLIPPING VOLUME elements.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)
SEE ALSO

INQUIRE MODELLING CLIPPING FACILITIES (3P)
SET MODELLING CLIPPING VOLUME 3 (3P)
RESTORE MODELLING CLIPPING VOLUME (3P)
NAME
SET MODELLING CLIPPING VOLUME – create structure element to set current 2D modelling clipping volume

SYNOPSIS
C Syntax
void
pset_model_clip_vol ( op, half_spaces )

FORTRAN Syntax
SUBROUTINE psmcv ( OP, NHALFS, HALFSP )

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET MODELLING CLIPPING VOLUME creates a structure element containing the operator and half spaces specified, and places the element in the current structure. During traversal, this element is used to modify the value of the current modelling clipping volume attribute, which affects all output primitives.

A SET MODELLING CLIPPING VOLUME element contains two-dimensional half-spaces. Each specified half-space is expanded to a three dimensional half-space by setting the z-coordinate of the point and normal vector to zero. These half-spaces are then transformed by the current composite modelling transformation. These half-spaces are intersected to form a clipping volume. If no half-spaces are provided, then the clipping volume is defined to be all of modelling coordinate space. Clipping volume is combined as specified by OPERATOR with current modelling clipping volume to form a new value of current modelling clipping volume. This volume is used to clip subsequent output primitives during structure traversal. The resulting clipping volume is not affected by subsequent modelling transformation elements encountered during structure traversal.

C Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>The operator that describes the application of the half-spaces in this element to the current modelling clipping volume. Operator values defined in phigs.h are:</td>
</tr>
<tr>
<td></td>
<td>PMC_REPLACE Replace</td>
</tr>
<tr>
<td></td>
<td>PMC_INTERSECT Intersect</td>
</tr>
<tr>
<td>half_spaces</td>
<td>The list of half-spaces to store in the element. Phalf_space_list is defined in phigs.h as:</td>
</tr>
<tr>
<td></td>
<td>typedef struct {</td>
</tr>
<tr>
<td></td>
<td>Pint num_half_spaces; /* number of half-spaces */</td>
</tr>
<tr>
<td></td>
<td>Phalf_space <em>half_spaces; /</em> list of half-spaces */</td>
</tr>
</tbody>
</table>
The number is the number of half-spaces in the array of Phalf_space elements pointed to by half-spaces, as specified by a point, and the normal to the plane containing the point, in the direction of the half-space. Phalf_space is defined in phigs.h as:

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

Ppoint and Pvec are similar structures (but have different semantics), defined by phigs.h:

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

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

The 2D point and normal vector are expanded to 3D by using z components equal to 0.

**FORTRAN Input Parameters**

- **OP** The clipping operator that describes the application of the half spaces in this element to the current modelling clipping volume. Operator values defined in phigs77.h are:
  - PMCREP  Replace
  - PMCINT  Intersect

- **NHALFS** The number of clipping half spaces in the array HALFSP.

- **HALFSP** The list of half spaces to store in the element, specified in a 2D FORTRAN array [4, NHALFS]. The first dimension of HALFSP is 4, where the four entries for one half space are:
  - x coordinate of the point
  - y coordinate of the point
  - x magnitude of the normal to the plane in the direction of the half space (that is, the plane containing the point)
  - y magnitude of the normal to the plane in the direction of the half space

*(modified 2 April 1993)*
that is, the plane containing the point
The second dimension is NHALFS.

Execution
If the current edit mode is INSERT, a SET MODELLING CLIPPING VOLUME element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

During traversal, this element is used to modify the value of the current modelling clipping volume attribute, which affects all output primitives. This volume is used to clip subsequent output primitives during structure traversal. The resulting clipping volume is not affected by subsequent modelling transformation elements encountered during structure traversal.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
INQUIRE MODELLING CLIPPING FACILITIES (3P)
SET MODELLING CLIPPING INDICATOR (3P)
RESTORE MODELLING CLIPPING VOLUME (3P)
SET MODELLING CLIPPING VOLUME 3 (3P)
NAME

SET MODELLING CLIPPING VOLUME 3 – create structure element to set current 3D modelling clipping volume

SYNOPSIS

C Syntax

```c
void pset_model_clip_vol3 ( op, half_spaces )
  Pint op;
  Phalf_space_list3 *half_spaces;
```

FORTRAN Syntax

```fortran
SUBROUTINE psmcv3 ( OP, NHALFS, HALFSP )
  INTEGER OP modelling clipping operator
  INTEGER NHALFS number of modelling clipping half spaces in list
  REAL HALFSP(6, NHALFS) list of modelling clipping half spaces
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET MODELLING CLIPPING VOLUME 3 creates a structure element containing the operator and half spaces specified, and places the element in the current structure. During traversal, this element is used to modify the value of the current modelling clipping volume attribute, which affects all output primitives.

A SET MODELLING CLIPPING VOLUME 3 element contains half-spaces in the model space. Each specified half-space is transformed by the current composite modelling transformation. These half-spaces are intersected to form a clipping volume. If no half-spaces are provided, then the clipping volume is defined to be all of modelling coordinate space. Clipping volume is combined as specified by OPERATOR with current modelling clipping volume to form a new value of current modelling clipping volume. This volume is used to clip subsequent output primitives during structure traversal. The resulting clipping volume is not affected by subsequent modelling transformation elements encountered during structure traversal.

C Input Parameters

`op` The operator that describes the application of the half-spaces in this element to the current modelling clipping volume. Operator values defined in phigs.h are:

- PMC_REPLACE Replace
- PMC_INTERSECT Intersect

`half_spaces` The list of half-spaces to store in the element. Phalf_space_list3 is defined in phigs.h as:

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

The number is the number of half-spaces in the array of Phalf_space3 elements
pointed to by half-spaces, as specified by a point, and the normal to the plane
containing the point, in the direction of the half-space. Phalf_space3 is defined in
phigs.h as:

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

Ppoint3 and Pvec3 are similar structures (but have different semantics), defined
by phigs.h:

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

typedef struct {
    Pfloat delta_x; /* x magnitude */
    Pfloat delta_y; /* y magnitude */
    Pfloat delta_z; /* z magnitude */
} Pvec3;

<table>
<thead>
<tr>
<th>FORTRAN Input Parameters</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP</td>
<td>The clipping operator that describes the application of the half spaces in this element to the current modelling clipping volume. Operator values defined in phigs77.h are:</td>
</tr>
<tr>
<td>PMCREP</td>
<td>Replace</td>
</tr>
<tr>
<td>PMCINT</td>
<td>Intersect</td>
</tr>
<tr>
<td>NHALFS</td>
<td>The number of clipping half spaces in the array HALFSP.</td>
</tr>
<tr>
<td>HALFSP</td>
<td>The list of half spaces to store in the element, specified in a 2D FORTRAN array [6, NHALFS]. The first dimension of HALFSP is 6, where the six entries for one half space are:</td>
</tr>
<tr>
<td></td>
<td>x coordinate of the point</td>
</tr>
<tr>
<td></td>
<td>y coordinate of the point</td>
</tr>
<tr>
<td></td>
<td>z coordinate of the point</td>
</tr>
<tr>
<td></td>
<td>x magnitude of the normal to the plane in the direction of the half space (that is, the plane containing the point)</td>
</tr>
</tbody>
</table>
y magnitude of the normal to the plane in the direction of the half space
(that is, the plane containing the point)

z magnitude of the normal to the plane in the direction of the half space
(that is, the plane containing the point)

The second dimension is NHALFS.

Execution
If the current edit mode is INSERT, a SET MODELLING CLIPPING VOLUME 3 element is inserted
into the currently-open structure after the element pointed to by the current element
pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by
the element pointer. In either case, the element pointer is updated to point to the new
element.

During traversal, this element is used to modify the value of the current modelling
clipping volume attribute, which affects all output primitives. This volume is used to clip
subsequent output primitives during structure traversal. The resulting clipping volume
is not affected by subsequent modelling transformation elements encountered during
structure traversal.

ERRORS
005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO
INQUIRE MODELLING CLIPPING FACILITIES (3P)
SET MODELLING CLIPPING INDICATOR (3P)
RESTORE MODELLING CLIPPING VOLUME (3P)
SET MODELLING CLIPPING VOLUME (3P)
### NAME
SET PATTERN REFERENCE POINT – create structure element to set current pattern reference point attribute

### SYNOPSIS
#### C Syntax
```c
void pset_pat_ref_point ( ref_pt )

Ppoint *ref_pt;  pattern reference point
```

#### FORTRAN Syntax
```fortran
SUBROUTINE psparf ( RFX, RFY )

REAL RFX, RFY  pattern reference point (MC)
```

### Required PHIGS Operating States
(PHOP, *, STOP, *)

### DESCRIPTION
#### Purpose
SET PATTERN REFERENCE POINT creates a structure element containing a value for the current pattern reference point attribute. The structure implicitly resets the current pattern reference vectors attribute as well. When the interior style is PATTERN, these attributes apply to the following output primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- SET OF FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

**Note:** This function places the appropriate data in the Central Structure Store, and the element may be written to SunPHIGS archives, but the structure element is currently ignored during traversal.

If the current edit mode is INSERT, a SET PATTERN REFERENCE POINT element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

#### C Input Parameter
`ref_pt` The pattern reference point is given in Modelling Coordinates (MC), using a 2D shorthand; 0 is used as the z coordinate. The Ppoint type is defined in phigs.h as:

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

---

896 modified 2 April 1993
FORTRAN Input

Parameters

\( RFX \) The \( x \) coordinate of the pattern reference point, in Modelling Coordinates (MC).

\( RFY \) The \( y \) coordinate of the pattern reference point, in Modelling Coordinates (MC).

Execution

When the SET PATTERN REFERENCE POINT element is traversed, the current pattern reference point entry in the PHIGS traversal state list is set to the \( x \) and \( y \) values contained in this element, and the \( z \) value 0. The current pattern reference vectors are set to \((1,0,0)\) and \((0,1,0)\). When the interior style (either individual or bundled, as selected by the current interior style \( ASF \)) is set to PATTERN, these attributes are used in conjunction with the current pattern width and height vectors for the area-defining primitives which follow in the structure network.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INQUIRE PATTERN FACILITIES (3P)
SET PATTERN SIZE (3P)
SET PATTERN REPRESENTATION (3P)
SET PATTERN REFERENCE POINT AND VECTORS (3P)

modified 2 April 1993
### NAME

SET PATTERN REFERENCE POINT AND VECTORS – create structure element to set current pattern reference point and vectors attributes

### SYNOPSIS

**C Syntax**

```c
void pset_pat_ref_point_vecs ( ref_pt, pat_ref_vec )
Ppoint3 *ref_pt; pattern reference point
Pvec3 pat_ref_vec[2]; direction vectors; 1=X axis of pattern, 2=Y axis of pattern
```

**FORTRAN Syntax**

```fortran
SUBROUTINE psprpv ( RFX, RFY, RFZ, RFVX, RFVY, RFVZ )
REAL RFX, RFY, RFZ pattern reference point (MC)
REAL RFVX(2), RFVY(2), RFVZ(2) pattern reference vectors (MC)
```

### Required PHIGS Operating States

(PHOP, *, STOP, *)

### DESCRIPTION

**Purpose**

SET PATTERN REFERENCE POINT AND VECTORS creates a structure element containing a value for the current pattern reference point and current pattern reference vectors attributes. When the interior style is PATTERN, these attributes apply to the following output primitives:

- FILL AREA
- FILL AREA 3
- FILL AREA SET
- FILL AREA SET 3
- SET OF FILL AREA SET 3 WITH DATA
- QUADRILATERAL MESH 3 WITH DATA
- TRIANGLE STRIP 3 WITH DATA

**Note:** This function places the appropriate data in the Central Structure Store, and the element may be written to SunPHIGS archives, but the structure element is currently ignored during traversal.

If the current edit mode is INSERT, a SET PATTERN REFERENCE POINT AND VECTORS element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

**C Input Parameters**

- `ref_pt` The pattern reference point, in Modelling Coordinates (MC). The Ppoint3 structure is defined in phigs.h as:

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

pat_ref_vec
A pointer to an array of two elements of type Pvec3. Array element 1 is the X
axis of pattern in MC. Array element 2 is the Y axis. The Pvec3 structure is
deefined in phigs.h as:

typedef struct {
    Pfloat delta_x; /* x magnitude */
    Pfloat delta_y; /* y magnitude */
    Pfloat delta_z; /* z magnitude */
} Pvec3;

FORTRAN Input

Parameters

RFX The x coordinate of the pattern reference point, in Modelling Coordinates (MC).

RFY The y coordinate of the pattern reference point, in MC.

RFZ The z coordinate of the pattern reference point, in MC.

RFVX The x coordinates of the two pattern reference vectors, in MC.

RFVY The y coordinates of the two pattern reference vectors, in MC.

RFVZ The z coordinates of the two pattern reference vectors, in MC.

Execution

When the SET PATTERN REFERENCE POINT AND VECTORS element is traversed, the current
pattern reference point and current pattern reference vectors entries in the PHIGS traversal
state list are set to the values contained in this element. When the interior style (either
individual or bundled, as selected by the current interior style ASF) is set to PATTERN,
these attributes are used in conjunction with the current pattern width and height vectors
for the area-defining primitives which follow in the structure network.

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INQUIRE PATTERN FACILITIES (3P)
SET PATTERN REFERENCE POINT (3P)
NAME

SET PATTERN REPRESENTATION – define pattern representation on specified workstation

SYNOPSIS

C Syntax

```c
void pset_pat_rep ( ws, index, rep )

Pint ws;
Pint index;    /* pattern bundle index */
Ppat_rep *rep; /* pattern representation pointer */
```

FORTRAN Syntax

```fortran
SUBROUTINE pspar ( WKID, PAI, DIMX, DIMY, ISC, ISR, DX, DY, COLIA )

INTEGER WKID          /* workstation identifier */
INTEGER PAI           /* pattern index */
INTEGER DIMX, DIMY    /* the dimensions of COLIA which contains the pattern */
                       /* colour index array */
INTEGER ISC, ISR      /* indices to start column, start row */
INTEGER DX, DY        /* number of columns, number of rows used */
INTEGER COLIA(DIMX, DIMY) /* pattern colour index array */
```

Required PHIGS Operating States

( PHOP, WSOP, *, * )

DESCRIPTION

Note: This function has C and FORTRAN bindings but its functionality has not yet been implemented.

C Input Parameters

- **ws**: The identifier of the workstation.
- **index**: This is an index to the workstation pattern bundle table.
- **rep**: A pointer to a Ppat_rep structure. Ppat_rep is defined as:

```c
typedef struct {
    Pint_size dims; /* pattern’s dimensions */
    Pint *colr_array; /* array of colours */
} Ppat_rep;
```

Pint_size is defined as:

```c
typedef struct {
    Pint size_x; /* dimension (number of divisions) along X */
    Pint size_y; /* dimension (number of divisions) along Y */
} Pint_size;
```

ERRORS

- 003 Ignoring function, function requires state ( PHOP, WSOP, *, * )
- 054 Ignoring function, the specified workstation is not open

modified 2 April 1993
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
112 Ignoring function, the pattern index value is less than one
103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
116 Ignoring function, one of the dimensions of pattern colour array is less than one
113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

INQUIRE PATTERN FACILITIES (3P)
INQUIRE LIST OF PATTERN INDICES (3P)
INQUIRE PATTERN REPRESENTATION (3P)
SET PATTERN REFERENCE POINT AND VECTORS (3P)
SET PATTERN SIZE (3P)
SET PATTERN REPRESENTATION PLUS (3PP)
NAME

SET PATTERN SIZE – create structure element to set current \textit{pattern size} attribute

SYNOPSIS

C Syntax

void

\texttt{pset\_pat\_size ( size\_x, size\_y )}

\texttt{P\textit{float}} \ \texttt{size\_x; \ pattern\ size \ x}

\texttt{P\textit{float}} \ \texttt{size\_y; \ pattern\ size \ y}

FORTRAN Syntax

\texttt{SUBROUTINE pspa ( SZX, SZY )}

\texttt{REAL} \ \texttt{SZX, SZY \ pattern\ size \ (MC)}

Required PHIGS Operating States

\texttt{(PHOP, *, STOP, *)}

DESCRIPTION

Purpose

SET PATTERN SIZE creates a structure element containing a value for the current pattern width vector and current pattern height vector attributes. When the interior style is \texttt{PATTERN}, these attributes apply to the following output primitives:

\begin{itemize}
  \item \texttt{FILL AREA}
  \item \texttt{FILL AREA 3}
  \item \texttt{FILL AREA SET}
  \item \texttt{FILL AREA SET 3}
  \item \texttt{SET OF FILL AREA SET 3 WITH DATA}
  \item \texttt{QUADRILATERAL MESH 3 WITH DATA}
  \item \texttt{TRIANGLE STRIP 3 WITH DATA}
\end{itemize}

\textbf{Note:} This function places the appropriate data in the Central Structure Store, and the element may be written to SunPHIGS archives, but the structure element is currently ignored during traversal.

If the current edit mode is \texttt{INSERT} when \texttt{SET PATTERN SIZE} is called, the \texttt{SET PATTERN SIZE} element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is \texttt{REPLACE}, the new element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

\begin{itemize}
  \item \texttt{size\_x} \ The \textit{x} coordinate of the \textit{pattern size}, specified in Modelling Coordinates (MC).
  \item \texttt{size\_y} \ The \textit{y} coordinate of the \textit{pattern size}, specified in Modelling Coordinates (MC).
\end{itemize}

FORTRAN Input Parameters

\begin{itemize}
  \item \texttt{SZX} \ The \textit{x} coordinate of the \textit{pattern size}, specified in MC.
  \item \texttt{SZY} \ The \textit{y} coordinate of the \textit{pattern size}, specified in MC.
\end{itemize}

Execution

When the \texttt{SET PATTERN SIZE} element is traversed, the current pattern width vector entry in the PHIGS traversal state list is set to the \textit{x} coordinate in the element’s \textit{pattern size}, and the current pattern height vector entry in the PHIGS traversal state list is set to the \textit{y} coordinate in

modified 2 April 1993
the element’s pattern size.
When the interior style (either individual or bundled, as selected by the current interior style ASF) is set to PATTERN, these attributes are used in conjunction with the current pattern reference point and vectors for the area-defining primitives which follow in the structure network.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

INQUIRE PATTERN FACILITIES (3P)
SET PATTERN REFERENCE POINT AND VECTORS (3P)
SET PATTERN REPRESENTATION (3P)
SET PICK FILTER – set PICK input device’s pick filter, which determines the pickable primitives

**SYNOPSIS**

**C Syntax**

```c
void pset_pick_filter ( ws, dev_num, filter )
```

- `ws`: workstation identifier
- `dev_num`: pick device number
- `filter`: pointer to a `Pfilter` structure containing the inclusion set and exclusion set of names.

**FORTRAN Syntax**

```fortran
SUBROUTINE pspkft ( WKID, PKDNR, ISN, IS, ESN, ES )
```

- `WKID`: workstation identifier
- `PKDNR`: pick device number
- `ISN`: number of names in the inclusion set
- `IS(ISN)`: inclusion set
- `ESN`: number of names in the exclusion set
- `ES(ESN)`: exclusion set

**Required PHIGS Operating States**

(PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

SET PICK FILTER sets the pick filter of a PICK input device, which is compared to the traversal-time current name set of each primitive to determine if the primitive is pickable by that device.

The filter contains an inclusion set and an exclusion set of names. The exclusion set has precedence over the inclusion set. During traversal, a primitive is pickable by the PICK input device if at least one name in the current name set is in the inclusion set, and no name in the current name set is in the exclusion set. Each name in the current name set, inclusion set, and exclusion set is a small positive integer.

**C Input Parameters**

- `ws`: The identifier of the workstation.
- `dev_num`: The device number of the PICK input device on this workstation whose pick filter is to be set.
- `filter`: A pointer to a `Pfilter` structure containing the inclusion set and exclusion set of names. `Pfilter` is defined in `phigs.h` as follows:

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

The `Pint_list` structure is defined in `phigs.h` as follows:
typedef struct {
    Pint num_ints; /* number of Pints in list */
    Pint *ints; /* list of integers */
} Pint_list;

**FORTRAN Input Parameters**

- **WKID**  The workstation identifier of the PICK input device workstation.
- **PKDNR**  The device number of the PICK input device whose pick filter is to be set.
- **ISN**  The number of names for the inclusion set.
- **IS**  An array containing the set of ISN names for the inclusion set.
- **ESN**  The number of names for the exclusion set.
- **ES**  An array containing the set of ESN names for the exclusion set.

**Execution**

SET PICK FILTER sets the PICK input device’s pick filter, which contains an inclusion set and an exclusion set of names, both empty by default. A workstation’s PICK input device can pick a primitive when at least one name in the current name set is in the PICK device’s pick filter inclusion set and no name in the current name set is in the device’s pick filter exclusion set. A device cannot pick any primitives when the device’s pick filter inclusion set is empty.

When the current name set is empty, no PICK device can pick subsequent primitives. The conceptual traversal of a posted structure network determines the primitive that is being picked. When it starts, the current name set is empty. During traversal, the member names specified by the ADD NAMES TO SET element are added to the current name set by the union operation on the sets. The REMOVE NAMES FROM SET elements remove names from the current name set.

**ERRORS**

- 003  Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054  Ignoring function, the specified workstation is not open
- 060  Ignoring function, specified workstation is not of category OUTIN
- 250  Ignoring function, the specified device is not available on the specified workstation

**SEE ALSO**

ADD NAMES TO SET (3P)
REMOVE NAMES FROM SET (3P)
INITIALIZE PICK (3P)
SET PICK MODE (3P)
REQUEST PICK (3P)
GET PICK (3P)
SAMPLE PICK (3P)
INQUIRE PICK DEVICE STATE (3P)

modified 2 April 1993
### NAME
SET PICK IDENTIFIER – create structure element to set current pick identifier attribute

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pspkid ( PKID )
INTEGER PKID  // pick identifier
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

### DESCRIPTION

**Purpose**

SET PICK IDENTIFIER creates a structure element containing a value for the current *pick identifier* attribute, which applies to all following output primitives in the structure network.

When the current edit mode is INSERT, a SET PICK IDENTIFIER element is inserted in the currently-open structure after the element pointed to by the current element pointer. When the edit mode is REPLACE, the new SET PICK IDENTIFIER element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

**C Input Parameter**

*pick_id*  // The *pick identifier* value.

**FORTRAN Input Parameter**

*PKID*  // *pick identifier* value.

**Execution**

The conceptual traversal of a posted structure network determines which primitive is being picked. When it starts, the current pick identifier is 0. When the SET PICK IDENTIFIER element is traversed, the current pick identifier is set to the element’s *pick identifier* value and is associated with all output primitives following in the structure network.

When a primitive is picked, the pick identifier at each level in the pick path is part of a PICK input device’s measure and can provide the application with auxiliary information about the picked primitive.

**Note:** The pick identifier is not used to determine which primitives are pickable; use ADD NAMES TO SET and SET PICK FILTER for this purpose.

### ERRORS

**005**  // Ignoring function, function requires state (PHOP, *, STOP, *)

### SEE ALSO

- INITIALIZE PICK (3P)
- SET PICK FILTER (3P)
- ESCAPE -19 (3P)
- REQUEST PICK (3P)

906 modified 2 April 1993
GET PICK (3P)
SAMPLE PICK (3P)
**NAME**  
SET PICK MODE – set pick device operating mode and echoing state

**SYNOPSIS**  
C Syntax  

```c
void pset_pick_mode ( ws, dev, mode, echo )

Pint ws;  // workstation identifier
Pint dev;  // pick device number
Pop_mode mode;  // operating mode
Pecho_switch echo;  // echo switch
```

FORTRAN Syntax  

```fortran
SUBROUTINE pspkm ( WKID, PKDNR, MODE, ESW )

INTEGER WKID  // workstation identifier
INTEGER PKDNR  // pick device number
INTEGER MODE  // operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW  // echo switch (PNECHO, PECHO)
```

**Required PHIGS Operating States**  
( PHOP, WSOP, *, *)

**DESCRIPTION**  

**Purpose**  
Use the SET PICK MODE subroutine to set the operating mode (Request, Sample, or Event) and the echo switch (Echo, No Echo) for a specified pick device on a specified workstation.

**C Input Parameters**

- **ws**  
The workstation identifier of the workstation associated with the device.

- **dev**  
The device number of the pick device to be set. See the AVAILABLE DEVICES section in INITIALIZE PICK 3 for a description of the available devices.

- **mode**  
Specifies the operating mode for the specified pick device. Pop_mode is an enumeration defined in phigs.h as follows:

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

- **echo**  
The echo switch value for the specified pick device. Pecho_switch is an enumeration defined in phigs.h as follows:

```c
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
```
FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>PKDNR</td>
<td>The device number of the pick device to be set. See the AVAILABLE DEVICES section in INITIALIZE PICK 3 for a description of the available devices.</td>
</tr>
<tr>
<td>MODE</td>
<td>The desired mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PREQU Request</td>
</tr>
<tr>
<td></td>
<td>PSAMPL Sample</td>
</tr>
<tr>
<td></td>
<td>PEVENT Event</td>
</tr>
<tr>
<td>ESW</td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PNECHO No Echo</td>
</tr>
<tr>
<td></td>
<td>PECHO Echo</td>
</tr>
</tbody>
</table>

Execution

The SET PICK MODE sets the operating mode of the specified pick device to Request, Sample, or Event, and the echo switch to Echo or No Echo. The default operating mode is Request. The default echo switch is Echo.

The operating mode controls how the input from the device is obtained.

- If the operating mode is Request, the subroutine REQUEST PICK or REQUEST PICK 3 may be used to add the specified device number to the device trigger’s list of recipients and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the REQUEST PICK subroutine returns the current input value of the device’s measure process, and the status OK. If a break occurs, the status NONE is returned.

- If the operating mode is Sample, the SAMPLE PICK or SAMPLE PICK 3 subroutine may be used to return the current input value of the device without waiting for the trigger to fire.

- If the operating mode is Event, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines AWAIT EVENT and/or GET PICK may then be used to read event reports from the queue.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process.

ERRORS

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 060 Ignoring function, specified workstation is not of category OUTIN
- 250 Ignoring function, the specified device is not available on the specified workstation

SEE ALSO

- INITIALIZE PICK (3P)
- SET PICK FILTER (3P)
- SET PICK IDENTIFIER (3P)

modified 2 April 1993
SET PICK MODE (3P)

REQUEST PICK (3P)
GET PICK (3P)
SAMPLE PICK (3P)
INQUIRE PICK DEVICE STATE (3P)
NAME
SET POLYLINE COLOUR INDEX – create structure element to set current polyline colour index attribute

SYNOPSIS
C Syntax
void pset_line_colr_ind ( colour )
    Pint colour;  polyline colour index

FORTRAN Syntax
SUBROUTINE pspclci ( COLI )
    INTEGER COLI  polyline colour index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET POLYLINE COLOUR INDEX creates a structure element containing a value for the current polyline colour index attribute.

When the current polyline colour index Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute indexes the current colour representation which applies to the following output primitives:

- POLYLINE
- POLYLINE 3
- GENERALIZED DRAWING PRIMITIVE (line primitives)
- GENERALIZED DRAWING PRIMITIVE 3 (line primitives)

C Input Parameter
colour An integer colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameter
COLI An integer colour index, which selects a colour value from the workstation’s colour table.

Execution
If the current edit mode is INSERT, a SET POLYLINE COLOUR INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET POLYLINE COLOUR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET POLYLINE COLOUR INDEX element is traversed, the current polyline colour index entry in the PHIGS traversal state list is set to the colour index.

When the current polyline colour index ASF is set to INDIVIDUAL, the polyline output primitives which follow in the structure network are drawn with the colour representation selected by the current polyline colour index from the workstation’s colour table.

modified 2 April 1993
If the colour index specified is not available on the workstation, colour index 1 will be used.

When the current polyline colour index ASF is set to BUNDLED, the polyline colour index is taken from the workstation’s representation indicated by the current polyline index. In this case, the colour index set with SET POLYLINE COLOUR INDEX has no effect.

ERRORS

005  Ignoring function, function requires state (PHOP, *, STOP, *)
113  Ignoring function, the colour index value is less than zero

SEE ALSO

SET COLOUR REPRESENTATION (3P)
SET INDIVIDUAL ASF (3P)
SET POLYLINE REPRESENTATION (3P)
SET POLYLINE COLOUR (3PP)
NAME
SET POLYLINE INDEX – create structure element containing polyline index attribute

SYNOPSIS
C Syntax

void
pset_line_ind ( index )
Pint index; polyline index

FORTRAN Syntax

SUBROUTINE pspli ( PLI )
INTEGER PLI polyline index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION

Purpose
SET POLYLINE INDEX creates a structure element containing a polyline index number, which selects an entry from the workstation’s polyline bundle table. This attribute applies to the output primitives:

POLYLINE
POLYLINE 3
GENERALIZED DRAWING PRIMITIVE (line primitives)
GENERALIZED DRAWING PRIMITIVE 3 (line primitives)

C Input Parameter
index A polyline index for a polyline representation on the workstation.

FORTRAN Input Parameter
PLI A polyline index for a polyline representation on the workstation.

Execution
If the current edit mode is INSERT, a SET POLYLINE INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET POLYLINE INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET POLYLINE INDEX element is traversed, the current polyline index value is set to the polyline index, which specifies an entry from the workstation’s polyline bundle table. The default polyline index is 1, and if the polyline index specified is not available on the workstation, 1 is used.

A polyline representation contains values for the following attributes:

- linetype
- linewidth scale factor
- polyline colour index

modified 2 April 1993
Line output primitives which follow in the structure network are drawn using the values from the specified representation for those attributes whose Aspect Source Flag (ASF) is set to BUNDLED. When the ASFs are INDIVIDUAL, the attributes come from the appropriate SET attributeP elements.

Polyline representations are defined with SET POLYLINE REPRESENTATION.

**_ERRORS**

005  Ignoring function, function requires state (PHOP, *, STOP, *)
100  Ignoring function, the bundle index value is less than one

**SEE ALSO**

SET INDIVIDUAL ASF (3P)
SET POLYLINE REPRESENTATION (3P)
SET POLYLINE REPRESENTATION PLUS (3PP)
NAME

SET POLYLINE REPRESENTATION – define polyline attribute bundle on workstation

SYNOPSIS

C Syntax

void pset_line_rep ( ws, index, rep )

Pint ws;  /* workstation identifier */
Pint index;  /* polyline bundle index */
Pline_bundle *rep;  /* polyline representation pointer */

FORTRAN Syntax

SUBROUTINE psplr ( WKID, PLI, LTYPE, LWIDTH, COLI )

INTEGER WKID /* workstation identifier */
INTEGER PLI /* polyline index */
INTEGER LTYPE /* linetype */
REAL LWIDTH /* linewidth scale factor */
INTEGER COLI /* colour index */

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET POLYLINE REPRESENTATION defines a bundle of polyline attributes for a specified entry in the workstation’s polyline bundle table. Depending on the ASF for each of the polyline attributes, the bundled attributes may apply to the following primitives:

POLYLINE
POLYLINE 3
GENERALIZED DRAWING PRIMITIVE  (line primitives)
GENERALIZED DRAWING PRIMITIVE 3 (line primitives)

C Input Parameters

ws  The identifier of the workstation for which the polyline representation is being defined.

index  The polyline index of the entry being defined.

rep  A pointer to a structure containing the attribute values defining the polyline representation, defined as follows:

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

modified 2 April 1993
rep->type is an integer value specifying a linetype; the following line types are defined:

1  PLINE_SOLID    Solid
2  PLINE_DASH     Dashed
3  PLINE_DOT      Dotted
4  PLINE_DASH_DOT Dot-dashed
0  PLINE_LONG_DASH Long-dashed
−1 PLINE_DOT_DASH_DOT Dot-dashed-dotted
−2 PLINE_CENTER   Center (long-short dashed)
−3 PLINE_PHANTOM  Phantom (long-short-short dashed)

Support for linetypes is workstation-dependent.

rep->width is the linewidth scale factor.

rep->colr_ind is the polyline colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameters

WKID  The identifier of the workstation for which the polyline representation is being defined.
PLI   The polyline index of the entry being defined.
LTYPE An integer value specifying a linetype; the following line types are defined:

1  PLSOLI    Solid
2  PLDASH    Dashed
3  PLDOT     Dotted
4  PLDASD    Dot-dashed
0  PNLONGDASH Long-dashed
−1 PNLDOTDASHDOT Dot-dashed-dot-dotted
−2 PLNCENTER Center (long-short dashed)
−3 PLNPHANTOM Phantom (long-short-short dashed)

Support for linetypes is workstation-dependent. See SET LINETYPE (3P) for caveats regarding the Center and Phantom linetypes.

LWIDTH The linewidth scale factor.

COLI  The polyline colour index, which selects a colour value from the workstation’s colour table.

Execution When SET POLYLINE REPRESENTATION is called, the polyline index entry in the table of defined polyline representations on the workstation is set to the linetype, linewidth scale factor, and polyline colour index values.

When polyline output primitives are displayed, the polyline representation specified by the current polyline index entry in the PHIGS traversal state list provides the polyline attributes for which the Aspect Source Flag (ASF) is BUNDLED. For example, when the...
current linetype ASF is set to BUNDLED, the effective linetype is the linetype attribute in the polyline representation selected by the current polyline index. The current polyline index is set by SET POLYLINE INDEX elements.

The polyline bundle table is numbered from 1.

See GENERALIZED DRAWING PRIMITIVE and GENERALIZED DRAWING PRIMITIVE 3 to determine which of the generalized primitives use the polyline attributes.

**ERRORS**

- 003 Ignoring function, function requires state (PHOP, WSOP, *, *)
- 054 Ignoring function, the specified workstation is not open
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
- 100 Ignoring function, the bundle index value is less than one
- 103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
- 104 Ignoring function, the specified line type is not available on the specified workstation
- 113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

- SET POLYLINE INDEX (3P)
- SET INDIVIDUAL ASF (3P)
- INQUIRE POLYLINE REPRESENTATION (3P)
- SET LINETYPE (3P)
- SET LINEWIDTH SCALE FACTOR (3P)
- SET POLYLINE COLOUR INDEX (3P)
- SET POLYLINE REPRESENTATION PLUS (3PP)
NAME
SET POLYMARKER COLOUR INDEX – create structure element to set current polymarker colour index attribute

SYNOPSIS
C Syntax
void
pset_marker_colr_ind ( colour )
Pint colour; polymarker colour index

FORTRAN Syntax
SUBROUTINE pspmci ( COLI )
INTEGER COLI polymarker colour index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET POLYMARKER COLOUR INDEX creates a structure element containing a value for the current polymarker colour index attribute.

When the current polymarker colour index Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute indexes the current colour representation which applies to the following output primitives:

POLYMARKER
POLYMARKER 3
GENERALIZED DRAWING PRIMITIVE -15
GENERALIZED DRAWING PRIMITIVE 3 -15

C Input Parameter
colour
An integer colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameter
COLI
An integer colour index, which selects a colour value from the workstation’s colour table.

Execution
If the current edit mode is INSERT, a SET POLYMARKER COLOUR INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET POLYMARKER COLOUR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET POLYMARKER COLOUR INDEX element is traversed, the current polymarker colour index entry in the PHIGS traversal state list is set to the colour index.

When the current polymarker colour index ASF is set to INDIVIDUAL, the polymarker output primitives which follow in the structure network are drawn with the colour representation selected by the current polymarker colour index from the workstation’s colour table.

918 modified 2 April 1993
If the colour index specified is not available on the workstation, colour index 1 will be used.

When the current polymarker colour index ASF is set to BUNDLED, the polymarker colour index is taken from the workstation’s representation indicated by the current polymarker index. In this case, the colour index set with SET POLYMARKER COLOUR INDEX has no effect.

**ERRORS**

- 005 Ignoring function, function requires state (PHOP, *, STOP, *)
- 113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

- SET COLOUR REPRESENTATION (3P)
- SET INDIVIDUAL ASF (3P)
- SET POLYMARKER REPRESENTATION (3P)
- SET POLYMARKER COLOUR (3PP)
NAME

SET POLYMARKER INDEX – create structure element containing polymarker representation index attribute

SYNOPSIS

C Syntax

void

pset_marker_ind ( index )

Pint index; polymarker index

FORTRAN Syntax

SUBROUTINE pspm ( PMI )

INTEGER PMI polymarker index

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET POLYMARKER INDEX creates a structure element containing a polymarker index number, which selects an entry from the workstation’s polymarker bundle table. This attribute applies to the output primitives:

POLYMARKER
POLYMARKER 3
GENERALIZED DRAWING PRIMITIVE -15
GENERALIZED DRAWING PRIMITIVE 3 -15

If the current edit mode is INSERT, a SET POLYMARKER INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET POLYMARKER INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

index A polymarker index for a polymarker representation on the workstation.

FORTRAN Input Parameter

PMI A polymarker index for a polymarker representation on the workstation.

Execution

When the SET POLYMARKER INDEX element is traversed, the current polymarker index value is set to the polymarker index, which specifies an entry from the workstation’s polymarker bundle table. The default polymarker index is 1, and if the polymarker index specified is not available on the workstation, 1 is used. When the ASFs are INDIVIDUAL, the attributes come from the appropriate SET attribute elements.

A polymarker representation contains values for the following attributes:

- marker type
- marker size scale factor
- polymarker colour index

modified 2 April 1993
Polymarker output primitives which follow in the structure network are drawn using the values from the specified representation for those attributes whose Aspect Source Flag (ASF) is set to BUNDLED.

Polymarker representations are defined with SET POLYMARKER REPRESENTATION.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>Ignoring function, function requires state (PHOP, *, STOP, *)</td>
</tr>
<tr>
<td>100</td>
<td>Ignoring function, the bundle index value is less than one</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET INDIVIDUAL ASF (3P)
- SET POLYMARKER REPRESENTATION (3P)
- SET POLYMARKER REPRESENTATION PLUS (3PP)
SunPHIGS Release 3.0

NAME
SET POLYMARKER REPRESENTATION – define polymarker attribute bundle on workstation

SYNOPSIS
C Syntax

```c
void pset_marker_rep ( ws, index, rep )
```

<table>
<thead>
<tr>
<th>C Input Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ws</td>
<td>The identifier of the workstation for which the polymarker representation is being defined.</td>
</tr>
<tr>
<td>index</td>
<td>The polymarker index of the entry being defined.</td>
</tr>
<tr>
<td>rep</td>
<td>A pointer to a structure containing the attribute values defining the polymarker representation, defined as follows:</td>
</tr>
</tbody>
</table>

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

FORTRAN Syntax

```fortran
SUBROUTINE psmr ( WKID, PMI, MTYPE, MSZSF, COLI )
```

<table>
<thead>
<tr>
<th>FORTRAN Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required PHIGS Operating States</td>
<td>(PHOP, WSOP, *, *)</td>
</tr>
</tbody>
</table>

DESCRIPTION
Purpose

SET POLYMARKER REPRESENTATION defines a bundle of polymarker attributes for a specified entry in the workstation’s polymarker bundle table.

Depending on the ASF for each of the polymarker attributes, the bundled attributes may apply to the following primitives:

- POLYMARKER
- POLYMARKER 3
- GENERALIZED DRAWING PRIMITIVE -15
- GENERALIZED DRAWING PRIMITIVE 3 -15

922 modified 2 April 1993
are defined:

<table>
<thead>
<tr>
<th>#</th>
<th>Marker Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PMARKER_DOT</td>
<td>Point (.)</td>
</tr>
<tr>
<td>2</td>
<td>PMARKER_PLUS</td>
<td>Plus sign (+)</td>
</tr>
<tr>
<td>3</td>
<td>PMARKER_ASTERISK</td>
<td>Asterisk (*)</td>
</tr>
<tr>
<td>4</td>
<td>PMARKER_CIRCLE</td>
<td>Circle (○)</td>
</tr>
<tr>
<td>5</td>
<td>PMARKER_CROSS</td>
<td>X-mark (×)</td>
</tr>
<tr>
<td>0</td>
<td>PMARKER_SQUARE</td>
<td>Square (□)</td>
</tr>
<tr>
<td>-1</td>
<td>PMARKER_BOWTIE_NE</td>
<td>Bowtie −1 ( □ )</td>
</tr>
<tr>
<td>-2</td>
<td>PMARKER_BOWTIE_NW</td>
<td>Bowtie −2 ( □ □ )</td>
</tr>
</tbody>
</table>

Support for marker types is workstation-dependent. The Point () marker type is always the smallest dot possible, regardless of the marker size scale factor value applicable.

`rep->size` is the marker size scale factor.

`rep->colr_ind` is the polymarker colour index, which selects a colour value from the workstation’s colour table.

### FORTRAN Input Parameters

- **WKID** The identifier of the workstation for which the polymarker representation is being defined.
- **PMI** The polymarker index of the entry being defined.
- **MTYPE** An integer value specifying a marker type; the following marker types are defined:

<table>
<thead>
<tr>
<th>#</th>
<th>Marker Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PPOINT</td>
<td>Point ()</td>
</tr>
<tr>
<td>2</td>
<td>PPLUS</td>
<td>Plus sign (+)</td>
</tr>
<tr>
<td>3</td>
<td>PAST</td>
<td>Asterisk (*)</td>
</tr>
<tr>
<td>4</td>
<td>POMARK</td>
<td>Circle (○)</td>
</tr>
<tr>
<td>5</td>
<td>PXMARK</td>
<td>X-mark (×)</td>
</tr>
<tr>
<td>0</td>
<td>PSQUARE</td>
<td>Square (□)</td>
</tr>
<tr>
<td>-1</td>
<td>PBOWTIENE</td>
<td>Bowtie −1 ( □ )</td>
</tr>
<tr>
<td>-2</td>
<td>PBOWTIENTW</td>
<td>Bowtie −2 ( □ □ )</td>
</tr>
</tbody>
</table>

Support for marker types is workstation-dependent. The Point () marker type is always the smallest dot possible, regardless of the marker size scale factor value applicable.

**MSZSF** The marker size scale factor.

**COLI** The polymarker colour index, which selects a colour value from the workstation’s colour table.

### Execution

When SET POLYMARKER REPRESENTATION is called, the polymarker index entry in the table of defined polymarker representations on the workstation is set to the marker type, marker size scale factor, and polymarker colour index values.

When polymarker output primitives are displayed, the polymarker representation specified by the current polymarker index entry in the PHIGS traversal state list provides the polymarker attributes for which the Aspect Source Flag (ASF) is BUNDLED. For

modified 2 April 1993
example, when the current marker type ASF is set to BUNDLED, the effective marker type is
the marker type attribute in the polymarker representation selected by the current
polymarker index. The current polymarker index is set by SET POLYMARKER INDEX
elements.

The polymarker bundle table is numbered from 1.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that
is, the workstation category is not OUTPUT, OUTIN, or MO)
100 Ignoring function, the bundle index value is less than one
103 Ignoring function, setting this bundle table entry would exceed the maximum
number of entries allowed in the workstation bundle table
105 Ignoring function, the specified marker type is not available on the specified
workstation
113 Ignoring function, the colour index value is less than zero

SEE ALSO

SET POLYMARKER INDEX (3P)
SET INDIVIDUAL ASF (3P)
INQUIRE POLYMARKER REPRESENTATION (3P)
SET MARKER TYPE (3P)
SET MARKER SIZE SCALE FACTOR (3P)
SET POLYMARKER COLOUR INDEX (3P)
SET POLYMARKER REPRESENTATION PLUS (3PP)
**NAME**

SET RENDERING COLOUR MODEL – create a structure element to set current rendering colour model

**SYNOPSIS**

**C Syntax**

```c
void
pset_rendering_colr_model ( colour_model )
```

```c
Pint colour_model;  colour model
```

**FORTRAN Syntax**

```fortran
SUBROUTINE psrcm ( MOD )
```

```fortran
INTEGER MOD colour rendering model
```

**Required PHIGS Operating States**

(PHOP, *, STOP, *)

**DESCRIPTION**

**Purpose**

SET RENDERING COLOUR MODEL specifies the colour model in which lighting calculation and colour interpolation are performed.

**C Input Parameters**

*colour_model*

The defined rendering colour models are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRCM_WS_DEP</td>
</tr>
<tr>
<td>1</td>
<td>PRCM_RGB</td>
</tr>
<tr>
<td>2</td>
<td>PRCM_CIE</td>
</tr>
<tr>
<td>3</td>
<td>PRCM_HSV</td>
</tr>
<tr>
<td>4</td>
<td>PRCM_HLS</td>
</tr>
</tbody>
</table>

**FORTRAN Parameters**

*MOD*

The rendering colour models are defined in phigs77.h as: PRCMWSDEP, PRCMRGB, PRCMCIE, PRCMHSV, PRCMHLS

**Execution**

Depending on the edit mode, a SET RENDERING COLOUR MODEL element is either inserted into the open structure after the element pointer or replaces the element pointed to by the element pointer. When an element of this type is interpreted, the *current rendering model* entry in the traversal state list is set to the value associated with the element. If the specific colour model is not available, then colour model 0, PRCM_WS_DEP, is used.

**ERRORS**

005 Ignoring function, function requires state (PHOP, *, STOP, *)

**SEE ALSO**

INQUIRE RENDERING COLOUR MODEL FACILITIES (3PP)

modified 2 April 1993
NAME
SET STRING MODE – set string device operating mode and echoing state

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

FORTRAN Syntax
SUBROUTINE psstm ( WKID, STDNR, MODE, ESW )
INTEGER WKID  // workstation identifier
INTEGER STDNR  // string device number
INTEGER MODE  // operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW  // echo switch (PNECHO, PECHO)

Required PHIGS Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
Use the SET STRING MODE subroutine to set the operating mode (Request, Sample, or Event) and the echo switch (Echo, No Echo) for a specified string device on a specified workstation.

C Input Parameters
ws  The workstation identifier of the workstation associated with the device.
dev  The device number of the string device to be set. See the AVAILABLE DEVICES section in INITIALIZE STRING 3 for a description of the available devices.
mode  Specifies the operating mode for the specified string device. Pop_mode is an enumeration defined in phigs.h as follows:

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

echo  The echo switch value for the specified string device. Pecho_switch is an enumeration defined in phigs.h as follows:

```c
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
```
FORTRAN Input

Parameters

<table>
<thead>
<tr>
<th>WKID</th>
<th>The workstation identifier of the workstation associated with the device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDNR</td>
<td>The device number of the string device to be set. See the AVAILABLE DEVICES section in INITIALIZE STRING 3 for a description of the available devices.</td>
</tr>
<tr>
<td>MODE</td>
<td>The desired mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td>ESW</td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
</tbody>
</table>

Execution

The SET STRING MODE sets the operating mode of the specified string device to Request, Sample, or Event, and the echo switch to Echo or No Echo. The default operating mode is Request. The default echo switch is Echo.

The operating mode controls how the input from the device is obtained.

- If the operating mode is Request, the subroutine REQUEST STRING, or REQUEST STRING 3 may be used to add the specified device number to the device trigger’s list of recipients and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the REQUEST STRING subroutine returns the current input value of the device’s measure process, and the status OK. If a break occurs, the status NONE is returned.

- The measure of a STRING device is a character string entered at the keyboard, and the trigger is a carriage return.

- If the operating mode is Sample, the SAMPLE STRING or SAMPLE STRING 3 subroutine may be used to return the current input value of the device without waiting for the trigger to fire. The current input value is the current contents of the input buffer.

- If the operating mode is Event, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines AWAIT EVENT and/or GET STRING may then be used to read event reports from the queue. The event report for a STRING device includes a device identifier and a character string.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process. The string device echo displays the currently-input character string in the echo area or echo volume for the device. The echo area is a 2D area specified with INITIALIZE STRING. An echo volume is a 3D display volume specified by INITIALIZE STRING 3.
Errors

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
061  Ignoring function, category of the specified workstation is not INPUT or OUTIN
250  Ignoring function, the specified device is not available on the specified workstation

See Also

INITIALIZE STRING (3P)
REQUEST STRING (3P)
GET STRING (3P)
SAMPLE STRING (3P)
INQUIRE STRING DEVICE STATE (3P)
NAME
SET STROKE MODE – set stroke device operating mode and echoing state

SYNOPSIS
C Syntax

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

FORTRAN Syntax

SUBROUTINE psskm ( WKID, SKDNR, MODE, ESW )
INTEGER WKID /* workstation identifier */
INTEGER SKDNR /* stroke device number */
INTEGER MODE /* operating mode (PREQU, PSAMPL, PEVENT) */
INTEGER ESW /* echo switch (PNECHO, PECHO) */

DESCRIPTION
Purpose
Use the SET STROKE MODE subroutine to set the operating mode (Request, Sample, or Event) and the echo switch (Echo or No Echo) for a specified stroke device on a specified workstation.

C Input Parameters

ws The workstation identifier of the workstation associated with the device.

dev The device number of the stroke device to be set. See the AVAILABLE DEVICES section in INITIALIZE STROKE 3 for a description of the available devices.

mode Specifies the operating mode for the specified stroke device. Pop_mode is an enumeration defined in phigs.h as follows:

typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
} Pop_mode;

echo The echo switch value for the specified stroke device. Pecho_switch is an enumeration defined in phigs.h as follows:

typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;

modified 2 April 1993
### FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WKID</strong></td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td><strong>SKDNR</strong></td>
<td>The device number of the stroke device to be set. See the <strong>AVAILABLE DEVICES</strong> section in INITIALIZE STROKE 3 for a description of the available devices.</td>
</tr>
<tr>
<td><strong>MODE</strong></td>
<td>The desired mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td>PREQU</td>
<td>Request</td>
</tr>
<tr>
<td>PSAMPL</td>
<td>Sample</td>
</tr>
<tr>
<td>PEVENT</td>
<td>Event</td>
</tr>
<tr>
<td><strong>ESW</strong></td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td>PNECHO</td>
<td>No echo</td>
</tr>
<tr>
<td>PECHO</td>
<td>Echo</td>
</tr>
</tbody>
</table>

### Execution

The `SET STROKE MODE` sets the operating mode of the specified stroke device to Request, Sample, or Event, and the echo switch to Echo or No Echo. The default operating mode is Request. The default echo switch is Echo.

The operating mode controls how the input from the device is obtained.

- If the operating mode is Request, the subroutine `REQUEST STROKE`, or `REQUEST STROKE 3` may be used to add the specified device number to the device trigger’s list of recipients and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the `REQUEST STROKE` subroutine returns the input value (the current coordinates of the device) and the status OK. If a break occurs, the status NONE is returned.
- If the operating mode is Sample, the `SAMPLE STROKE` or `SAMPLE STROKE 3` subroutine may be used to return the current input value of the device without waiting for the trigger to fire.
- If the operating mode is Event, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines `AWAIT EVENT` and/or `GET STROKE` may then be used to read event reports from the queue.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process. The stroke echo updates the cursor position.

### ERRORS

- **003** Ignoring function, function requires state (PHOP, WSOP, *, *)
- **054** Ignoring function, the specified workstation is not open
- **061** Ignoring function, specified workstation is not of category INPUT or OUTIN
- **250** Ignoring function, the specified device is not available on the specified workstation

---

modified 2 April 1993
SEE ALSO

INITIALIZE STROKE (3P)
INITIALIZE STROKE 3 (3P)
REQUEST STROKE 3 (3P)
GET STROKE 3 (3P)
SAMPLE STROKE 3 (3P)
INQUIRE STROKE DEVICE STATE 3 (3P)
NAME

SET TEXT ALIGNMENT – create structure element to set current text alignment attribute

SYNOPSIS

C Syntax

```c
void pset_text_align ( text_align )
Ptext_align *text_align; text alignment
```

FORTRAN Syntax

```fortran
SUBROUTINE pstxal ( TXALH, TXALV )
INTEGER TXALH text alignment horizontal
INTEGER TXALV text alignment vertical
(PHOP, *, STOP, *)
```

DESCRIPTION

Purpose

SET TEXT ALIGNMENT creates a structure element containing a value for the current text alignment attribute, which positions the text string in relation to the text position. This attribute applies to the TEXT, TEXT 3, GDP -17, and GDP 3 -17 output primitives:

If the current edit mode is INSERT, a SET TEXT ALIGNMENT element is inserted into the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET TEXT ALIGNMENT element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameters

`text_align`

A pointer to the text alignment structure, defined as follows:

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

`text_align->hor` is the horizontal component. This is an enumerated value, and may be one of:

- PHOR_NORM Normal
- PHOR_LEFT Left
- PHOR_CTR Center
- PHOR_RIGHT Right

`text_align-> vert` is the vertical component. This is an enumerated value, and may be one of:

- PVERT_NORM Normal
- PVERT_TOP Top
- PVERT_CAP Cap
- PVERT_HALF Half
- PVERT_BASE Base

modified 2 April 1993
FORTRAN Input

Parameters

**TXALH** The text alignment horizontal component. This is an enumerated value, and may be one of:

- PAHNOR Normal
- PALEFT Left
- PACENT Center
- PARITE Right

**TXALV** The text alignment vertical component. This is an enumerated value, and may be one of:

- PAVNOR Normal
- PATOP Top
- PACAP Cap
- PAHALF Half
- PABASE Base
- PABOTT Bottom

Execution

When the SET TEXT ALIGNMENT element is traversed, the current text alignment entry in the PHIGS traversal state list is set to text alignment. This attribute is used to position text strings from text output primitives which follow in the structure network, in relation to the text position provided with each text output primitive. The horizontal component has four values; the vertical component, six. The two components of the alignment can be considered individually.

Imagine first rendering the text string using all other text attributes, and then moving the entire text string to place the text extent parallelogram that surrounds the character bodies in the correct position in relation to the text position. (The size and shape of the text is entirely specified by the other attributes.) This movement is oriented by the character up and base vectors; consider the direction of the character up vector to be vertical, and that of the character base vector to be horizontal.

The horizontal alignment of Left or Right requires the corresponding side of the parallelogram to pass through the text position. The horizontal alignment of Center causes the text position to lie midway between the left and right sides of the parallelogram.

The vertical alignment corresponds to one of the five horizontal lines through the definition of a character. (These lines are in the same location for every character in a single font.) The vertical alignment of Top or Bottom requires the corresponding side of the parallelogram to pass through the text position. The vertical alignment of Cap causes the text position to lie on the capline of the string (when the text path is Left or Right), or on the capline of the topmost character in the string (text path is Up or Down). The vertical alignment of Base causes the text position to lie on the baseline of the entire string (when the text path is Left or Right) or on the baseline of the bottom character in the string (text path is Up or Down). The vertical alignment of Half causes the text position to lie on the halfline of the entire string (when the text path is Left or Right) or on a line midway between the capline and baseline.

modified 2 April 1993
between the halflines of the top and bottom characters (text path is Up or Down).

The Normal value of either text alignment component causes an effect equivalent to one of the other values of the same component. PHIGS defines which other value is used to be the natural alignment for the text path value used:

<table>
<thead>
<tr>
<th>Text Path Value</th>
<th>Equivalent (Horizontal, Vertical) Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>(Left, Base)</td>
</tr>
<tr>
<td>LEFT</td>
<td>(Right, Base)</td>
</tr>
<tr>
<td>UP</td>
<td>(Center, Base)</td>
</tr>
<tr>
<td>DOWN</td>
<td>(Center, Top)</td>
</tr>
</tbody>
</table>

The default text alignment is (Normal, Normal); the default text path is Right.

Example

(Center, Top) text alignment might be used with text path Right to center a chart’s x-axis label under the x-axis, without calculating the combined size of the characters in the string. (Right, Center) might be used with text path Down to center the right edge of a chart’s y-axis label along the y-axis. Each character faces normally, but the characters in the string would proceed down the display, to the left of the y-axis.

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET ANNOTATION TEXT ALIGNMENT (3P)
SET TEXT PATH (3P)
NAME  SET TEXT COLOUR INDEX – create structure element to set current text colour index attribute

SYNOPSIS
C Syntax
void
pset_text_colr_ind ( colour )

Pint colour; text colour index

FORTRAN Syntax
SUBROUTINE pstxci ( COLI )

INTEGER COLI text colour index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET TEXT COLOUR INDEX creates a structure element containing a value for the current text colour index attribute.

When the current text colour index Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute indexes the current colour representation which applies to the output primitives:

- TEXT
- TEXT 3
- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3
- GENERALIZED DRAWING PRIMITIVE -17
- GENERALIZED DRAWING PRIMITIVE 3 -17
- GENERALIZED DRAWING PRIMITIVE -18
- GENERALIZED DRAWING PRIMITIVE 3 -18

C Input Parameter
colour An integer colour index, which selects a colour value from the workstation’s colour table.

FORTRAN Input Parameter
COLI An integer colour index, which selects a colour value from the workstation’s colour table.

Execution
If the current edit mode is INSERT, a SET TEXT COLOUR INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET TEXT COLOUR INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

When the SET TEXT COLOUR INDEX element is traversed, the current text colour index entry in the PHIGS traversal state list is set to the colour index.
When the current text colour index ASF is set to INDIVIDUAL, the text output primitives which follow in the structure network are drawn with the colour representation selected by the current text colour index from the workstation’s colour table.

If the colour index specified is not available on the workstation, colour index 1 will be used.

When the current text colour index ASF is set to BUNDLED, the text colour index is taken from the workstation’s representation indicated by the current text index. In this case, the colour index set with SET TEXT COLOUR INDEX has no effect.

**ERRORS**

- 005 Ignoring function, function requires state (PHOP, *, STOP, *)
- 113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

- **SET COLOUR REPRESENTATION (3P)**
- **SET INDIVIDUAL ASF (3P)**
- **SET TEXT COLOUR (3PP)**
- **SET TEXT REPRESENTATION (3P)**
NAME

SET TEXT FONT – create structure element to set current text font attribute

SYNOPSIS

C Syntax

void pset_text_font ( font )

Pint font;  text font

FORTRAN Syntax

SUBROUTINE pstxfn ( FONT )

INTEGER FONT  text font

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET TEXT FONT creates a structure element containing a value for the current text font attribute, which selects a font from those available on the workstation.

When the text font ASF is set to INDIVIDUAL, this attribute applies to the output primitives:

   TEXT
   TEXT 3
   ANNOTATION TEXT RELATIVE
   ANNOTATION TEXT RELATIVE 3
   GENERALIZED DRAWING PRIMITIVE -17
   GENERALIZED DRAWING PRIMITIVE 3 -17
   GENERALIZED DRAWING PRIMITIVE -18
   GENERALIZED DRAWING PRIMITIVE 3 -18

If the current edit mode is INSERT, then a SET TEXT FONT element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET TEXT FONT element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameters

   font  The text font, specified as an index to the workstation’s non-writable table of available fonts.

   All the font indices have named constants defined in phigs.h. See INTRO INTERNATIONALIZATION (7P) for a list of valid font indices.

FORTRAN Input Parameters

   FONT  The text font, specified as an index to the workstation’s non-writable table of available fonts. The fonts available depend on the character set used.

   All the font indices have named constants defined in phigs77.h. See INTRO INTERNATIONALIZATION (7P) for a list of valid font indices.

modified 2 April 1993
When the `SET TEXT FONT` element is traversed, the current text font entry in the PHIGS traversal state list is set to `text font`. When the current text font ASF (Aspect Source Flag) is set to `INDIVIDUAL`, the `text font` is applied to text strings in output primitives that follow in the structure network.

The text font attribute selects a font from those available on the workstation. The default `text font` is 1; the default `text precision` is String Precision. The default is used when the font selected is not available in the precision selected on the workstation.

When the text font ASF is set to `BUNDLED`, the text font is taken from the workstation’s representation indicated by the `current text index`. In this case, the font set with `SET TEXT FONT` has no effect.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>Ignoring function, function requires state (<code>PHOP</code>, <code>*</code>, <code>STOP</code>, <code>*</code>)</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- `INQUIRE TEXT FACILITIES (3P)`
- `SET INDIVIDUAL ASF (3P)`
- `SET TEXT PRECISION (3P)`
- `SET TEXT REPRESENTATION (3P)`
- `INTRO INTERNATIONALIZATION (7P)`
NAME

SET TEXT INDEX – create structure element containing text representation index attribute

SYNOPSIS

C Syntax

void

pset_text_ind ( index )

Pint index;  text index

FORTRAN Syntax

SUBROUTINE pstxi ( TXI )

INTEGER TXI  text index

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET TEXT INDEX creates a structure element containing a text index number, which selects an entry from the workstation’s text bundle table. This attribute applies to the output primitives:

- TEXT
- TEXT 3
- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3
- GENERALIZED DRAWING PRIMITIVE -17
- GENERALIZED DRAWING PRIMITIVE 3 -17
- GENERALIZED DRAWING PRIMITIVE -18
- GENERALIZED DRAWING PRIMITIVE 3 -18

If the current edit mode is INSERT, a SET TEXT INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET TEXT INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter

index  A text index for a text representation on the workstation.

FORTRAN Input Parameter

TXI  A text index for a text representation on the workstation.

Execution

When the SET TEXT INDEX element is traversed, the current text index value is set to the text index, which specifies an entry from the workstation’s text bundle table. The default text index is 1, and if the text index specified is not available on the workstation, 1 is used.

A text representation contains values for the following attributes:

- text font
- text precision
- character expansion factor
- character spacing
- text colour index

Text output primitives which follow in the structure network are drawn using the values from the specified representation for those attributes whose Aspect Source Flag (ASF) is set to BUNDLED. When the ASFs are INDIVIDUAL, the attributes come from the appropriate SET attribute elements.

Text representations are defined with SET TEXT REPRESENTATION.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)
100 Ignoring function, the bundle index value is less than one

SEE ALSO

SET TEXT REPRESENTATION PLUS (3PP)
SET INDIVIDUAL ASF (3P)
SET TEXT REPRESENTATION (3P)
NAME
SET TEXT PATH – create structure element to set current text path attribute

SYNOPSIS
C Syntax
void
pset_text_path ( path )
Ptex_path path; text path

FORTRAN Syntax
SUBROUTINE pstxp ( TXP )
INTEGER TXP text path (PRIGHT, PLEFT, PUP, PDOWN)

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET TEXT PATH creates a structure element containing a value for the text path attribute, which controls the direction in which the string is written relative to the character up and base vectors. This attribute applies to the following output primitives

TEXT
TEXT 3
GENERALIZED DRAWING PRIMITIVE -18
GENERALIZED DRAWING PRIMITIVE 3 -18

that follow in the structure network.

If the current edit mode is INSERT, then a SET TEXT PATH element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET TEXT PATH element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
path The text path is one of the following enumerated values:
PPATH_RIGHT Right
PPATH_LEFT Left
PPATH_UP Up
PPATH_DOWN Down

FORTRAN Input Parameter
TXP The text path is one of the following enumerated values:
PRIGHT Right
PLEFT Left
PUP Up
PDOWN Down

Execution
When the SET TEXT PATH element is traversed, the current text path entry in the PHIGS traversal state list is set to text path. The text path determines the direction of displacement between one character and the next in a string, defined in relation to the

modified 2 April 1993
character base and up vectors.

If text path is set to Right, then the text string is written along a baseline in the direction of the CHARACTER BASE VECTOR. If text path is set to Left, then the text string is written along a baseline in the direction opposite to the CHARACTER BASE VECTOR. If text path is set to Up, then the text string is written in the direction of the CHARACTER UP VECTOR. If text path is set to Down, the text string is written in the direction opposite to the CHARACTER UP VECTOR.

Note: text path controls only the direction in which the string is written. The position of the string in relation to the text position point is controlled by the text alignment.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET CHARACTER UP VECTOR (3P)
SET TEXT ALIGNMENT (3P)
NAME
SET TEXT PRECISION – create structure element to set current text precision attribute

SYNOPSIS
C Syntax
void
pset_text_prec ( precision )
Ptext_prec precision;

FORTRAN Syntax
SUBROUTINE ptxpr ( PREC )
INTEGER PREC text precision ( PSTRP, PCHARP, PSTRKP )

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET TEXT PRECISION creates a structure element containing a value for the current text precision attribute, which controls how closely the text must be drawn in relation to the font definition and the applicable text attributes.

When the current text precision Aspect Source Flag (ASF) is set to INDIVIDUAL, this attribute defines the text precision applied to the output primitives:

TEXT
TEXT 3
ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3
GENERALIZED DRAWING PRIMITIVE -17
GENERALIZED DRAWING PRIMITIVE 3 -17
GENERALIZED DRAWING PRIMITIVE -18
GENERALIZED DRAWING PRIMITIVE 3 -18

C Input Parameter
precision
The text precision is an enumerated type with one of the following values:

PPREC_STRING String precision
PPREC_CHAR Character precision
PPREC_STROKE Stroke precision

FORTRAN Input Parameter
PRECP The text precision is one of the following values:

PSTRP String precision
PCHARP Character precision
PSTRKP Stroke precision

Execution
If the current edit mode is INSERT, then a SET TEXT PRECISION element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, then the new SET TEXT PRECISION element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point

modified 2 April 1993
When the \texttt{SET TEXT PRECISION} element is traversed, the current text precision entry in the PHIGS traversal state list is set to text precision.

When the current text precision \texttt{ASF} is set to \texttt{INDIVIDUAL}, the text output primitives which follow in the structure network are then drawn with the precision specified.

The text precision attribute allows the tradeoff of quality and performance for text primitives.

\textit{Stroke} \quad Stroke precision demands adherence to all text attributes. Clipping must be performed on portions of characters, but need not involve vector strokes. Stroke precision provides the highest quality available.

\textit{Character} \quad Character precision allows the implementation to clip on a character by character basis. If any portion of a character is outside the clipping limits, then the entire character can be clipped.

\textit{String} \quad String precision allows the implementation to ignore the character base and up vectors, text path, text alignment, and character spacing, and permits clipping in an implementation-dependent way.

SunPHIGS presently adheres to all text attributes, regardless of the text precision. SunPHIGS fonts are always stroke fonts, not raster fonts.

The default text font is 1. The default text precision is \texttt{String} precision. The default is used when the font selected is not available in the selected precision on the workstation.

When the current text precision \texttt{ASF} is set to \texttt{BUNDLED}, the text precision is taken from the workstation's representation indicated by the current text index. In this case, the text precision set with \texttt{SET TEXT PRECISION} has no effect.

\section*{ERRORS}

005 \quad Ignoring function, function requires state (\texttt{PHOP, *, STOP, *})

\section*{SEE ALSO}

- \texttt{SET INDIVIDUAL ASF (3P)}
- \texttt{SET TEXT FONT (3P)}
- \texttt{SET TEXT REPRESENTATION (3P)}
NAME

SET TEXT REPRESENTATION – define text attribute bundle on workstation

SYNOPSIS

**C Syntax**

```c
void pset_text_rep ( ws, index, rep )
Pint ws;    \*workstation identifier
Pint index; \*text bundle index
Ptext_bundle *rep; \*text representation pointer
```

**FORTRAN Syntax**

```fortran
SUBROUTINE pstxr ( WKID, TXI, FONT, PREC, CHXP, CHSP, COLI )
INTEGER WKID \*workstation identifier
INTEGER TXI \*text index
INTEGER FONT \*text font
INTEGER PREC \*text precision (PSTRP, PCHARP, PSTRKP)
REAL CHXP \*character expansion factor
REAL CHSP \*character spacing
INTEGER COLI \*text colour index
```

**Required PHIGS Operating States**

(HP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

SET TEXT REPRESENTATION defines a bundle of text attributes for a specified entry in the workstation’s text bundle table. Depending on the ASF for each of the text attributes, the bundled attributes may apply to the following primitives:

- TEXT
- TEXT 3
- ANNOTATION TEXT RELATIVE
- ANNOTATION TEXT RELATIVE 3
- GENERALIZED DRAWING PRIMITIVE -17
- GENERALIZED DRAWING PRIMITIVE 3 -17
- GENERALIZED DRAWING PRIMITIVE -18
- GENERALIZED DRAWING PRIMITIVE 3 -18

**C Input Parameters**

- `ws` The identifier of the workstation for which the text representation is being defined.
- `index` The text index of the entry being defined.
- `rep` A pointer to a structure containing the attribute values defining the text representation, defined as follows:

```c
typedef struct {
    Pint font;    \/* text font */
    PtextPrec prec; \/* text precision */
    Pfloat char_expan; \/* character char_expansion factor */
} text_representation;
```

modified 2 April 1993
Pfloat char_space; /* character spacing */
Pint colr_ind; /* text colour index */

} Ptext_bundle;

rep->font is a text font index.
rep->prec is a text precision enumeration value, one of:
PPREC_STRING String precision
PPREC_CHAR Character precision
PPREC_STROKE Stroke precision

rep->char_expan is the character expansion factor value.
rep->char_space is the character spacing factor value.
rep->colr_ind is the text colour, which selects a colour value from the workstation's colour table.

FORTRAN Input Parameters

WKID The identifier of the workstation for which the text representation is being defined.

TXI The text index of the entry being defined.

FONT The text font index, possibly workstation-dependent.

PREC The text precision enumeration value, one of:
PSTRP String precision
PCHARP Character precision
PSTRKP Stroke precision

CHXP The character expansion factor value.

CHSP The character spacing factor value.

COLI The text colour index that selects a colour value from the workstation's colour table.

Execution

When set text representation is called, the text index entry in the table of defined text representations on the workstation is set to the text font index, text precision, character expansion factor, character spacing factor, and text colour index values.

When text output primitives are displayed, the text representation specified by the current text index entry in the PHIGS traversal state list provides the text attributes for which the Aspect Source Flag (ASF) is BUNDLED. For example, when the current text font ASF is set to BUNDLED, the effective text font is the text font attribute in the text representation selected by the current text index. The current text index is set by set text index structure elements.

The text precision specifies the level of precision used to draw the text characters in relation to the font definition and applicable text attributes.
The character expansion factor is used to scale the standard width of the characters established by the width-to-height ratio specified by the font design. An expansion factor of less than one produces narrower characters; an expansion factor of greater than one produces wider characters, relative to their height.

The character spacing factor is specified as a fraction of the current character height. A positive value inserts additional space between adjacent character bodies in the text string; a negative value causes the adjacent character bodies to overlap. The default value is 0.0, so the character bodies are placed adjacent to one another with no additional space beyond that already in the font’s design.

The text bundle table is numbered from 1.

**ERRORS**

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100 Ignoring function, the bundle index value is less than one
103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
106 Ignoring function, the specified font is not available for the requested text precision on the specified workstation
113 Ignoring function, the colour index value is less than zero

**SEE ALSO**

INQUIRE TEXT REPRESENTATION (3P)
SET TEXT COLOUR INDEX (3P)
SET CHARACTER EXPANSION FACTOR (3P)
SET CHARACTER SPACING (3P)
SET TEXT REPRESENTATION PLUS (3PP)
SET INDIVIDUAL ASF (3P)
SET TEXT FONT (3P)
SET TEXT PRECISION (3P)

modified 2 April 1993
NAME

SET VALUATOR MODE – set valuator device operating mode and echoing state

SYNOPSIS

C Syntax

```c
void pset_val_mode ( ws, dev, mode, echo )
```

```c
Pint ws;  // workstation identifier
Pint dev; // valuator device number
Pop_mode mode; // operating mode
Pecho_switch echo; // echo switch
```

FORTRAN Syntax

```fortran
SUBROUTINE psvlm ( WKID, VLDNR, MODE, ESW )
```

```fortran
INTEGER WKID  // workstation identifier
INTEGER VLDNR // valuator device number
INTEGER MODE  // operating mode (PREQU, PSAMPL, PEVENT)
INTEGER ESW   // echo switch (PNECHO, PECHO)
```

Required PHIGS

Operating States

( PHOP, WSOP, *, * )

DESCRIPTION

Purpose

Use the SET VALUATOR MODE subroutine to set the operating mode (Request, Sample, or Event) and the echo switch (Echo, No Echo) for a specified valuator device on a specified workstation.

C Input Parameters

- `ws` The workstation identifier of the workstation associated with the device.
- `dev` The device number of the valuator device to be set. See the AVAILABLE DEVICES section in INITIALIZE VALUATOR 3 for a description of the available devices.
- `mode` Specifies the operating mode for the specified valuator device. Pop_mode is an enumeration defined in phigs.h as follows:

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

- `echo` The echo switch value for the specified valuator device. Pecho_switch is an enumeration defined in phigs.h as follows:

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

948 modified 2 April 1993
FORTRAN Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The workstation identifier of the workstation associated with the device.</td>
</tr>
<tr>
<td>VLDNR</td>
<td>The device number of the valuator device to be set. See the AVAILABLE DEVICES section in INITIALIZE VALUATOR 3 for a description of the available devices.</td>
</tr>
<tr>
<td>MODE</td>
<td>The desired mode of the device. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PREQU Request</td>
</tr>
<tr>
<td></td>
<td>PSAMPL Sample</td>
</tr>
<tr>
<td></td>
<td>PEVENT Event</td>
</tr>
<tr>
<td>ESW</td>
<td>The echo flag. Valid values as defined in phigs77.h are:</td>
</tr>
<tr>
<td></td>
<td>PNECHO No echo</td>
</tr>
<tr>
<td></td>
<td>PECHO Echo</td>
</tr>
</tbody>
</table>

Execution

The SET VALUATOR MODE sets the operating mode of the specified valuator device to Request, Sample, or Event, and the echo switch to Echo or No Echo. The default operating mode is Request. The default echo switch is Echo.

The operating mode controls how the input from the device is obtained.

- If the operating mode is Request, the subroutine REQUEST VALUATOR, or REQUEST VALUATOR 3 may be used to add the specified device number to the device trigger's list of recipients and suspend PHIGS until the trigger fires or the operator executes a break. If the trigger fires, the REQUEST VALUATOR subroutine returns the current input value and the status OK. If a break occurs, the status NONE is returned.

- If the operating mode is Sample, the SAMPLE VALUATOR or SAMPLE VALUATOR 3 subroutine may be used to return the current input value of the device without waiting for the trigger to fire.

- If the operating mode is Event, the input values generated by the device when its trigger fires are added as event reports to the event queue. The subroutines AWAITS EVENT and/or GET VALUATOR may then be used to read event reports from the queue.

The echo switch controls whether the echoing specified by the prompt/echo type for this device is performed as part of the measure process.

ERRORS

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>061</td>
<td>Ignoring function, specified workstation is not of category INPUT or of category OUTIN</td>
</tr>
<tr>
<td>250</td>
<td>Ignoring function, the specified device is not available on the specified workstation</td>
</tr>
</tbody>
</table>

modified 2 April 1993
<table>
<thead>
<tr>
<th>SEE ALSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE VALUATOR (3P)</td>
</tr>
<tr>
<td>REQUEST VALUATOR (3P)</td>
</tr>
<tr>
<td>GET VALUATOR (3P)</td>
</tr>
<tr>
<td>SAMPLE VALUATOR (3P)</td>
</tr>
<tr>
<td>INQUIRE VALUATOR DEVICE STATE (3P)</td>
</tr>
<tr>
<td>See Event Input Mode</td>
</tr>
<tr>
<td>See Request Input Mode</td>
</tr>
<tr>
<td>See Sample Input Mode</td>
</tr>
</tbody>
</table>
NAME
SET VIEW INDEX – create structure element containing view index attribute

SYNOPSIS
C Syntax
void
pset_view_ind ( index )
  Pint index;  view index

FORTRAN Syntax
SUBROUTINE psvwi ( VIEWI )
  INTEGER VIEWI  view index

Required PHIGS Operating States
(PHOP, *, STOP, *)

DESCRIPTION
Purpose
SET VIEW INDEX creates a structure element containing a view index number, which selects an entry from the workstation’s view table. The view representation selected maps primitives from World Coordinates to Normalized Projection Coordinates. This attribute applies to all output primitives.

If the current edit mode is INSERT, a SET VIEW INDEX element is inserted in the currently-open structure after the element pointed to by the current element pointer. If the edit mode is REPLACE, the new SET VIEW INDEX element replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the new element.

C Input Parameter
index  A view index for a view representation on the workstation.

FORTRAN Input Parameter
VIEWI  A view index for a view representation on the workstation.

Execution
When the SET VIEW INDEX element is traversed, the current view index value is set to the view index, which specifies an entry from the workstation’s view table. The default view index is 0, and if the view index specified is not available on the workstation, view index 0 is used.

The view representation maps output primitives from World Coordinates to Normalized Projection Coordinates by:

- Applying a view orientation matrix to transform the primitives to the Viewing Reference Coordinate system (VRC).
- Applying a view mapping matrix to transform the primitives to the Normalized Projection Coordinate system (NPC).
- Optionally clipping the primitives to specified clipping limits in NPC according to separate clipping indicators for the x, y, and z dimensions.

modified 2 April 1993
A view representation is set using SET VIEW REPRESENTATION or SET VIEW REPRESENTATION 3. See the descriptions of these functions for more information on the representations and their default values.

**ERRORS**

- 005  Ignoring function, function requires state (PHOP, *, STOP, *)
- 114  Ignoring function, the view index value is less than zero

**SEE ALSO**

- SET VIEW REPRESENTATION (3P)
- SET VIEW REPRESENTATION 3 (3P)
NAME
SET VIEW REPRESENTATION – define 2D view representation entry on workstation

SYNOPSIS
C Syntax

void

pset_view_rep ( ws, index, rep )

Pint ws;  workstation identifier
Pint index;  view index
Pview_rep *rep;  view representation

FORTRAN Syntax

SUBROUTINE psvwr ( WKID, VIEWI, VWORMT, VWMPMT, VWCPLM, XYCLPI )

INTEGER WKID  workstation identifier
INTEGER VIEWI  view index
REAL VWORMT(3, 3)  view orientation matrix
REAL VWMPMT(3, 3)  view mapping matrix
REAL VWCPLM(4)  view clipping limits (NPC) (XMIN, XMAX, YMIN, YMAX)
INTEGER XYCLPI  x-y clipping indicator (PNCLIP, PCLIP)

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION
Purpose

SET VIEW REPRESENTATION defines a view representation entry in the workstation’s view table, using a two-dimensional model. A view representation defines how the composited objects are to be represented in relation to the viewer.

A view representation controls the viewing stage of the transformation pipeline, which transforms coordinates in World Coordinates (WC) to Normalized Projection Coordinates (NPC), and optionally clips to the limits of NPC space.

C Input Parameters

ws  The identifier of the workstation for which the view representation is being defined.

index  The view index of the entry being defined.

rep  A pointer to a Pview_rep structure containing the values in the view representation, defined in phigs.h as follows:

typedef struct {
    Pmatrix ori_matrix;  /* view orientation matrix */
    Pmatrix map_matrix;  /* view mapping matrix */
    Plimit clip_limit;  /* clipping limits */
    Pclip_ind xy_clip;  /* x-y clipping indicator */
} Pview_rep;

Pmatrix is defined in phigs.h as follows:

typedef Pfloat Pmatrix[3][3];

modified 2 April 1993 953
Plimit is defined in phigs.h as follows:

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

Pclip_ind is defined in phigs.h as follows:
typedef enum {
    PIND_NO_CLIP,
    PIND_CLIP
} Pclip_ind;

The ori_matrix is the 3×3 homogeneous view orientation matrix, probably from EVALUATE VIEW ORIENTATION MATRIX.

The map_matrix is the 3×3 homogeneous view mapping matrix, probably from EVALUATE VIEW MAPPING MATRIX.

The clip_limit contains the 4 Pfloat clipping limits: x_min, x_max, y_min, and y_max. The xy_clip contains the x-y clip indicator for the x and y axes.

**FORTRAN Input Parameters**

**WKID** The identifier of the workstation for which the view representation is being defined.

**VIEWI** The view index of the entry being defined.

**VWORMT** An array containing the 3×3 homogeneous view orientation matrix, probably from EVALUATE VIEW ORIENTATION MATRIX.

**VWMPMT** An array containing the 3×3 homogeneous view mapping matrix, probably from EVALUATE VIEW MAPPING MATRIX.

**VWCPLM** An array containing the 4 REAL clipping limits: XMIN, XMAX, YMIN, and YMAX.

**XYCLPI** The x-y clipping indicator for the x and y axes.

Valid clipping indicators as defined in phigs77.h are one of the enumeration values:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PNCLIP No clipping</td>
</tr>
<tr>
<td>1</td>
<td>PCLIP Clipping</td>
</tr>
</tbody>
</table>
Execution

SET VIEW REPRESENTATION defines a view representation entry in the workstation’s view table using a two-dimensional model.

The current view index attribute applied to each output primitive at traversal time determines which view representation is used to map the primitive from WC space to NPC space, typically with a parallel or perspective projection.

This viewing process for each output primitive is accomplished in three stages:

1. The view orientation matrix is applied to orient View Reference Coordinate (VRC) space with respect to WC space. Use the utility function EVALUATE VIEW ORIENTATION MATRIX to generate this matrix from a view reference point and view up vector specified in WC. The view reference point becomes the origin of VRC and the view up vector orients the VRC v axis.

2. The view mapping matrix is applied to map the VRC system to the NPC system. Use the utility function EVALUATE VIEW MAPPING MATRIX to generate this matrix. The utility accepts a specified view area called a window in VRC space to be mapped onto a specified projection viewport in NPC space. The projection is established in relation to the projection reference point, also specified in VRC space. The projection may be perspective or parallel, oblique or non-oblique.

3. Clipping limits are optionally applied. The view clipping limits define an area of NPC. When a clipping indicator is set to Clipping, any portion of an object beyond that limit is not displayed.

Note: These clipping limits are typically identical to the projection viewport provided to EVALUATE VIEW MAPPING MATRIX. If the clipping limits differ from the projection viewport, then the object portions mapped by the view mapping matrix to be within the clipping limits in NPC are displayed.

In any case, objects are subject to workstation clipping, which may not be disabled. The maximum volume of NPC space that may be displayed is from 0 to 1 in all dimensions.

The immediate visual effect of changing a view representation depends on the workstation’s display update state. When SET VIEW REPRESENTATION is called, the display update state may prohibit immediate updating (that is, regeneration) of the workstation’s display. In this case, the view index view table entry’s viewing transformation update state is set to PENDING, and the following values for the entry are set to the values being requested by the function invocation:

- requested view orientation matrix
- requested view mapping matrix
- requested view clipping limits
- requested x-y clipping indicator

The next time the workstation is updated, for any view table entry with the viewing transformation update state set to PENDING, the values stored in the requested entries replace the corresponding current values for that view index, and the view entry’s viewing transformation update state is set to NOTPENDING. Then regeneration proceeds, and the
new viewing representations take effect.

**Default Representation**

View index 0 is the default for the current view index in the traversal state list. View table entry 0 is the default view representation, and cannot be changed. The effect of the default representation is to map the VRC rectangle \([0,1] \times [0,1]\) to the NPC rectangle \([0,1] \times [0,1]\) without scaling, rotation, or translation. It contains the following values:

- **view orientation matrix**: \(\text{Identity Matrix}\)
- **view mapping matrix**: \(\text{Identity Matrix}\)
- **view clipping limits**: \(\text{XMIN} = 0 \quad \text{XMAX} = 1\) \(\text{YMIN} = 0 \quad \text{YMAX} = 1\)
- **x-y clipping indicator**: \(\text{Perform Clipping}\)

Some view mapping matrices may cause problems with rendering. See **EVALUATE VIEW MAPPING MATRIX 3 (3P)** for more detailed information.

**ERRORS**

- 003 Ignoring function, function requires state \((\text{PHOP, WSOP, *, *)}\)
- 054 Ignoring function, the specified workstation is not open
- 057 Ignoring function, specified workstation is of category MI
- 115 Ignoring function, the view index value is less than one
- 150 Ignoring function, setting this view table entry would exceed the maximum number of entries allowed in the workstation’s view table
- 153 Ignoring function, invalid view clipping limits; \(\text{XMIN} \geq \text{XMAX}\), \(\text{YMIN} \geq \text{YMAX}\), or \(\text{ZMIN} > \text{ZMAX}\)
- 154 Ignoring function, the view clipping limits are not within NPC range

**SEE ALSO**

- **SET VIEW INDEX (3P)**
- **EVALUATE VIEW ORIENTATION MATRIX (3P)**
- **EVALUATE VIEW MAPPING MATRIX (3P)**
- **SET VIEW TRANSFORMATION INPUT PRIORITY (3P)**
- **INQUIRE VIEW REPRESENTATION (3P)**
- **UPDATE WORKSTATION (3P)**
- **SET VIEW REPRESENTATION 3 (3P)**
NAME
SET VIEW REPRESENTATION 3 – define 3D view representation entry on workstation

SYNOPSIS
C Syntax
void
pset_view_rep3 ( ws, index, rep )
Pint ws; workstation identifier
Pint index; view index
Pview_rep3 *rep; view representation

FORTRAN Syntax
SUBROUTINE psvwr3 ( WKID, VIEWI, VWORMT, VWMPMT, VWCPLM, XYCLPI, BCLIPI, FCLIPI )
INTEGER WKID workstation identifier
INTEGER VIEWI view index
REAL VWORMT(4, 4) view orientation matrix
REAL VWMPMT(4, 4) view mapping matrix
REAL VWCPLM(6) view clipping limits (NPC) (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
INTEGER XYCLPI x-y clipping indicator (PNCLIP, PCLIP)
INTEGER BCLIPI back clipping indicator (PNCLIP, PCLIP)
INTEGER FCLIPI front clipping indicator (PNCLIP, PCLIP)

Required PHIGS
Operating States
(PHOP, WSOP, *, *)

DESCRIPTION
Purpose
SET VIEW REPRESENTATION 3 defines a view representation entry in the workstation’s view
table, using a three-dimensional model. A view representation defines how the
composited objects are to be represented in relation to the viewer.

A view representation controls the viewing stage of the transformation pipeline, which
transforms coordinates in World Coordinates (WC) to Normalized Projection Coordinates
(NPC), and optionally clips to the limits of NPC space.

C Input Parameters
ws The identifier of the workstation for which the view representation is being
defined.
index The view index of the entry being defined.
rep A pointer to a structure containing the values in the 3D view representation,
declared as follows:
typedef struct {
  Pmatrix3 ori_matrix; /* orientation matrix */
  Pmatrix3 map_matrix; /* mapping matrix */
  Plimit3 clip_limit; /* clipping limits */
  Pclip_ind xy_clip; /* x-y clipping indicator */
  Pclip_ind back_clip; /* back clipping indicator */
}
Pclip_ind front_clip; /* front clipping indicator */
} Pview_rep3;

Pmatrix3 is defined in phigs.h as follows:
typedef Pfloat Pmatrix3[4][4];

Plimit3 is defined in phigs.h as follows:
typedef struct {
    Pfloat x_min; /* x min */
    Pfloat x_max; /* x max */
    Pfloat y_min; /* y min */
    Pfloat y_max; /* y max */
    Pfloat z_min; /* z min */
    Pfloat z_max; /* z max */
} Plimit3;

Pclip_ind is defined in phigs.h as follows:
typedef enum {
    PIND_NO_CLIP
    PIND_CLIP
} Pclip_ind;

The ori_matrix is the 4 × 4 homogeneous view orientation matrix, probably from EVALUATE VIEW ORIENTATION MATRIX 3.

The map_matrix is the 4 × 4 homogeneous view mapping matrix, probably from EVALUATE VIEW MAPPING MATRIX 3.

The clip_limit contains the 6 Pfloat clipping limits: xmin, xmax, ymin, ymax, zmin, and zmax.

The xy_clip contains the x-y clip indicator for the x and y axes.

The back_clip contains the back clipping indicator, which controls clipping against the back plane, sometimes called the yon plane.

The front_clip contains the front clipping indicator, which controls clipping against the front plane, sometimes called the fore plane.

---

**FORTRAN Input Parameters**

**WKID**  The identifier of the workstation for which the view representation is being defined.

**VIEWI**  The view index of the entry being defined.

**VWORMT**  An array containing the 4 × 4 homogeneous view orientation matrix, probably from EVALUATE VIEW ORIENTATION MATRIX 3.

**VWMPMT**  An array containing the 4 × 4 homogeneous view mapping matrix, probably from EVALUATE VIEW MAPPING MATRIX 3.
VWCPLM
An array containing the 6 REAL clipping limits: XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.

XYCLPI
The x-y clip indicator for the x and y axes.

BCLPI
The back clipping indicator, which controls clipping against the back plane, sometimes called the yon plane.

FCLPI
The front clipping indicator, which controls clipping against the front plane, sometimes called the fore plane.

Valid clipping indicators as defined in phigs77.h are one of the enumeration values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PNCLIP No clipping</td>
</tr>
<tr>
<td>1</td>
<td>PCLIP Clipping</td>
</tr>
</tbody>
</table>

Execution

SET VIEW REPRESENTATION 3 defines a view representation entry in the workstation’s view table using a three-dimensional model.

The current view index attribute applied to each output primitive at traversal time determines which view representation is used to map the primitive from WC space to NPC space, typically with a parallel or perspective projection.

This viewing process for each output primitive is accomplished in three stages:

1. The view orientation matrix is applied to orient View Reference Coordinate (VRC) space with regard to WC space. Use the utility function EVALUATE VIEW ORIENTATION MATRIX 3 to generate this matrix from a view reference point, view plane normal vector, and view up vector specified in WC. The view reference point becomes the origin of VRC; the view plane normal and the view up vector orient the VRC space (VRC u and y axes) in relation to the WC axes.

2. The view mapping matrix is applied to map the VRC system to the NPC system. Use the utility function EVALUATE VIEW MAPPING MATRIX 3 to generate this matrix. The utility accepts a specified view volume called a window in VRC space to be mapped onto a specified projection viewport in NPC space. The projection is established in relation to the projection reference point, also specified in VRC space. The projection may be perspective or parallel, oblique or non-oblique.

3. Clipping limits are optionally applied. The view clipping limits define an area of NPC. When a clipping indicator is set to Clipping, any portion of an object beyond that limit is not displayed.

Note: These clipping limits are typically identical to the projection viewport provided to EVALUATE VIEW MAPPING MATRIX 3. If the clipping limits differ from the projection viewport, the object portions mapped by the view mapping matrix to be within the clipping limits in NPC are displayed.
In any case, objects are subject to workstation clipping, which may not be disabled. The maximum volume of NPC space that may be displayed is from 0 to 1 in all dimensions. The immediate visual effect of changing a view representation depends on the workstation’s display update state. When SET VIEW REPRESENTATION is called, the display update state may prohibit immediate updating (that is, regeneration) of the workstation’s display. In this case, the view index view table entry’s viewing transformation update state is set to PENDING, and the following values for the entry are set to the values being requested by the function invocation:

- requested view orientation matrix
- requested view mapping matrix
- requested view clipping limits
- requested x-y clipping indicator
- requested back clipping indicator
- requested front clipping indicator

The next time the workstation is updated, for any view table entry with the viewing transformation update state set to PENDING, the values stored in the requested entries replace the corresponding current values for that view index, and the view entry’s viewing transformation update state is set to NOTPENDING. Then regeneration proceeds, and the new viewing representations take effect.

**Default Representation**

View index 0 is the default for the current view index in the traversal state list. View table entry 0 is the default view representation, and cannot be changed. The effect of the default representation is to map the cube \([0,1] \times [0,1] \times [0,1]\) in VRC space to the cube \([0,1] \times [0,1] \times [0,1]\) in NPC space, without any scaling, rotation, or translation, and to clip the image in NPC space at the boundaries of the same unit cube. It contains the following values:

- view orientation matrix: Identity Matrix
- view mapping matrix: Identity Matrix
- view clipping limits: \(XMIN = 0\) \(XMAX = 1\), \(YMIN = 0\) \(YMAX = 1\), \(ZMIN = 0\) \(ZMAX = 1\)
- x-y clipping indicator: Perform Clipping
- back clipping indicator: Perform Clipping
- front clipping indicator: Perform Clipping

Some view mapping matrices may cause problems with rendering. See EVALUATE VIEW MAPPING MATRIX 3 (3) for more detailed information.

**ERRORS**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>057</td>
<td>Ignoring function, specified workstation is of category MI</td>
</tr>
<tr>
<td>115</td>
<td>Ignoring function, the view index value is less than one</td>
</tr>
</tbody>
</table>

960 modified 2 April 1993
150 Ignoring function, setting this view table entry would exceed the maximum number of entries allowed in the workstation’s view table
153 Ignoring function, invalid view clipping limits; XMIN \geq XMAX, YMIN \geq YMAX, or ZMIN > ZMAX
154 Ignoring function, the view clipping limits are not within NPC range

SEE ALSO

SET VIEW INDEX (3P)
EVALUATE VIEW ORIENTATION MATRIX (3P)
EVALUATE VIEW MAPPING MATRIX (3P)
SET VIEW TRANSFORMATION INPUT PRIORITY (3P)
INQUIRE VIEW REPRESENTATION (3P)
UPDATE WORKSTATION (3P)
SET VIEW REPRESENTATION (3P)
**NAME**  
SET VIEW TRANSFORMATION INPUT PRIORITY – assign relative priority of view representation to use when transforming input values

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE psvtip ( WKID, VIEWI, RFVWIX, RELPRI )
INTEGER WKID /* workstation identifier */
INTEGER VIEWI /* view index */
INTEGER RFVWIX /* reference view index */
INTEGER RELPRI /* relative priority (PHIGHR, PLOWER) */
```

**DESCRIPTION**

**Purpose**
SET VIEW TRANSFORMATION INPUT PRIORITY assigns a relative priority to an entry in the workstation’s view table, to be used when transforming LOCATOR or STROKE operator input coordinates from Device Coordinates (DC) to World Coordinates (WC).

**C Input Parameters**

- `ws` The identifier of the workstation for which the view transformation input priority is being set.
- `index` The priority is being set for this `view index`.
- `ref_index` The priority of `index` is set just higher or lower than that of this `reference view index`.
- `priority` The relative priority for `index`, relative to the `ref_index`; one of the enumerated values:
  - `PPRI_HIGHER` Higher
  - `PPRI_LOWER` Lower

**FORTRAN Input Parameters**

- `WKID` The identifier of the workstation for which the view transformation input priority is being set.
- `VIEWI` The priority is being set for this `view index`.
- `RFVWIX` The priority of `VIEWI` is set just higher or lower than that of this `reference view index`.
- `RELPRI` The relative priority for `VIEWI`, relative to the `VWORMT`; one of the enumerated values:
  - `PHIGHR` Higher
  - `PLOWER` Lower

962 modified 2 April 1993
Execution

SET VIEW TRANSFORMATION INPUT PRIORITY assigns a relative priority to a view representation entry in the workstation’s view table, to be used when transforming LOCATOR or STROKE operator input coordinates from DC to WC. The priority of view index is set just higher or just lower than the priority of the reference view index.

LOCATOR or STROKE operator input data are transformed by the inverse viewing pipeline, from DC to WC. First, the inverse workstation transformation (in effect when the input is generated) maps the DC values to Normalized Projection Coordinates (NPC), always resulting in values in the NPC unit cube. Then, the NPC positions need to be mapped to WC by the inverse of one of the viewing transformations; the relative priority order of the viewing transformations is used to determine which viewing transformation inverse to apply.

Views with lower priority than view 0’s will never be used to map input data. View 0 is the identity transformation encompassing the entire NPC space cube and cannot be changed. Therefore, any input position is necessarily in view 0, and lower priority views are effectively disabled. The initial ordering, when a workstation is opened, gives view number 0 the highest priority, view 1 the next highest, and so on. All views except 0 are disabled.

If the view index is the same as the reference view index, the function has no effect.

The relative view transformation input priorities may be obtained using INQUIRE LIST OF VIEW INDICES.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)
054 Ignoring function, the specified workstation is not open
057 Ignoring function, specified workstation is of category MI
114 Ignoring function, the view index value is less than zero
101 Ignoring function, the specified representation has not been defined

SEE ALSO

SET VIEW REPRESENTATION 3 (3P)
INQUIRE LIST OF VIEW INDICES (3P)


**NAME**

SET WORKSTATION VIEWPORT – set 2D viewport limits for specified workstation

**SYNOPSIS**

### C Syntax

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

### FORTRAN Syntax

```fortran
SUBROUTINE pswkv ( WKID, XMIN, XMAX, YMIN, YMAX )
INTEGER   WKID    workstation identifier
REAL      XMIN, XMAX, YMIN, YMAX    workstation viewport limits (DC)
```

**Required PHIGS Operating States**

( PHOP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

SET WORKSTATION VIEWPORT defines a 2D area in Device Coordinate (DC) space, at a constant z value. The abstract image within the workstation window, defined in Normalized Projection Coordinate (NPC) space, is mapped to this viewport.

Together, the workstation window and the workstation viewport define the **workstation transformation** that converts the image from NPC to DC of workstation’s physical display surface. The workstation window defines what within NPC space is displayed; the workstation viewport defines where the image is displayed in DC space.

Control of the workstation transformation allows you to choose a portion of the abstract image composed in NPC space for display on the workstation, without changing the definition of the image in NPC space.

**C Input Parameters**

- **ws**
  The identifier of the workstation for which the viewport is to be set.

- **viewport**
  A pointer to a structure containing the DC limits for the 2D workstation viewport, defined as follows:

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

**FORTRAN Input Parameters**

- **WKID**
  The identifier of the workstation for which the viewport is to be set.

- **XMIN**
  The x minimum value in DC for the 2D workstation viewport.

- **XMAX**
  The x maximum value in DC for the 2D workstation viewport.

- **YMIN**
  The y minimum value in DC for the 2D workstation viewport.

964 modified 2 April 1993
**Execution**

The current workstation viewport defines the area in Device Coordinates in which the abstract image within the current workstation window is displayed. The workstation window is defined in Normalized Projection Coordinates. Together, the workstation window and the workstation viewport define the *workstation transformation* that converts the image from NPC to DC of workstation’s physical display surface.

The $x$ minimum must be non-negative and less than $x$ maximum and $y$ minimum must be non-negative and less than $y$ maximum. In addition, all values must be within the workstation’s DC range.

`SET WORKSTATION VIEWPORT` sets the $x$ and $y$ components of the requested workstation viewport in the specified workstation’s state list to the values specified. The $z$ component of the requested workstation viewport and current workstation viewport are not changed. The effect of calling `SET WORKSTATION VIEWPORT` is visible only after the requested workstation viewport replaces the current workstation viewport. The time at which this occurs depends on the workstation’s display update state. This action is performed immediately, and the workstation transformation update state is set to NOTPENDING, if any one of the following is true:

1. The workstation’s display update state allows update.
2. The workstation’s modification mode is any value other than *No Immediate Visual Effect*, and the dynamic modification accepted for workstation transformation entry in the workstation description table is set to Immediate.
3. The display space empty status in the workstation state list is EMPTY.

Otherwise, the workstation transformation update state is set to PENDING, and the requested workstation viewport will not replace the current workstation viewport until the next time the workstation is updated. The workstation transformation update state will be set to NOTPENDING at that time.

If the current workstation window and viewport do not have the same aspect ratios, the workstation transformation preserves the proportions of the image by mapping the workstation window to the largest possible area of the workstation viewport such that:

- The aspect ratio of the window in $x$ and $y$ is maintained.
- The lower left hand corner of the window is mapped to the lower left hand corner of the viewport.

If the aspect ratios of the workstation window and viewport are different, there will be unused space along the upper or right-hand edges of the viewport, but not both. The default workstation transformation maps the entire NPC view plane, $[0,1] \times [0,1] \times [0,1]$, onto the largest square area in the workstation display space including the display’s lower left corner.
| ERRORS | 003  | Ignoring function, function requires state (PHOP, WSOP, *, *) |
|        | 054  | Ignoring function, the specified workstation is not open |
|        | 057  | Ignoring function, specified workstation is of category MI |
|        | 152  | Ignoring function, invalid viewport; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN > ZMAX |
|        | 157  | Ignoring function, the workstation viewport is not within display space |

SEE ALSO

SET WORKSTATION WINDOW (3P)
SET WORKSTATION VIEWPORT 3 (3P)
### NAME
SET WORKSTATION VIEWPORT 3 – set 3D viewport limits for specified workstation

### SYNOPSIS

**C Syntax**
```c
void
pset_ws_vp3 ( ws, viewport )

Pint ws; // workstation id
Plimit3 *viewport; // workstation viewport limits
```

**FORTRAN Syntax**
```
SUBROUTINE pswkv3 ( WKID, WKVP )

INTEGER WKID // workstation identifier
REAL WKVP(6) // workstation viewport limits (DC) (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
```

### Required PHIGS Operating States
(PHOP, WSOP, *, *)

### DESCRIPTION

**Purpose**

SET WORKSTATION VIEWPORT 3 defines a 3D volume in Device Coordinate (DC) space. The abstract image within the workstation window, defined in Normalized Projection Coordinate (NPC) space, will be mapped to this viewport.

Together, the workstation window and the workstation viewport define the *workstation transformation* that converts the image from NPC to DC on the workstation’s physical display surface. The workstation window defines what within NPC space will be displayed; the workstation viewport defines where the image is displayed in DC space.

Control of the workstation transformation allows you to choose a portion of the abstract image composed in NPC space for display on the workstation, without changing the definition of the image in NPC space.

**C Input Parameters**

- **ws** The identifier of the workstation for which the viewport is to be set.
- **viewport** A pointer to a structure containing the six DC *workstation viewport limits*, defined as follows:

```c
typedef struct {
    Pfloat x_min;        /* x minimum */
    Pfloat x_max;        /* x maximum */
    Pfloat y_min;        /* y minimum */
    Pfloat y_max;        /* y maximum */
    Pfloat z_min;        /* z minimum */
    Pfloat z_max;        /* z maximum */
} Plimit3;
```

### modified 2 April 1993

967
FORTRAN Input

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKID</td>
<td>The identifier of the workstation for which the viewport is to be set.</td>
</tr>
<tr>
<td>WKVP</td>
<td>An array containing the six workstation viewport limits (DC) to set, ordered as follows:</td>
</tr>
</tbody>
</table>

  - x minimum
  - x maximum
  - y minimum
  - y maximum
  - z minimum
  - z maximum

Execution

The current workstation viewport defines the volume in Device Coordinates in which the abstract image within the current workstation window is displayed. The workstation window is defined in Normalized Projection Coordinates. Used together, the workstation window and the workstation viewport define an isotropic workstation transformation that converts the image from NPC to DC of workstation’s physical display surface.

The x minimum must be non-negative and less than x maximum, the y minimum must be non-negative and less than y maximum, and the z minimum must be non-negative and less than z maximum. In addition, all values must be within the workstation’s DC range.

SET WORKSTATION VIEWPORT 3 sets the requested workstation viewport in the specified workstation’s state list to the values specified in viewport. The effect of calling SET WORKSTATION VIEWPORT 3 is visible only after the requested workstation viewport replaces the current workstation viewport. The time at which this occurs depends on the workstation’s display update state. This action is performed immediately, and the workstation transformation update state is set to NOTPENDING, if any one of the following is true:

1. The workstation’s display update state allows update.
2. The workstation’s modification mode is any value other than No Immediate Visual Effect, and the dynamic modification accepted for workstation transformation entry in the workstation description table is set to Immediate.
3. The display space empty status in the workstation state list is EMPTY.

Otherwise, the workstation transformation update state is set to PENDING, and the requested workstation viewport will not replace the current workstation viewport until the next time the workstation is updated. The workstation transformation update state is set to NOTPENDING at that time.

If the current workstation window and viewport do not have the same aspect ratios, the workstation transformation will preserve the proportions of the image by mapping the workstation window to the largest parallelepiped within the workstation viewport such that:

- The aspect ratio of the window in x and y is maintained.
- The lower left hand corner of the window closest to 0 is mapped to the lower left hand corner of the viewport furthest from the observer.
The $z$ extent of the workstation window is mapped to the entire $z$ extent of the workstation viewport.

If the aspect ratios of the workstation window and viewport are different, there will be unused space along the upper or right-hand edges of the viewport, but not both.

The default workstation transformation maps the entire NPC view volume, $[0,1] \times [0,1] \times [0,1]$, onto the largest square in the workstation display space including the display's lower left corner furthest from the observer.

**ERRORS**

003  Ignoring function, function requires state (PHOP, WSOP, *, *)
054  Ignoring function, the specified workstation is not open
057  Ignoring function, specified workstation is of category MI
152  Ignoring function, invalid viewport; $XMIN \geq XMAX$, $YMIN \geq YMAX$, or $ZMIN > ZMAX$
157  Ignoring function, the workstation viewport is not within display space

**SEE ALSO**

SET WORKSTATION WINDOW 3 (3P)
SET WORKSTATION VIEWPORT (3P)
**NAME**

SET WORKSTATION WINDOW – specify 2D window to be displayed on specified workstation

**SYNOPSIS**

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pswkw ( WKID, XMIN, XMAX, YMIN, YMAX )
INTEGER WKID /* workstation identifier */
REAL XMIN, XMAX, YMIN, YMAX /* workstation window limits (NPC) */
```

**Required PHIGS Operating States**

(HP, WSOP, *, *)

**DESCRIPTION**

**Purpose**

SET WORKSTATION WINDOW defines the 2D area in Normalized Projection Coordinate (NPC) space to be displayed on the specified workstation. The workstation window is a rectangular box in NPC space, which is mapped to the workstation viewport, defined in Device Coordinate (DC) space.

Together, the workstation window and the workstation viewport define the *workstation transformation* that converts the image from NPC to DC on the workstation’s physical display surface. The workstation window defines what within NPC space is displayed; the workstation viewport defines where the image is displayed in DC space.

Control of the workstation window allows you to choose a portion of the abstract image composed in NPC space for display on the workstation, without changing the definition of the image in NPC space.

**C Input Parameters**

- **ws** The identifier of the workstation for which the window is to be set.
- **window** A pointer to a structure containing the four 2D NPC *workstation window limits*, defined as follows:

  ```c
typedef struct {
    Pfloat x_min; /* x minimum */
    Pfloat x_max; /* x maximum */
    Pfloat y_min; /* y minimum */
    Pfloat y_max; /* y maximum */
  } Plimit;
```

**FORTRAN Input Parameters**

- **WKID** The identifier of the workstation for which the window is to be set.
- **XMIN** The *x minimum* value in NPC for the 2D workstation window.
- **XMAX** The *x maximum* value in NPC for the 2D workstation window.
Execution

The current workstation window defines the rectangular box in Normalized Projection Coordinates (NPC) containing the portion of the abstract image to be displayed within the current workstation viewport. Together, the workstation window and the workstation viewport define an isotropic workstation transformation that converts the image from NPC to DC on the workstation’s physical display surface.

The range for each of the workstation window limits is [0,1]. In addition, \( x_{\text{minimum}} \) must be less than \( x_{\text{maximum}} \), and \( y_{\text{minimum}} \) must be less than \( y_{\text{maximum}} \). Output is automatically clipped at the workstation window limits, and this clipping cannot be disabled.

SET WORKSTATION WINDOW sets the \( x \) and \( y \) components of the requested workstation window in the specified workstation’s state list to the values specified. The \( z \) component of the requested workstation window and current workstation window is not changed.

The effect of calling SET WORKSTATION WINDOW is visible only after the requested workstation window replaces the current workstation window. The time at which this occurs depends on the workstation’s display update state. This action is performed immediately, and the workstation transformation update state is set to NOTPENDING, if any one of the following is true:

1. The workstation’s display update state allows update.
2. The workstation’s modification mode is any value other than No Immediate Visual Effect, and the dynamic modification accepted for workstation transformation entry in the workstation description table is set to Immediate.
3. The display space empty status in the workstation state list is EMPTY.

Otherwise, the workstation transformation update state is set to PENDING, and the requested workstation window will not replace the current workstation window until the next time the workstation is updated. The workstation transformation update state will be set to NOTPENDING at that time.

If the current workstation window and viewport do not have the same aspect ratios, the workstation transformation will preserve the proportions of the image by mapping the workstation window to the largest possible area within of workstation viewport such that

- The aspect ratio of the window in \( x \) and \( y \) is maintained.
- The lower left hand corner of the window is mapped to the lower left hand corner of the viewport.

If the aspect ratios of the workstation window and viewport are different, there will be unused space along the upper or right-hand edges of the viewport, but not both. The default workstation transformation maps the entire NPC view volume, \([0,1] \times [0,1] \times [0,1]\), onto the largest square in the workstation display space including the display’s lower left corner.

modified 2 April 1993
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Ignoring function, function requires state (PHOP, WSOP, *, *)</td>
</tr>
<tr>
<td>054</td>
<td>Ignoring function, the specified workstation is not open</td>
</tr>
<tr>
<td>057</td>
<td>Ignoring function, specified workstation is of category MI</td>
</tr>
<tr>
<td>151</td>
<td>Ignoring function, invalid window; XMIN ≥ XMAX, YMIN ≥ YMAX, or ZMIN &gt; ZMAX</td>
</tr>
<tr>
<td>156</td>
<td>Ignoring function, the workstation window limits are not within NPC range</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- SET WORKSTATION VIEWPORT (3P)
- SET WORKSTATION WINDOW 3 (3P)
### NAME

SET WORKSTATION WINDOW 3 – specify 3D window to be displayed on specified workstation

### SYNOPSIS

**C Syntax**

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

**FORTRAN Syntax**

```fortran
SUBROUTINE pswkw3 ( WKID, WKWN )
INTEGER WKID         /* workstation identifier */
REAL WKWN(6)         /* workstation window limits (NPC) (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX) */
```

### Required PHIGS Operating States

(PHOP, WSOP, *, *)

### DESCRIPTION

#### Purpose

SET WORKSTATION WINDOW 3 defines the 3D volume in Normalized Projection Coordinate (NPC) space to be displayed on the specified workstation. The workstation window is a rectangular box in NPC space, which is mapped to the workstation viewport, defined in Device Coordinate (DC) space.

Together, the workstation window and the workstation viewport define the *workstation transformation* that converts the image from NPC to DC on the workstation’s physical display surface. The workstation window defines what within NPC space is displayed; the workstation viewport defines where the image is displayed in DC space.

Control of the workstation window allows you to choose a portion of the abstract image composed in NPC space for display on the workstation, without changing the definition of the image in NPC space.

**C Input Parameters**

**ws**

The identifier of the workstation for which the window is to be set.

**window**

A pointer to a structure containing the six NPC *workstation window limits*, defined as follows:

```c
typedef struct {
    Pfloat x_min;       /* x minimum */
    Pfloat x_max;       /* x maximum */
    Pfloat y_min;       /* y minimum */
    Pfloat y_max;       /* y maximum */
    Pfloat z_min;       /* z minimum */
    Pfloat z_max;       /* z maximum */
} Plimit3;
```
### FORTRAN Input Parameters

- **WKID** - The identifier of the workstation for which the window is to be set.
- **WKWN** - An array containing the six workstation window limits to set (in NPC), ordered as follows:
  - x minimum
  - x maximum
  - y minimum
  - y maximum
  - z minimum
  - z maximum

### Execution

The current workstation window defines the rectangular box in Normalized Projection Coordinates (NPC) containing the portion of the abstract image to be displayed within the current workstation viewport. Together, the workstation window and the workstation viewport define an isotropic workstation transformation that converts the image from NPC to DC on the workstation’s physical display surface.

The range for each of the workstation window limits is [0,1]. In addition, x minimum must be less than x maximum, y minimum must be less than y maximum, and z minimum must be less than or equal to z maximum. Output is automatically clipped at the workstation window limits, and this clipping cannot be disabled.

SET WORKSTATION WINDOW 3 sets the requested workstation window in the specified workstation’s state list to the values specified in window. The effect of calling SET WORKSTATION WINDOW 3 is visible only after the requested workstation window replaces the current workstation window. The time at which this occurs depends on the workstation’s display update state. This action is performed immediately, and the workstation transformation update state is set to NOTPENDING, if any one of the following is true:

1. The workstation’s display update state allows update.
2. The workstation’s modification mode is any value other than No Immediate Visual Effect, and the dynamic modification accepted for workstation transformation entry in the workstation description table is set to Immediate.
3. The display space empty status in the workstation state list is EMPTY.

Otherwise, the workstation transformation update state is set to PENDING, and the requested workstation window will not replace the current workstation window until the next time the workstation is updated. The workstation transformation update state will be set to NOTPENDING at that time.

If the current workstation window and viewport do not have the same aspect ratios, the workstation transformation will preserve the proportions of the image by mapping the workstation window to the largest parallelepiped within the workstation viewport such that:

- The aspect ratio of the window in x and y is maintained.
- The lower left hand corner of the window closest to 0 is mapped to the lower left
The hand corner of the viewport furthest from the observer.

- The \( z \) extent of the workstation window is mapped to the entire \( z \) extent of the workstation viewport.

If the aspect ratios of the workstation window and viewport are different, there will be unused space along the upper or right-hand edges of the viewport, but not both.

The default workstation transformation maps the entire NPC view volume, \([0,1] \times [0,1] \times [0,1]\), onto the largest square in the workstation display space including the display’s lower left corner furthest from the observer.

**ERRORS**

003 Ignoring function, function requires state ( PHOP, WSOP, *, *

054 Ignoring function, the specified workstation is not open

057 Ignoring function, specified workstation is of category MI

151 Ignoring function, invalid window; \( X_{\text{MIN}} \geq X_{\text{MAX}}, Y_{\text{MIN}} \geq Y_{\text{MAX}}, \text{or} \ Z_{\text{MIN}} \geq Z_{\text{MAX}} \)

156 Ignoring function, the workstation window limits are not within NPC range

**SEE ALSO**

SET WORKSTATION VIEWPORT 3 (3P)

SET WORKSTATION WINDOW (3P)
Index

A
Abbreviations, description, 1
ADD NAMES TO SET
  padd_names_set—C function, 22
  pads—FORTRAN function, 22
ANNOTATION TEXT RELATIVE
  panno_text_rel—C function, 24
  patr—FORTRAN function, 24
ANNOTATION TEXT RELATIVE 3
  panno_text_rel3—C function, 28
  patr3—FORTRAN function, 28
APPLICATION DATA
  pap—FORTRAN function, 32
  pappl_data—C function, 32
ARCHIVE ALL STRUCTURES
  par_all_structs—C function, 34
  parast—FORTRAN function, 34
Archive Files
  ARCHIVE ALL STRUCTURES, 34
  ARCHIVE STRUCTURE NETWORKS, 36
  ARCHIVE STRUCTURES, 38
  CLOSE ARCHIVE FILE, 62
  DELETE ALL STRUCTURES FROM ARCHIVE, 81
  DELETE STRUCTURE NETWORKS FROM ARCHIVE, 91
  DELETE STRUCTURES FROM ARCHIVE, 93
  INQUIRE ALL CONFLICTING STRUCTURES,
Aspect Source Flags, continued
  SET INTERIOR COLOUR INDEX, 857
  SET INTERIOR STYLE, 864
  SET INTERIOR STYLE INDEX, 867
  SET LINETYPE, 871
  SET LINEWIDTH SCALE FACTOR, 873
  SET MARKER TYPE, 886
  SET POLYLINE COLOUR INDEX, 911
  SET POLYLINE INDEX, 913
  SET TEXT COLOUR INDEX, 935
  SET TEXT FONT, 937
  SET TEXT PRECISION, 943

Attribute Representations
  INQUIRE COLOUR REPRESENTATION, 415
  INQUIRE EDGE REPRESENTATION, 505
  INQUIRE INTERIOR REPRESENTATION, 599
  INQUIRE PATTERN REPRESENTATION, 660
  INQUIRE POLYLINE REPRESENTATION, 683
  INQUIRE POLYMARKER REPRESENTATION, 689
  INQUIRE TEXT REPRESENTATION, 756
  INQUIRE VIEW REPRESENTATION, 769
  SET COLOUR REPRESENTATION, 811
  SET EDGE REPRESENTATION, 827
  SET INTERIOR REPRESENTATION, 861
  SET PATTERN REPRESENTATION, 900
  SET POLYLINE REPRESENTATION, 915
  SET POLYMARKER REPRESENTATION, 922
  SET TEXT REPRESENTATION, 945
  SET VIEW REPRESENTATION, 953
  SET VIEW REPRESENTATION 3, 957

Attributes, Bundled
  SET INDIVIDUAL ASF, 853

Attributes, Edge Attributes
  SET EDGE COLOUR INDEX, 821
  SET EDGE FLAG, 823
  SET EDGE INDEX, 825
  SET EDGETYPE, 830
  SET EDGEWIDTH SCALE FACTOR, 833

Attributes, Filled Area Interior Attributes
  SET INTERIOR COLOUR INDEX, 857
  SET INTERIOR INDEX, 859
  SET INTERIOR STYLE, 864
  SET INTERIOR STYLE INDEX, 867

Attributes, Filled Area Interior Attributes, continued
  SET PATTERN REFERENCE POINT, 896
  SET PATTERN REFERENCE POINT AND VECTORS, 898
  SET PATTERN SIZE, 902

Attributes, Line Attributes
  SET LINETYPE, 871
  SET LINEWIDTH SCALE FACTOR, 873
  SET POLYLINE COLOUR INDEX, 911
  SET POLYLINE INDEX, 913

Attributes, Marker Attributes
  SET MARKER SIZE SCALE FACTOR, 884
  SET MARKER TYPE, 886
  SET POLYMARKER COLOUR INDEX, 918
  SET POLYMARKER INDEX, 920

Attributes, Modelling and Transformation Attributes
  RESTORE MODELLING CLIPPING VOLUME, 317
  SET GLOBAL TRANSFORMATION, 843
  SET GLOBAL TRANSFORMATION 3, 845
  SET LOCAL TRANSFORMATION, 875
  SET LOCAL TRANSFORMATION 3, 878
  SET MODELLING CLIPPING INDICATOR, 888
  SET MODELLING CLIPPING VOLUME, 890
  SET MODELLING CLIPPING VOLUME 3, 893
  SET VIEW INDEX, 951

Attributes, Special Attributes
  ADD NAMES TO SET, 22
  REMOVE NAMES FROM SET, 294
  SET HLHSR IDENTIFIER, 849
  SET PICK IDENTIFIER, 906

Attributes, Text Attributes
  SET ANNOTATION STYLE, 787
  SET ANNOTATION TEXT ALIGNMENT, 789
  SET ANNOTATION TEXT CHARACTER HEIGHT, 792
  SET ANNOTATION TEXT CHARACTER UP VECTOR, 794
  SET ANNOTATION TEXT PATH, 796
  SET CHARACTER EXPANSION FACTOR, 798
  SET CHARACTER HEIGHT, 800
  SET CHARACTER SPACING, 802
  SET CHARACTER UP VECTOR, 804

Index-2
Attributes, Text Attributes, continued
  SET TEXT ALIGNMENT, 932
  SET TEXT COLOUR INDEX, 935
  SET TEXT FONT, 937
  SET TEXT INDEX, 939
  SET TEXT PATH, 941
  SET TEXT PRECISION, 943

AWAIT EVENT
  pawait_event—C function, 40
  pwait—FORTRAN function, 40

B
BUILD TRANSFORMATION MATRIX
  pbltm—FORTRAN function, 43
  pbldt_tran_matrix—C function, 43
BUILD TRANSFORMATION MATRIX 3
  pbltm3—FORTRAN function, 46
  pbldt_tran_matrix3—C function, 46

Bundled Attributes
  SET INDIVIDUAL ASF, 853

C
C Binding Notes, description, 4
C functions
  padd_names_set, 22
  panno_text_rel, 24
  panno_text_rel3, 28
  pappl_data, 32
  par_all_structs, 34
  par_struct_nets, 36
  par_structs, 38
  pawait_event, 40
  pbldt_tran_matrix, 43
  pbldt_tran_matrix3, 46
  pcell_array, 49
  pcell_array3, 52
  pchange_struct_id, 55
  pchange_struct_id_refs, 57
  pchange_struct_refs, 60
  pc1ose_ar_file, 62
  pc1ose_phigs, 64
  pc1ose_struct, 65
  pc1ose_ws(ws)_id, 66
  pcompose_matrix, 68
  pcompose_matrix3, 70
  pcompose_tran_matrix, 72
  pcompose_tran_matrix3, 75
  pcopy_all elems_struct, 78
  pcreate_store, 79
  pdel_all structs, 80
  pdel_all structs_ar, 81
  pdel_elem, 82
  pdel_elem_range, 83
  pdel_elems_labels, 84
  pdel_store, 86
  pdel_strct, 87
  pdel_strct_net, 89
  pdel_strct_nets_ar, 91
  pdel_strct_net3, 93
  pelem_search, 95
  pemergency_close_phigs, 103
  pempty_strct, 104
  perr_hand, 105
  perr_log, 107
  peval_view_map_matrix, 109
  peval_view_map_matrix3, 112
  peval_view_ori_matrix, 117
  peval_view_ori_matrix3, 119
  pexec_struct, 122
  pfill_area, 124
  pfill_area_set, 129
  pfill_area_set3, 132
  pfill_area3, 126
  pflush_events, 135
  pget_choice, 137
  pget_item_type, 139
  pget_loc, 140
  pget_loc3, 142
  pget_pick, 144
  pget_string, 147
  pget_stroke, 149
  pget_stroke3, 151
  pget_val, 153
  pincr_spa_search, 154
  pincr_spa_search3, 160
  pinit_choice, 166
  pinit_choice3, 175
  pinit_loc, 184
  pinit_loc3, 191

Index—3
<table>
<thead>
<tr>
<th>C functions, continued</th>
<th>C functions, continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>pinit_pick, 198</td>
<td>pinq_elem_content, 509</td>
</tr>
<tr>
<td>pinit_pick3, 206</td>
<td>pinq_elem_ptr, 566</td>
</tr>
<tr>
<td>pinit_string, 214</td>
<td>pinq_elem_type_size, 567</td>
</tr>
<tr>
<td>pinit_string3, 219</td>
<td>pinq_err_hand_mode, 574</td>
</tr>
<tr>
<td>pinit_stroke, 224</td>
<td>pinq_gdp, 575</td>
</tr>
<tr>
<td>pinit_stroke3, 232</td>
<td>pinq_gdp3, 578</td>
</tr>
<tr>
<td>pinit_val, 241</td>
<td>pinq_gse_facs, 581</td>
</tr>
<tr>
<td>pinit_val3, 250</td>
<td>pinq_highl_filter, 583</td>
</tr>
<tr>
<td>pinq_all_conf structs, 388</td>
<td>pinq_hlhsr_id_facs, 586</td>
</tr>
<tr>
<td>pinq_anno_facs, 390</td>
<td>pinq_hlhsr_mode, 592</td>
</tr>
<tr>
<td>pinq_ar_files, 393</td>
<td>pinq_hlhsr_mode_facs, 589</td>
</tr>
<tr>
<td>pinq_ar_st, 395</td>
<td>pinq_in_overf, 594</td>
</tr>
<tr>
<td>pinq_choice_st, 396</td>
<td>pinq_int_facs, 596</td>
</tr>
<tr>
<td>pinq_choice_st3, 401</td>
<td>pinq_int_rep, 599</td>
</tr>
<tr>
<td>pinq_colr_facs, 406</td>
<td>pinq_invis_filter, 602</td>
</tr>
<tr>
<td>pinq_colr_map_method_st, 409</td>
<td>pinq_line_facs, 680</td>
</tr>
<tr>
<td>pinq_colr_model, 411</td>
<td>pinq_line_rep, 683</td>
</tr>
<tr>
<td>pinq_colr_model_facs, 413</td>
<td>pinq_list_avail_gdp, 605</td>
</tr>
<tr>
<td>pinq_colr_rep, 415</td>
<td>pinq_list_avail_gdp3, 607</td>
</tr>
<tr>
<td>pinq_conf_res, 418</td>
<td>pinq_list_avail_gse, 609</td>
</tr>
<tr>
<td>pinq_conf structs_net, 420</td>
<td>pinq_list_avail_ws_types, 612</td>
</tr>
<tr>
<td>pinq_cur_elem_content, 423</td>
<td>pinq_list_colr_inds, 614</td>
</tr>
<tr>
<td>pinq_cur_elem_type_size, 426</td>
<td>pinq_list_edge_inds, 616</td>
</tr>
<tr>
<td>pinq_def_choice_data, 428</td>
<td>pinq_list_int_inds, 618</td>
</tr>
<tr>
<td>pinq_def_choice_data3, 433</td>
<td>pinq_list_line_inds, 622</td>
</tr>
<tr>
<td>pinq_def_disp_upd_st, 438</td>
<td>pinq_list_marker_inds, 624</td>
</tr>
<tr>
<td>pinq_def_loc_data, 440</td>
<td>pinq_list_pat_inds, 620</td>
</tr>
<tr>
<td>pinq_def_loc_data3, 445</td>
<td>pinq_list_text inds, 626</td>
</tr>
<tr>
<td>pinq_def_pick_data, 450</td>
<td>pinq_list_view_inds, 628</td>
</tr>
<tr>
<td>pinq_def_pick_data3, 455</td>
<td>pinq_loc_st, 630</td>
</tr>
<tr>
<td>pinq_def_string_data, 460</td>
<td>pinq_loc_st3, 635</td>
</tr>
<tr>
<td>pinq_def_string_data3, 464</td>
<td>pinq_marker_facs, 686</td>
</tr>
<tr>
<td>pinq_def_stroke_data, 468</td>
<td>pinq_marker_rep, 689</td>
</tr>
<tr>
<td>pinq_def_stroke_data3, 473</td>
<td>pinq_model_clip_facs, 640</td>
</tr>
<tr>
<td>pinq_def_val_data, 478</td>
<td>pinq_more_simult_events, 642</td>
</tr>
<tr>
<td>pinq_def_val_data3, 483</td>
<td>pinq_num_avail_in, 644</td>
</tr>
<tr>
<td>pinq_disp_space_size, 488</td>
<td>pinq_num_disp_pris, 646</td>
</tr>
<tr>
<td>pinq_disp_upd_st, 494</td>
<td>pinq_open_struct, 648</td>
</tr>
<tr>
<td>pinq_display_space_size3, 491</td>
<td>pinq_open_wss, 717</td>
</tr>
<tr>
<td>pinq_dyns_structs, 497</td>
<td>pinq_pat_fac, 658</td>
</tr>
<tr>
<td>pinq_dyns_ws_attrs, 499</td>
<td>pinq_pat_rep, 660</td>
</tr>
<tr>
<td>pinq_edge_facs, 502</td>
<td>pinq_paths_ances, 650</td>
</tr>
<tr>
<td>pinq_edge_rep, 505</td>
<td>pinq_paths_descs, 654</td>
</tr>
<tr>
<td>pinq_edit_mode, 508</td>
<td>pinq_phigs_facs, 663</td>
</tr>
</tbody>
</table>
C functions, continued

pinq_pick_st, 666
pinq_pick_st3, 673
pinq_posted_structs, 692
pinq_pred_colr_rep, 695
pinq_pred_edge_rep, 698
pinq_pred_int_rep, 701
pinq_pred_line_rep, 707
pinq_pred_marker_rep, 709
pinq_pred_pat_rep, 704
pinq_pred_text_rep, 711
pinq_pred_view_rep, 714
pinq_string_st, 721
pinq_string_st3, 726
pinq_stroke_st, 731
pinq_stroke_st3, 736
pinq_struct_ids, 742
pinq_struct_st, 744
pinq_struct_status, 745
pinq_sys_st, 747
pinq_text_extent, 748
pinq_text_facs, 753
pinq_text_rep, 756
pinq_val_st, 759
pinq_val_st3, 763
pinq_view_facs, 767
pinq_view_rep, 769
pinq_ws_cat, 772
pinq_ws_class, 774
pinq_ws_conn_type, 776
pinq_ws_st, 780
pinq_ws_st_table, 778
pinq_ws_tran, 781
pinq_ws_tran3, 784
pinq_ws_posted, 719
pinterpret_item, 789
pmessage, 792
poffset_elem_ptr, 794
popen_ar_file, 796
popen_phigs, 798
popen_struct, 799
popen_ws, 800
ppolyline, 802
ppolyline3, 804

Index-5
C functions, continued

pset_choice_mode, 806
pset_colr_model, 809
pset_colr_rep, 811
pset_conf_res, 814
pset_disp_upd_st, 816
pset_edge_colr_ind, 821
pset_edge_flag, 823
pset_edge_ind, 825
pset_edge_rep, 827
pset_edge_type, 830
pset_edge_width, 833
pset_edit_mode, 835
pset_elem_ptr, 837
pset_elem_ptr_label, 839
pset_err_hand, 840
pset_err_hand_mode, 841
pset_global_tran, 843
pset_global_tran3, 845
pset_highl_filter, 847
pset_hlhsr_id, 849
pset_hlhsr_mode, 851
pset_indiv_asf, 853
pset_int_colr_ind, 857
pset_int_ind, 859
pset_int_rep, 861
pset_int_style, 864
pset_int_style_ind, 867
pset_invis_filter, 869
pset_line_colr_ind, 911
pset_line_ind, 913
pset_line_rep, 915
pset_linetype, 871
pset lineWidth, 873
pset_loc_mode, 881
pset_local_tran, 875
pset_local_tran3, 878
pset_marker_colr_ind, 918
pset_marker_ind, 920
pset_marker_rep, 922
pset_marker_size, 884
pset_marker_type, 886
pset_model_clip_ind, 888
pset_model_clip_vol, 890
pset_model_clip_vol3, 893

C functions, continued

pset_pat_ref_point, 896, 898
pset_pat_rep, 900
pset_pat_size, 902
pset_pick_filter, 904
pset_pick_id, 906
pset_pick_mode, 908
pset_rendering_colr_model, 925
pset_string_mode, 926
pset_stroke_mode, 929
pset_text_align, 932
pset_text_colr_ind, 935
pset_text_font, 937
pset_text_ind, 939
pset_text_path, 941
pset_text_prec, 943
pset_text_rep, 945
pset_val_mode, 948
pset_view_ind, 951
pset_view_rep, 953
pset_view_rep3, 957
pset_view_tran_in_pri, 962
pset_ws_vp, 964
pset_ws_vp3, 967
pset_ws_win, 970
pset_ws_win3, 973
ptext, 365
ptext3, 368
ptran_point, 371
ptran_point3, 373
ptranslate, 375
ptranslate3, 377
punpost_all_structs, 382
punpost_struct, 383
pupd_ws, 385
pwrite_item, 387

CELL ARRAY

pca—FORTRAN function, 49
pcell_array—C function, 49

CELL ARRAY 3

pca3—FORTRAN function, 52
pcell_array3—C function, 52

CHANGE STRUCTURE IDENTIFIER

pchange_struct_id—C function, 55
pcstid—FORTRAN function, 55

Index—6
CHANGE STRUCTURE IDENTIFIER AND REFERENCES
  pchange_struct_id_refs—C function, 57
  pcstir—FORTRAN function, 57

CHANGE STRUCTURE REFERENCES
  pchange_struct_refs—C function, 60
  pcstrf—FORTRAN function, 60

Choice Input Devices
  GET CHOICE, 137
  INITIALIZE CHOICE, 166
  INITIALIZE CHOICE 3, 175
  INQUIRE CHOICE DEVICE STATE, 396
  INQUIRE DEFAULT CHOICE DEVICE DATA, 428
  INQUIRE DEFAULT CHOICE DEVICE DATA 3, 433
  REQUEST CHOICE, 296
  SAMPLE CHOICE, 342
  SET CHOICE MODE, 806

CLOSE ARCHIVE FILE
  pclarf—FORTRAN function, 62
  pclose_ar_file—C function, 62

CLOSE PHIGS
  pclophg—C function, 64
  pclph—FORTRAN function, 64

CLOSE STRUCTURE
  pclose_struct—C function, 65
  pclst—FORTRAN function, 65

CLOSE WORKSTATION
  pclose_ws(ws)_id—C function, 66
  pclwk—FORTRAN function, 66

Colour
  INQUIRE COLOUR FACILITIES, 406
  INQUIRE COLOUR MODEL, 411
  INQUIRE COLOUR MODEL FACILITIES, 413
  INQUIRE COLOUR REPRESENTATION, 415
  INQUIRE LIST OF COLOUR INDICES, 614
  INQUIRE PREDEFINED COLOUR REPRESENTATION, 695
  SET COLOUR MODEL, 809
  SET COLOUR REPRESENTATION, 811
  SET EDGE COLOUR INDEX, 821
  SET INTERIOR COLOUR INDEX, 857
  SET POLYLINE COLOUR INDEX, 917
  SET POLYMARKER COLOUR INDEX, 918

  Colour, continued
    SET TEXT COLOUR INDEX, 935

Colour Model
  SET RENDERING COLOUR MODEL, 925

Common Problems
  description, 2

COMPOSE MATRIX
  pcom—FORTRAN function, 68
  pcompose_matrix—C function, 68

COMPOSE MATRIX 3
  pcom3—FORTRAN function, 70
  pcompose_matrix3—C function, 70

COMPOSE TRANSFORMATION MATRIX
  pcompose_tran_matrix—C function, 72
  pcotm—FORTRAN function, 72

COMPOSE TRANSFORMATION MATRIX 3
  pcompose_tran_matrix3—C function, 75
  pcotm3—FORTRAN function, 75

COPY ALL ELEMENTS FROM STRUCTURE
  pcelst—FORTRAN function, 78
  pcopy_allelems_struct—C function, 78

CREATE STORE, 79
  pcreate_store—C function, 79

Current Element Pointer
  INQUIRE ELEMENT POINTER, 566
  OFFSET ELEMENT POINTER, 264
  SET ELEMENT POINTER, 837
  SET ELEMENT POINTER AT LABEL, 839

D

Data Records
  PACK DATA RECORD, 278
  UNPACK DATA RECORD, 379

DELETE ALL STRUCTURES
  pdas—FORTRAN function, 80
  pdel_all structs—C function, 80

DELETE ALL STRUCTURES FROM ARCHIVE
  pdasar—FORTRAN function, 81
  pdel_all structs ar—C function, 81

DELETE ELEMENT
  pdel—FORTRAN function, 82
  pdel_elem—C function, 82

DELETE ELEMENT RANGE
  pdel_elem_range—C function, 83
DELETE ELEMENT RANGE, continued
  pdelra—FORTRAN function, 83
DELETE ELEMENTS BETWEEN LABELS
  pdel elems_labels—C function, 84
  pdel1b—FORTRAN function, 84
DELETE STORE, 86
  pdel store—C function, 86
DELETE STRUCTURE
  pdel struct—C function, 87
  pdst—FORTRAN function, 87
DELETE STRUCTURE NETWORK
  pdel struct net—C function, 89
  pdsn—FORTRAN function, 89
DELETE STRUCTURE NETWORKS FROM ARCHIVE
  pdel struct nets ar—C function, 91
  pdsn ar—FORTRAN function, 91
DELETE STRUCTURES FROM ARCHIVE
  pdel structs ar—C function, 93
  pdstar—FORTRAN function, 93
Detectability
  ADD NAMES TO SET, 22
  INQUIRE PICK DEVICE STATE, 666
  REMOVE NAMES FROM SET, 294
  SET PICK FILTER, 904
Deviations from PHIGS Standard, description, 2
Device coordinates
  INQUIRE DISPLAY SPACE SIZE, 0
  INQUIRE DISPLAY SPACE SIZE 3, 0
Display Priority
  INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED, 646
  INQUIRE POSTED STRUCTURES, 692
  POST STRUCTURE, 288
Dynamic Modifications
  INQUIRE DYNAMICS OF STRUCTURES, 497
  INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES, 499
  SET DISPLAY UPDATE STATE, 816

E
Edge Attributes
  SET EDGE COLOUR INDEX, 821
  SET EDGE FLAG, 823
Edge Attributes, continued
  SET EDGE INDEX, 825
  SET EDGETYPE, 830
  SET EDGEWIDTH SCALE FACTOR, 833
Editing
  DELETE ELEMENT, 82
  DELETE ELEMENT RANGE, 83
  DELETE ELEMENTS BETWEEN LABELS, 84
  SET DISPLAY UPDATE STATE, 816
  SET EDIT MODE, 835
  SET ELEMENT POINTER, 837
  SET ELEMENT POINTER AT LABEL, 839
Element Pointer
  INQUIRE ELEMENT POINTER, 566
  OFFSET ELEMENT POINTER, 264
  SET ELEMENT POINTER, 837
  SET ELEMENT POINTER AT LABEL, 839
ELEMENT SEARCH
  pelem_search—C function, 95
  pels—FORTRAN function, 95
Elements
  LABEL, 260
  Elements, attributes, see Attributes
  Elements, miscellaneous
    APPLICATION DATA, 32
    EXECUTE STRUCTURE, 122
  Elements, primitives, see Primitives
  Elements, transformations, see Transformation Matrix
EMERGENCY CLOSE PHIGS
  peclph—FORTRAN function, 103
  pemergency_close_phigs—C function, 103
EMPTY STRUCTURE
  pempty struct—C function, 104
  pemst—FORTRAN function, 104
ERROR HANDLING
  perhnd—FORTRAN function, 105
  perr hand—C function, 105
ERROR LOGGING
  perlog—FORTRAN function, 107
  perr_log—C function, 107
Errors
  ERROR HANDLING, 105
  ERROR LOGGING, 107
  OPEN PHIGS, 267
Errors, continued
SET ERROR HANDLING, 840
SET ERROR HANDLING MODE, 841
EVALUATE VIEW MAPPING MATRIX
peval_view_map_matrix—C function, 109
pevvm—FORTRAN function, 109
EVALUATE VIEW MAPPING MATRIX 3
peval_view_map_matrix3—C function, 112
pevvm3—FORTRAN function, 112
EVALUATE VIEW ORIENTATION MATRIX
peval_view_ori_matrix—C function, 117
pevom—FORTRAN function, 117
EVALUATE VIEW ORIENTATION MATRIX 3
peval_view_ori_matrix3—C function, 119
pevom3—FORTRAN function, 119
Event Input Mode
AWAIT EVENT, 40
FLUSH DEVICE EVENTS, 135
GET CHOICE, 137
GET LOCATOR, 140
GET PICK, 144
GET STRING, 147
GET STROKE 3, 151
GET VALUATOR, 153
INQUIRE MORE SIMULTANEOUS EVENTS, 642
EXECUTE STRUCTURE
pexec_struct—C function, 122
pexst—FORTRAN function, 122
Extensions to PHIGS Standard description, 2

F
FILL AREA
pfa—FORTRAN function, 124
pfill_area—C function, 124
FILL AREA 3
pfa3—FORTRAN function, 126
pfill_area3—C function, 126
FILL AREA SET
pfas—FORTRAN function, 129
pfill_area_set—C function, 129
FILL AREA SET 3
pfas3—FORTRAN function, 132
pfill_area_set3—C function, 132

Filled Area Interior Attributes
SET INTERIOR COLOUR INDEX, 857
SET INTERIOR INDEX, 859
SET INTERIOR STYLE, 864
SET INTERIOR STYLE INDEX, 867
SET PATTERN REFERENCE POINT, 896
SET PATTERN REFERENCE POINT AND VECTORS, 898
SET PATTERN SIZE, 902

Filled Area Primitives
FILL AREA, 124
FILL AREA 3, 126
FILL AREA SET, 129
FILL AREA SET 3, 132

FLUSH DEVICE EVENTS
pflush—FORTRAN function, 135
pflush_events—C function, 135

Fonts
SET TEXT FONT, 937
SET TEXT REPRESENTATION, 945

FORTRAN Binding Notes
description, 3

FORTRAN Data Records
PACK DATA RECORD, 278
UNPACK DATA RECORD, 379

FORTRAN functions
pads, 22
pap, 32
parast, 34
parsn, 36
parst, 38
patr, 24
patr3, 28
pbltm, 43
pbltm3, 46
pca, 49
pca3, 52
pcelst, 78
pclarf, 62
pclph, 64
pclst, 65
pclwk, 66
pcom, 68
pcotm, 70

Index—9
FORTRAN functions, continued

pcotm3, 75
pcstid, 55
pcstir, 57
pcstrf, 60
pdas, 80
pdasar, 81
pdel, 82
pdel1b, 84
pdelra, 83
pdsn, 89
pdsn, 91
pdst, 87
pdstar, 93
peclep, 103
pels, 95
pemst, 104
perhnd, 105
perlog, 107
pevmm, 109
pevmm3, 112
pevom, 117
pevom3, 119
pexst, 122
pfa, 124
pfa3, 126
pfas, 129
pfas3, 132
pflush, 135
pgtch, 137
pgtitm, 139
pgtlc, 140
pgtlc3, 142
pgtpk, 144
pgtsk, 149
pgtsk3, 151
pgtst, 147
pgtvl, 153
piitm, 259
pinch, 166
pinch3, 175
pinlc, 184
pinlc3, 191
pinpk, 198
pinpk3, 206

FORTRAN functions, continued

pinsk, 224
pinsk3, 232
pinsn, 214
pinsn3, 219
pinvl, 214
pinvl3, 250
plss, 154
plss3, 160
plb, 260
pmss, 262
poparfl, 265
pophh, 267
popst, 270
popwk, 272
posep, 264
ppl, 280
ppl3, 282
ppm, 284
ppm3, 286
ppost, 288
pprec, 278
pqanf, 300
pqarfl, 393
pqars, 395
pqcco, 423
pqcets, 426
pqcf, 406
pqchs, 396
pqchs3, 401
pqcmd, 411
pqcmdf, 413
pqcnrs, 418
pqr, 415
pqcst, 388
pqcstn, 420
pqdc, 428
pqdc3, 433
pqddus, 438
pqdlc, 440
pqdlc3, 445
pqdp, 646
pqdpk, 450
pqdpk3, 455
pqdsk, 468

Index–10
<table>
<thead>
<tr>
<th>FORTRAN functions, continued</th>
<th>FORTRAN functions, continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>pqdsk3, 473</td>
<td>pqopst, 648</td>
</tr>
<tr>
<td>pqdsp, 488</td>
<td>pqopwk, 717</td>
</tr>
<tr>
<td>pqdsp3, 491</td>
<td>pqpaf, 658</td>
</tr>
<tr>
<td>pqdst, 460</td>
<td>qgpa, 650</td>
</tr>
<tr>
<td>pqdst3, 464</td>
<td>pqpar, 660</td>
</tr>
<tr>
<td>pqdstr, 497</td>
<td>pqpcr, 695</td>
</tr>
<tr>
<td>pqdswa, 499</td>
<td>pqpde, 654</td>
</tr>
<tr>
<td>pqdus, 494</td>
<td>pqpedr, 698</td>
</tr>
<tr>
<td>pqdvl, 478</td>
<td>pqphf, 663</td>
</tr>
<tr>
<td>pqdvl3, 483</td>
<td>pqpif, 701</td>
</tr>
<tr>
<td>pqe, 614</td>
<td>pqpks, 666</td>
</tr>
<tr>
<td>pqeco, 509</td>
<td>pqpks3, 673</td>
</tr>
<tr>
<td>pqedf, 502</td>
<td>pqlf, 680</td>
</tr>
<tr>
<td>pqedm, 508</td>
<td>pqlr, 683</td>
</tr>
<tr>
<td>pqedr, 505</td>
<td>pqpmf, 686</td>
</tr>
<tr>
<td>pqe, 616</td>
<td>pqpmr, 689</td>
</tr>
<tr>
<td>pqegd3, 607</td>
<td>pqs, 692</td>
</tr>
<tr>
<td>pqegdp, 605</td>
<td>pqp, 704</td>
</tr>
<tr>
<td>pqqegse, 609</td>
<td>pqpplr, 707</td>
</tr>
<tr>
<td>pqeli, 618</td>
<td>p qppmr, 709</td>
</tr>
<tr>
<td>pqep, 566</td>
<td>pqs, 711</td>
</tr>
<tr>
<td>pqepal, 620</td>
<td>pqt, 714</td>
</tr>
<tr>
<td>pqepal, 620</td>
<td>pqsid, 742</td>
</tr>
<tr>
<td>pqepali, 622</td>
<td>pqsim, 745</td>
</tr>
<tr>
<td>pqepali, 622</td>
<td>pqsks, 731</td>
</tr>
<tr>
<td>pqerhm, 574</td>
<td>pqsks3, 736</td>
</tr>
<tr>
<td>pqets, 567</td>
<td>pqst, 742</td>
</tr>
<tr>
<td>pqetxi, 626</td>
<td>pqst, 745</td>
</tr>
<tr>
<td>pqewk, 612</td>
<td>pqsts, 721</td>
</tr>
<tr>
<td>pqewk, 612</td>
<td>pqsts3, 726</td>
</tr>
<tr>
<td>pqgdp, 575</td>
<td>pqst, 745</td>
</tr>
<tr>
<td>pqgdp3, 578</td>
<td>pqst, 745</td>
</tr>
<tr>
<td>pqgsef, 581</td>
<td>pqsys, 747</td>
</tr>
<tr>
<td>pqghlt, 583</td>
<td>pqtxf, 753</td>
</tr>
<tr>
<td>pqhrf, 586</td>
<td>pqtxr, 756</td>
</tr>
<tr>
<td>pqhrf, 586</td>
<td>pqt, 756</td>
</tr>
<tr>
<td>pqhrm, 592</td>
<td>pqt, 748</td>
</tr>
<tr>
<td>pqhrm, 592</td>
<td>pqvis, 759</td>
</tr>
<tr>
<td>pqlf, 596</td>
<td>pqvis3, 763</td>
</tr>
<tr>
<td>pqiqov, 594</td>
<td>pqvwr, 767</td>
</tr>
<tr>
<td>pqir, 599</td>
<td>pqwkr, 769</td>
</tr>
<tr>
<td>pqivfr, 602</td>
<td>pqwk, 776</td>
</tr>
<tr>
<td>pqics, 635</td>
<td>pqwkca, 772</td>
</tr>
<tr>
<td>pqics3, 635</td>
<td>pqwkcl, 774</td>
</tr>
<tr>
<td>pqili, 644</td>
<td>pqwkpo, 719</td>
</tr>
<tr>
<td>pqmcl, 640</td>
<td>pqwks1, 778</td>
</tr>
<tr>
<td>pqmcl, 640</td>
<td>pqwkst, 780</td>
</tr>
</tbody>
</table>

Index-11
<table>
<thead>
<tr>
<th>FORTRAN functions, continued</th>
<th>FORTRAN functions, continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>pqwkt, 781</td>
<td>psedr, 827</td>
</tr>
<tr>
<td>pqwkt3, 784</td>
<td>psedt, 830</td>
</tr>
<tr>
<td>prast, 318</td>
<td>psep, 837</td>
</tr>
<tr>
<td>prdltm, 290</td>
<td>pseplb, 839</td>
</tr>
<tr>
<td>prepan, 320</td>
<td>pserhm, 841</td>
</tr>
<tr>
<td>prepde, 324</td>
<td>psewsc, 833</td>
</tr>
<tr>
<td>pres, 294</td>
<td>psgmt, 843</td>
</tr>
<tr>
<td>presn, 330</td>
<td>psgmt3, 845</td>
</tr>
<tr>
<td>prest, 332</td>
<td>pshlft, 847</td>
</tr>
<tr>
<td>prmcv, 317</td>
<td>pshrid, 849</td>
</tr>
<tr>
<td>pro, 334</td>
<td>pshrhm, 851</td>
</tr>
<tr>
<td>prox, 336</td>
<td>psiasf, 853</td>
</tr>
<tr>
<td>proy, 338</td>
<td>psici, 857</td>
</tr>
<tr>
<td>proz, 340</td>
<td>psii, 859</td>
</tr>
<tr>
<td>prgch, 296</td>
<td>psir, 861</td>
</tr>
<tr>
<td>prqlc, 298</td>
<td>psis, 864</td>
</tr>
<tr>
<td>prqlc3, 301</td>
<td>psisi, 867</td>
</tr>
<tr>
<td>prqpk, 304</td>
<td>psivft, 869</td>
</tr>
<tr>
<td>prqsk, 309</td>
<td>pslc, 881</td>
</tr>
<tr>
<td>prqsk3, 312</td>
<td>pslm, 875</td>
</tr>
<tr>
<td>prqst, 307</td>
<td>psfmt, 878</td>
</tr>
<tr>
<td>prqvl, 315</td>
<td>psln, 871</td>
</tr>
<tr>
<td>prsid, 328</td>
<td>pslwsc, 873</td>
</tr>
<tr>
<td>prst, 291</td>
<td>psmch, 342</td>
</tr>
<tr>
<td>psans, 787</td>
<td>psmcli, 888</td>
</tr>
<tr>
<td>psatal, 789</td>
<td>psmcv, 890</td>
</tr>
<tr>
<td>psatch, 792</td>
<td>psmcv3, 893</td>
</tr>
<tr>
<td>psatcu, 794</td>
<td>psmk, 886</td>
</tr>
<tr>
<td>psatp, 796</td>
<td>psmksc, 884</td>
</tr>
<tr>
<td>psc, 361</td>
<td>psmlc, 344</td>
</tr>
<tr>
<td>psc3, 363</td>
<td>psmlc3, 346</td>
</tr>
<tr>
<td>psc3c3, 366</td>
<td>psmmpk, 348</td>
</tr>
<tr>
<td>psc3c, 363</td>
<td>psmmsk, 353</td>
</tr>
<tr>
<td>pschh, 800</td>
<td>psmmsk3, 356</td>
</tr>
<tr>
<td>pschm, 806</td>
<td>psmmst, 351</td>
</tr>
<tr>
<td>pschsp, 802</td>
<td>psmv1, 359</td>
</tr>
<tr>
<td>pschsp3, 804</td>
<td>pspaa, 902</td>
</tr>
<tr>
<td>pschxp, 798</td>
<td>pspar, 900</td>
</tr>
<tr>
<td>pscmd, 809</td>
<td>psparf, 896</td>
</tr>
<tr>
<td>pscnrs, 814</td>
<td>pspkft, 904</td>
</tr>
<tr>
<td>pscr, 811</td>
<td>pspkid, 906</td>
</tr>
<tr>
<td>pseco, 816</td>
<td>pspkm, 908</td>
</tr>
<tr>
<td>psecdci, 821</td>
<td>psplci, 911</td>
</tr>
<tr>
<td>psedfg, 823</td>
<td>pspli, 913</td>
</tr>
<tr>
<td>psedi, 825</td>
<td></td>
</tr>
<tr>
<td>psedm, 835</td>
<td></td>
</tr>
</tbody>
</table>

Index−12
FORTRAN functions, continued
psplr, 915
pspmci, 918
pspmi, 920
pspmr, 922
psprpv, 898
psrcm, 925
psskm, 929
psstm, 926
pstxal, 932
pstxci, 935
pstxfn, 937
psti, 939
pstdp, 941
pstdp3, 943
pstr, 945
psvlm, 948
psvtip, 962
psvwi, 951
psvwr, 953
psvwr3, 957
pswkv, 964
pswkv3, 967
pswkw, 970
pswkw3, 973
ptp, 371
ptp3, 373
ptr, 375
ptr3, 377
ptx, 365
ptx3, 368
pupast, 382
pupost, 383
purec, 379
puwk, 385
pwait, 40
pwitm, 387

G
Generalized Drawing Primitives
INQUIRE GENERALIZED DRAWING PRIMITIVES, 575
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES, 605

Generalized Drawing Primitives 3

Generalized Drawing Primitives 3, continued
INQUIRE GENERALIZED DRAWING PRIMITIVE 3, 578
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3, 607

Generalized Structure Elements
INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES, 581
INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS, 609

GET CHOICE
pget_choice—C function, 137
pgtch—FORTRAN function, 137

GET ITEM TYPE FROM METAFILE
pget_item_type—C function, 139
pgtitm—FORTRAN function, 139

GET LOCATOR
pget_loc—C function, 140
pgtlc—FORTRAN function, 140

GET LOCATOR 3
pget_loc3—C function, 142
pgtlc3—FORTRAN function, 142

GET PICK
pget_pick—C function, 144
pgtpk—FORTRAN function, 144

GET STRING
pget_string—C function, 147
pgtst—FORTRAN function, 147

GET STROKE
pget_stroke—C function, 149
pgtsk—FORTRAN function, 149

GET STROKE 3
pget_stroke3—C function, 151
pgtsk3—FORTRAN function, 151

GET VALUATOR
pget_val—C function, 153
pgtvl—FORTRAN function, 153

H
Hidden Line/Hidden Surface Removal
INQUIRE HLHSR IDENTIFIER FACILITIES, 586
INQUIRE HLHSR MODE, 592
INQUIRE HLHSR MODE FACILITIES, 589

Index–13
Hidden Line/Hidden Surface Removal, continued
SET HLHSR IDENTIFIER, 849
SET HLHSR MODE, 851
Highlighting
ADD NAMES TO SET, 22
INQUIRE HIGHLIGHTING FILTER, 583
REMOVE NAMES FROM SET, 294
SET HIGHLIGHTING FILTER, 847
HLHSR
INQUIRE HLHSR IDENTIFIER FACILITIES, 586
INQUIRE HLHSR MODE, 592
INQUIRE HLHSR MODE FACILITIES, 589
SET HLHSR IDENTIFIER, 849
SET HLHSR MODE, 851

I
Include Files, description, 2
INCREMENTAL SPATIAL SEARCH
pincr_spa_search—C function, 154
piss—FORTRAN function, 154
INCREMENTAL SPATIAL SEARCH 3
pincr_spa_search3—C function, 160
piss3—FORTRAN function, 160
INITIALIZE CHOICE
pinch—FORTRAN function, 166
pinit_choice—C function, 166
INITIALIZE CHOICE 3
pinch3—FORTRAN function, 175
pinit_choice3—C function, 175
INITIALIZE LOCATOR
pinit_loc—C function, 184
pinloc—FORTRAN function, 184
INITIALIZE LOCATOR 3
pinit_loc3—C function, 191
pinit_loc3—FORTRAN function, 191
INITIALIZE PICK
pinit_pick—C function, 198
pinpick—FORTRAN function, 198
INITIALIZE PICK 3
pinit_pick3—C function, 206
pinpick3—FORTRAN function, 206
INITIALIZE STRING
pinit_string—C function, 214
INITIALIZE STRING, continued
pinst—FORTRAN function, 214
INITIALIZE STRING 3
pinit_string3—C function, 219
pinst3—FORTRAN function, 219
INITIALIZE STROKE
pinit_stroke—C function, 224
pinsk—FORTRAN function, 224
INITIALIZE STROKE 3
pinit_stroke3—C function, 232
pinsk3—FORTRAN function, 232
INITIALIZE VALUATOR
pinit_val—C function, 241
pinvl—FORTRAN function, 241
INITIALIZE VALUATOR 3
pinit_val3—C function, 250
pinvl3—FORTRAN function, 250
Input Modes
AWAIT EVENT, 40
FLUSH DEVICE EVENTS, 135
GET CHOICE, 137
GET LOCATOR, 140
GET PICK, 144
GET STRING, 147
GET STROKE 3, 151
GET VALUATOR, 153
INQUIRE MORE SIMULTANEOUS EVENTS, 642
REQUEST CHOICE, 296
REQUEST LOCATOR 3, 301
REQUEST PICK, 304
REQUEST STRING, 307
REQUEST STROKE 3, 312
REQUEST VALUATOR, 315
SAMPLE CHOICE, 342
SAMPLE LOCATOR 3, 346
SAMPLE PICK, 348
SAMPLE STRING, 351
SAMPLE STROKE 3, 356
SAMPLE VALUATOR, 359
SET CHOICE MODE, 806
SET LOCATOR MODE, 881
SET PICK MODE, 908
SET STRING MODE, 926
SET STROKE MODE, 929

Index—14
INQUIRE CURRENT ELEMENT CONTENT, continued
  pqceco—FORTRAN function, 423
INQUIRE CURRENT ELEMENT TYPE AND SIZE
  pinq_cur_elem_type_size—C function, 426
  pqcets—FORTRAN function, 426
INQUIRE DEFAULT CHOICE DEVICE DATA
  pinq_def_choice_data—C function, 428
  pqdch—FORTRAN function, 428
INQUIRE DEFAULT CHOICE DEVICE DATA 3
  pinq_def_choice_data3—C function, 433
  pqdch3—FORTRAN function, 433
INQUIRE DEFAULT DISPLAY UPDATE STATE
  pinq_def_disp_upd_st—C function, 438
  pqddus—FORTRAN function, 438
INQUIRE DEFAULT LOCATOR DEVICE DATA
  pinq_def_loc_data—C function, 440
  pqdlc—FORTRAN function, 440
INQUIRE DEFAULT LOCATOR DEVICE DATA 3
  pinq_def_loc_data3—C function, 445
  pqdlc3—FORTRAN function, 445
INQUIRE DEFAULT PICK DEVICE DATA
  pinq_def_pick_data—C function, 450
  pqdpk—FORTRAN function, 450
INQUIRE DEFAULT PICK DEVICE DATA 3
  pinq_def_pick_data3—C function, 455
  pqdpk3—FORTRAN function, 455
INQUIRE DEFAULT STRING DEVICE DATA
  pinq_def_string_data—C function, 460
  pqdst—FORTRAN function, 460
INQUIRE DEFAULT STRING DEVICE DATA 3
  pinq_def_string_data3—C function, 464
  pqdst3—FORTRAN function, 464
INQUIRE DEFAULT STROKE DEVICE DATA
  pinq_def_stroke_data—C function, 468
  pqdsk—FORTRAN function, 468
INQUIRE DEFAULT STROKE DEVICE DATA 3
  pinq_def_stroke_data3—C function, 473
  pqdsk3—FORTRAN function, 473
INQUIRE DEFAULT VALUATOR DEVICE DATA
  pinq_def_val_data—C function, 478
  pqdv1—FORTRAN function, 478
INQUIRE DEFAULT VALUATOR DEVICE DATA 3
INQUIRE DEFAULT VALUATOR DEVICE DATA
3, continued
pinq_def_val_data3—C function, 483
pqdvl3—FORTRAN function, 483

INQUIRE DISPLAY SPACE SIZE
pinq_disp_space_size—C function, 488
pqdsp—FORTRAN function, 488

INQUIRE DISPLAY SPACE SIZE 3
pinq_display_space_size3—C function, 491
pqdsp3—FORTRAN function, 491

INQUIRE DISPLAY UPDATE STATE
pinq_disp_upd_st—C function, 494
pqdus—FORTRAN function, 494

INQUIRE DYNAMICS OF STRUCTURES
pinq_dynastructs—C function, 497
pqdstr—FORTRAN function, 497

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES
pinq_dynwsattrs—C function, 499
pqdswa—FORTRAN function, 499

INQUIRE EDGE FACILITIES
pinq_edge_facs—C function, 502
pqedf—FORTRAN function, 502

INQUIRE EDGE REPRESENTATION
pinq_edge_rep—C function, 505
pqedr—FORTRAN function, 505

INQUIRE EDIT MODE
pinq_edit_mode—C function, 508
pqedm—FORTRAN function, 508

INQUIRE ELEMENT CONTENT
pinq_elem_content—C function, 509
pqeco—FORTRAN function, 509

INQUIRE ELEMENT POINTER
pinq_elem_ptr—C function, 566
pqep—FORTRAN function, 566

INQUIRE ELEMENT TYPE AND SIZE
pinq_elem_type_size—C function, 567
pqets—FORTRAN function, 567

INQUIRE ERROR HANDLING MODE
pinq_err_hand_mode—C function, 574
pqerhm—FORTRAN function, 574

INQUIRE GENERALIZED DRAWING PRIMITIVE
pinq_gdp—C function, 575

INQUIRE GENERALIZED DRAWING PRIMITIVE, continued
pqgd—FORTRAN function, 575

INQUIRE GENERALIZED DRAWING PRIMITIVE
3
pinq_gdp3—C function, 578
pqgd3—FORTRAN function, 578

INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES
pinq_gse_facs—C function, 581
pqgsef—FORTRAN function, 581

INQUIRE HIGHLIGHTING FILTER
pinq_highl_filter—C function, 583
pqhlft—FORTRAN function, 583

INQUIRE HLHSR IDENTIFIER FACILITIES
pinq_hlhsr_id_facs—C function, 586
pqhrif—FORTRAN function, 586

INQUIRE HLHSR MODE
pinq_hlhgsr_mode—C function, 592
pqhrm—FORTRAN function, 592

INQUIRE HLHSR MODE FACILITIES
pinq_hlhgsr_mode_facs—C function, 589
pqhrmf—FORTRAN function, 589

INQUIRE INPUT QUEUE OVERFLOW
pinq_in_overf—C function, 594
pqiqov—FORTRAN function, 594

INQUIRE INTERIOR FACILITIES
pinq_int_facs—C function, 596
pqif—FORTRAN function, 596

INQUIRE INTERIOR REPRESENTATION
pinq_int_rep—C function, 599
pqir—FORTRAN function, 599

INQUIRE INVISIBILITY FILTER
pinq_invis_filter—C function, 602
pqivft—FORTRAN function, 602

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES
pinq_list_avail_gdp—C function, 605
pqegdp—FORTRAN function, 605

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3
pinq_list_avail_gdp3—C function, 607
pqegd3—FORTRAN function, 607

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 4
pinq_list_avail_gdp4—C function, 609
pqegd4—FORTRAN function, 609

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 5
pinq_list_avail_gdp5—C function, 611
pqegd5—FORTRAN function, 611

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 6
pinq_list_avail_gdp6—C function, 613
pqegd6—FORTRAN function, 613

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 7
pinq_list_avail_gdp7—C function, 615
pqegd7—FORTRAN function, 615

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 8
pinq_list_avail_gdp8—C function, 617
pqegd8—FORTRAN function, 617

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 9
pinq_list_avail_gdp9—C function, 619
pqegd9—FORTRAN function, 619

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 10
pinq_list_avail_gdp10—C function, 621
pqegd10—FORTRAN function, 621

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 11
pinq_list_avail_gdp11—C function, 623
pqegd11—FORTRAN function, 623

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 12
pinq_list_avail_gdp12—C function, 625
pqegd12—FORTRAN function, 625

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 13
pinq_list_avail_gdp13—C function, 627
pqegd13—FORTRAN function, 627

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 14
pinq_list_avail_gdp14—C function, 629
pqegd14—FORTRAN function, 629

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 15
pinq_list_avail_gdp15—C function, 631
pqegd15—FORTRAN function, 631

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 16
pinq_list_avail_gdp16—C function, 633
pqegd16—FORTRAN function, 633

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 17
pinq_list_avail_gdp17—C function, 635
pqegd17—FORTRAN function, 635

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 18
pinq_list_avail_gdp18—C function, 637
pqegd18—FORTRAN function, 637

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 19
pinq_list_avail_gdp19—C function, 639
pqegd19—FORTRAN function, 639

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 20
pinq_list_avail_gdp20—C function, 641
pqegd20—FORTRAN function, 641

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 21
pinq_list_avail_gdp21—C function, 643
pqegd21—FORTRAN function, 643

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 22
pinq_list_avail_gdp22—C function, 645
pqegd22—FORTRAN function, 645

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 23
pinq_list_avail_gdp23—C function, 647
pqegd23—FORTRAN function, 647

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 24
pinq_list_avail_gdp24—C function, 649
pqegd24—FORTRAN function, 649

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 25
pinq_list_avail_gdp25—C function, 651
pqegd25—FORTRAN function, 651

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 26
pinq_list_avail_gdp26—C function, 653
pqegd26—FORTRAN function, 653

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 27
pinq_list_avail_gdp27—C function, 655
pqegd27—FORTRAN function, 655

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 28
pinq_list_avail_gdp28—C function, 657
pqegd28—FORTRAN function, 657

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 29
pinq_list_avail_gdp29—C function, 659
pqegd29—FORTRAN function, 659

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 30
pinq_list_avail_gdp30—C function, 661
pqegd30—FORTRAN function, 661

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 31
pinq_list_avail_gdp31—C function, 663
pqegd31—FORTRAN function, 663

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 32
pinq_list_avail_gdp32—C function, 665
pqegd32—FORTRAN function, 665

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 33
pinq_list_avail_gdp33—C function, 667
pqegd33—FORTRAN function, 667

INDEX
STRUCTURE ELEMENTS
pinq_list_avail_gse—C function, 609
pqegse—FORTRAN function, 609
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
pinq_list_avail_ws_types—C function, 612
pqewk—FORTRAN function, 612
INQUIRE LIST OF COLOUR INDICES
pinq_list_colr_inds—C function, 614
pqeci—FORTRAN function, 614
INQUIRE LIST OF EDGE INDICES
pinq_list_edge_inds—C function, 616
pqeedi—FORTRAN function, 616
INQUIRE LIST OF INTERIOR INDICES
pinq_list_int_inds—C function, 618
pqeii—FORTRAN function, 618
INQUIRE LIST OF PATTERN INDICES
pinq_list_pat_inds—C function, 620
pqepai—FORTRAN function, 620
INQUIRE LIST OF POLYLINE INDICES
pinq_list_line_inds—C function, 622
pqepli—FORTRAN function, 622
INQUIRE LIST OF POLYMARKER INDICES
pinq_list_marker inds—C function, 624
pqepmi—FORTRAN function, 624
INQUIRE LIST OF TEXT INDICES
pinq_list_text_inds—C function, 626
pqetxi—FORTRAN function, 626
INQUIRE LIST OF VIEW INDICES
pinq_list_view_inds—C function, 628
pqevwi—FORTRAN function, 628
INQUIRE LOCATOR DEVICE STATE
pinq_loc_st—C function, 630
pqlcs—FORTRAN function, 630
INQUIRE LOCATOR DEVICE STATE 3
pinq_loc_st3—C function, 635
pqlcs3—FORTRAN function, 635
INQUIRE MODELLING CLIPPING FACILITIES
pinq_model_clip_facs—C function, 640
pqmclf—FORTRAN function, 640
INQUIRE MORE SIMULTANEOUS EVENTS
pinq_more_simult_events—C function, 642
pqsim—FORTRAN function, 642
INQUIRE MORE SIMULTANEOUS EVENTS, continued
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES
pinq_num_avail_in—C function, 644
pqli—FORTRAN function, 644
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED
pinq_num_disp_pris—C function, 646
pqdp—FORTRAN function, 646
INQUIRE OPEN STRUCTURE
pinq_open_struct—C function, 648
pqopst—FORTRAN function, 648
INQUIRE PATHS TO ANCESTORS
pinq_paths_ances—C function, 650
pqpan—FORTRAN function, 650
INQUIRE PATHS TO DESCENDANTS
pinq_paths_descs—C function, 654
pqpde—FORTRAN function, 654
INQUIRE PATTERN FACILITIES
pinq_pat_fac—C function, 658
pqpaf—FORTRAN function, 658
INQUIRE PATTERN REPRESENTATION
pinq_pat_rep—C function, 660
pqpar—FORTRAN function, 660
INQUIRE PHIGS FACILITIES
pinq_phigs_facs—C function, 663
pqphf—FORTRAN function, 663
INQUIRE PICK DEVICE STATE
pinq_pick_st—C function, 666
pqpks—FORTRAN function, 666
INQUIRE PICK DEVICE STATE 3
pinq_pick_st3—C function, 673
pqpks3—FORTRAN function, 673
INQUIRE POLYLINE FACILITIES
pinq_line_facs—C function, 680
pqplf—FORTRAN function, 680
INQUIRE POLYLINE REPRESENTATION
pinq_line_rep—C function, 683
pqplr—FORTRAN function, 683
INQUIRE POLYMARKER FACILITIES
pinq_marker_fac—C function, 686
pqpmf—FORTRAN function, 686
INQUIRE POLYMARKER REPRESENTATION
  pinq_marker_rep—C function, 689
  pqpm—FORTRAN function, 689

INQUIRE POSTED STRUCTURES
  pinq_posted_structs—C function, 692
  pqpost—FORTRAN function, 692

INQUIRE PREDEFINED COLOUR REPRESENTATION
  pinq_pred_colr_rep—C function, 695
  pqpcr—FORTRAN function, 695

INQUIRE PREDEFINED EDGE REPRESENTATION
  pinq_pred_edge_rep—C function, 698
  pqpedr—FORTRAN function, 698

INQUIRE PREDEFINED INTERIOR REPRESENTATION
  pinq_pred_int_rep—C function, 701
  pqpir—FORTRAN function, 701

INQUIRE PREDEFINED PATTERN REPRESENTATION
  pinq_pred_pat_rep—C function, 704
  pqppar—FORTRAN function, 704

INQUIRE PREDEFINED POLYLINE REPRESENTATION
  pinq_pred_line_rep—C function, 707
  pqplmr—FORTRAN function, 707

INQUIRE PREDEFINED POLYMARKER REPRESENTATION
  pinq_pred_marker_rep—C function, 709
  pqppmr—FORTRAN function, 709

INQUIRE PREDEFINED TEXT REPRESENTATION
  pinq_pred_text_rep—C function, 711
  pqptxr—FORTRAN function, 711

INQUIRE PREDEFINED VIEW REPRESENTATION, See Appendix C of Getting Started with SunPHIGS, See Workstation Transformation
  pinq_pred_view_rep—C function, 714
  pqpvwr—FORTRAN function, 714

INQUIRE SET OF OPEN WORKSTATIONS
  pinq_open_wss—C function, 717
  pqopwk—FORTRAN function, 717

INQUIRE SET OF WORKSTATIONS TO WHICH POSTED
  pinq_wss_posted—C function, 719
  pqwkpo—FORTRAN function, 719

INQUIRE STRING DEVICE STATE
  pinq_string_st—C function, 721
  pqsts—FORTRAN function, 721

INQUIRE STRING DEVICE STATE 3
  pinq_string_st3—C function, 726
  pqsts3—FORTRAN function, 726

INQUIRE STROKE DEVICE STATE
  pinq_stroke_st—C function, 731
  pqsk—FORTRAN function, 731

INQUIRE STROKE DEVICE STATE 3
  pinq_stroke_st3—C function, 736
  pqsk3—FORTRAN function, 736

INQUIRE STRUCTURE IDENTIFIERS
  pinq_struct_ids—C function, 742
  pqsid—FORTRAN function, 742

INQUIRE STRUCTURE STATE VALUE
  pinq_struct_st—C function, 744
  pqstrs—FORTRAN function, 744

INQUIRE STRUCTURE STATUS
  pinq_struct_status—C function, 745
  pqstst—FORTRAN function, 745

INQUIRE SYSTEM STATE VALUE
  pinq_sys_st—C function, 747
  pqsys—FORTRAN function, 747

INQUIRE TEXT EXTENT
  pinq_text_extent—C function, 748
  pqtxx—FORTRAN function, 748

INQUIRE TEXT FACILITIES
  pinq_text_facs—C function, 753
  pqtxf—FORTRAN function, 753

INQUIRE TEXT REPRESENTATION
  pinq_text_rep—C function, 756
  pqtxr—FORTRAN function, 756

INQUIRE VALUATOR DEVICE STATE
  pinq_val_st—C function, 759
  pqvls—FORTRAN function, 759

INQUIRE VALUATOR DEVICE STATE 3
  pinq_val_st3—C function, 763
  pqvls3—FORTRAN function, 763

INQUIRE VIEW FACILITIES
  pinq_view_facs—C function, 767
  pqvwf—FORTRAN function, 767

INQUIRE VIEW REPRESENTATION
  pinq_view_rep—C function, 769
INQUIRE VIEW REPRESENTATION, continued
pqvwr—FORTRAN function, 769
INQUIRE WORKSTATION CATEGORY
pinq_ws_cat—C function, 772
pqwkca—FORTRAN function, 772
INQUIRE WORKSTATION CLASSIFICATION
pinq_ws_class—C function, 772
pqwkcl—FORTRAN function, 774
INQUIRE WORKSTATION CONNECTION AND
TYPE
pinq_ws_conn_type—C function, 776
pqwkc—FORTRAN function, 776
INQUIRE WORKSTATION STATE TABLE
LENGTHS
pinq_ws_st_table—C function, 778
pqwksl—FORTRAN function, 778
INQUIRE WORKSTATION STATE VALUE
pinq_ws_st—C function, 780
pqwkst—FORTRAN function, 780
INQUIRE WORKSTATION TRANSFORMATION
pinq_ws_tran—C function, 781
pqwkt—FORTRAN function, 781
INQUIRE WORKSTATION TRANSFORMATION 3
pinq_ws_tran3—C function, 784
pqwkt3—FORTRAN function, 784
Insert Mode
SET EDIT MODE, 835
INTERPRET ITEM
piitm—FORTRAN function, 259
pinterpret_item—C function, 259
Introduction to SunPHIGS, 1
Introduction, INTRO PHIGS, 1
Invisibility
ADD NAMES TO SET, 22
INQUIRE INVISIBILITY FILTER, 602
REMOVE NAMES FROM SET, 294
SET INVISIBILITY FILTER, 869

L
LABEL, See Editing
plabel—C function, 260
plb—FORTRAN function, 260
Limitations, description, 2
Line Attributes

Line Attributes, continued
SET LINETYPE, 871
SET LINEWIDTH SCALE FACTOR, 873
SET POLYLINE COLOUR INDEX, 911
SET POLYLINE INDEX, 913
Line Primitives
POLYLINE, 280
POLYLINE 3, 282
Locator Input Devices
GET LOCATOR, 140
GET LOCATOR 3, 142
INITIALIZE LOCATOR, 184
INITIALIZE LOCATOR 3, 191
INQUIRE DEFAULT LOCATOR DEVICE
DATA, 440
INQUIRE DEFAULT LOCATOR DEVICE
DATA 3, 445
INQUIRE LOCATOR DEVICE STATE, 630
INQUIRE LOCATOR DEVICE STATE 3, 635
REQUEST LOCATOR, 298
REQUEST LOCATOR 3, 301
SAMPLE LOCATOR, 344
SAMPLE LOCATOR 3, 346
SET LOCATOR MODE, 881

M
Marker Attributes
SET MARKER SIZE SCALE FACTOR, 884
SET MARKER TYPE, 886
SET POLYMARKER COLOUR INDEX, 918
SET POLYMARKER INDEX, 920
Marker Primitives
POLYMARKER, 284
POLYMARKER 3, 286
MESSAGE
pmessage—C function, 262
pmsg—FORTRAN function, 262
Metafiles
GET ITEM TYPE FROM METAFILE, 139
INTERPRET ITEM, 259
OPEN WORKSTATION, 272
READ ITEM FROM METAFILE, 290
WRITE ITEM TO METAFILE, 387
Modelling and Transformation Attributes
RESTORE MODELLING CLIPPING VOLUME,
Modelling and Transformation Attributes, continued

- SET GLOBAL TRANSFORMATION, 843
- SET GLOBAL TRANSFORMATION 3, 845
- SET LOCAL TRANSFORMATION, 875
- SET LOCAL TRANSFORMATION 3, 878
- SET MODELLING CLIPPING INDICATOR, 888
- SET MODELLING CLIPPING VOLUME, 890
- SET MODELLING CLIPPING VOLUME 3, 893
- SET VIEW INDEX, 951

Modelling Clipping

- INQUIRE MODELLING CLIPPING FACILITIES, 640
- RESTORE MODELLING CLIPPING VOLUME, 317
- SET MODELLING CLIPPING INDICATOR, 888
- SET MODELLING CLIPPING VOLUME, 890
- SET MODELLING CLIPPING VOLUME 3, 893

Modifications

- SET DISPLAY UPDATE STATE, 816

Namesets

- ADD NAMES TO SET, 22
- INCREMENTAL SPATIAL SEARCH, 154
- INCREMENTAL SPATIAL SEARCH 3, 160
- REMOVE NAMES FROM SET, 294
- SET HIGHLIGHTING FILTER, 847
- SET INVISIBILITY FILTER, 869
- SET PICK FILTER, 904

OFFSET ELEMENT POINTER

- offset_elem_ptr—C function, 264
- posep—FORTRAN function, 264

OPEN ARCHIVE FILE

- popar—FORTRAN function, 265
- popen_ar_file—C function, 265

OPEN PHIGS

- popen_phigs—C function, 267
- popph—FORTRAN function, 267

OPEN STRUCTURE

- popen_struct—C function, 270

OPEN STRUCTURE, continued

- popst—FORTRAN function, 270

OPEN WORKSTATION

- popen_ws—C function, 272
- popwk—FORTRAN function, 272

Operating States

- EMERGENCY CLOSE PHIGS, 103
- INQUIRE ARCHIVE STATE VALUE, 395
- INQUIRE OPEN STRUCTURE, 648
- INQUIRE STRUCTURE STATE VALUE, 744
- INQUIRE SYSTEM STATE VALUE, 747
- INQUIRE WORKSTATION STATE VALUE, 780

P

PACK DATA RECORD

- pprec—FORTRAN function, 278
- padd_names_set—C function, 22
- pads—FORTRAN function, 22
- panno_text_re1—C function, 24
- panno_text_re13—C function, 28
- pap—FORTRAN function, 32
- pappl_data—C function, 32
- par_all_structs—C function, 34
- par_struct Nets—C function, 36
- par_structs—C function, 38
- parast—FORTRAN function, 34
- parsn—FORTRAN function, 36
- parst—FORTRAN function, 38
- patr—FORTRAN function, 24
- patr3—FORTRAN function, 28

Pattern Fill

- INQUIRE LIST OF PATTERN INDICES, 620
- INQUIRE PATTERN FACILITIES, 658
- INQUIRE PATTERN REPRESENTATION, 660
- INQUIRE PREDEFINED PATTERN REPRESENTATION, 704
- SET PATTERN REFERENCE POINT, 896
- SET PATTERN REFERENCE POINT AND VECTORS, 898
- SET PATTERN REPRESENTATION, 900
- SET PATTERN SIZE, 902

Patterned Lines

- SET EDGETYPE, 830

Index—20
Patterned Lines, continued

SET LINETYPE, 871

pawait_event—C function, 40
pbltm—FORTRAN function, 43
pbltm3—FORTRAN function, 46
pbuild_tran_matrix—C function, 43
pbuild_trans_matrix3—C function, 46

PBUTTON C Macros

INITIALIZE CHOICE, 166

pca—FORTRAN function, 49
pca3—FORTRAN function, 52
pcell_array—C function, 49
pcell_array3—C function, 52
pcelst—FORTRAN function, 78
pchange_struct_id—C function, 55
pchange_struct_id_refs—C function, 57
pchange_struct_refs—C function, 60
pclarf—FORTRAN function, 62
pclose_ar_file—C function, 62
pclose_phigs—C function, 64
pclose_struct—C function, 65
pclose_ws(ws)_id—C function, 66
pciph—FORTRAN function, 64
pclst—FORTRAN function, 65
pclwk—FORTRAN function, 66
pcom—FORTRAN function, 68
pcom3—FORTRAN function, 70
pcompose_matrix—C function, 68
pcompose_matrix3—C function, 70
pcompose_trans_matrix—C function, 72
pcompose_trans_matrix3—C function, 75
pcopy_all_elems_struct—C function, 78
pcotm—FORTRAN function, 72
pcotm3—FORTRAN function, 75
pcreate_store—C function, 79
pcsid—FORTRAN function, 55
pcstir—FORTRAN function, 57
pcstrf—FORTRAN function, 60
pdas—FORTRAN function, 80
pdasar—FORTRAN function, 81
pdel—FORTRAN function, 82
pdel_all_structs—C function, 80
pdel_all_structs_ar—C function, 81
pdel_all—C function, 82
pdel_elem—C function, 82
pdel_elem_range—C function, 83
pdel_elems_labels—C function, 84
pdel_store—C function, 86
pdel_struct—C function, 87
pdel_struct_net—C function, 89
pdel_struct_nets_ar—C function, 91
pdel_structs_ar—C function, 93
pdel1b—FORTRAN function, 84
pdelra—FORTRAN function, 83
pdsn—FORTRAN function, 89
pdsn—FORTRAN function, 91
pdsst—FORTRAN function, 87
pdsr—FORTRAN function, 93
peclph—FORTRAN function, 103
peclst—FORTRAN function, 104
pelem_search—C function, 95
pels—FORTRAN function, 95
pemergency_close_phigs—C function, 103
pempty_struct—C function, 104
pemst—FORTRAN function, 104
perhnd—FORTRAN function, 105
perlog—FORTRAN function, 107
perr_hand—C function, 105
perr_log—C function, 107
pemal_view_map_matrix—C function, 109
pemal_view_map_matrix3—C function, 112
pemal_view_ori_matrix—C function, 117
pemal_view_ori_matrix3—C function, 119
pemal_view_map_matrix—C function, 109
pemal_view_map_matrix3—C function, 112
pemal_view_map_matrix—C function, 117
pemal_view_ori_matrix3—C function, 119
pemal—FORTRAN function, 112
pemal—FORTRAN function, 117
pemal—FORTRAN function, 119
pemal—FORTRAN function, 122
pexst—FORTRAN function, 122
pfa—FORTRAN function, 124
pfas—FORTRAN function, 126
pfas—FORTRAN function, 129
pfas3—FORTRAN function, 132
PFCN_KEY C Macros

INITIALIZE CHOICE 3, 175

pfcnkey FORTRAN Functions

Index—21
pfcnkey FORTRAN Functions, continued

INITIALIZE CHOICE, 166
INITIALIZE CHOICE 3, 175
pfill_area—C function, 124
pfill_area_set—C function, 129
pfill_area_set3—C function, 132
pfill_area3—C function, 126
pflush—FORTRAN function, 135
pflush_events—C function, 135
pget_choice—C function, 137
pget_item_type—C function, 139
pget_loc—C function, 140
pget_loc3—C function, 142
pget_pick—C function, 144
pget_string—C function, 147
pget_stroke—C function, 149
pget_stroke3—C function, 151
pget_val—C function, 153
pgtch—FORTRAN function, 137
pgtitm—FORTRAN function, 139
pgtlc—FORTRAN function, 140
pgtlc3—FORTRAN function, 142
pgtpk—FORTRAN function, 144
pgtsk—FORTRAN function, 149
pgtsk3—FORTRAN function, 151
pgtst—FORTRAN function, 147
pgtvl—FORTRAN function, 153

Pick Input Devices
GET PICK, 144
INITIALIZE PICK, 198
INQUIRE DEFAULT PICK DEVICE DATA, 450
INQUIRE DEFAULT PICK DEVICE DATA 3, 455
INQUIRE PICK DEVICE STATE, 666
INQUIRE PICK DEVICE STATE 3, 673
REQUEST PICK, 304
SAMPLE PICK, 348
SET PICK FILTER, 904
SET PICK IDENTIFIER, 906
SET PICK MODE, 908
piitm—FORTRAN function, 259
pinch—FORTRAN function, 166
pinch3—FORTRAN function, 175

pincr_spa_search—C function, 154
pincr_spa_search3—C function, 160
pinit_choice—C function, 166
pinit_choice3—C function, 175
pinit_loc—C function, 184
pinit_loc3—C function, 191
pinit_pick—C function, 198
pinit_pick3—C function, 206
pinit_string—C function, 214
pinit_string3—C function, 219
pinit_stroke—C function, 224
pinit_stroke3—C function, 232
pinit_val1—C function, 241
pinit_val3—C function, 250
pinlc—FORTRAN function, 184
pinlc3—FORTRAN function, 191
pinpk—FORTRAN function, 198
pinpk3—FORTRAN function, 206
pinq_all_conf_structs—C function, 388
pinq_anno_facs—C function, 390
pinq_ar_files—C function, 393
pinq_ar_st—C function, 395
pinq_choice_st—C function, 396
pinq_choice_st3—C function, 401
pinq_colr_facs—C function, 406
pinq_colr_map_method_st—C function, 409
pinq_colr_model—C function, 411
pinq_colr_model_facs—C function, 413
pinq_colr_rep—C function, 415
pinq_conf_res—C function, 418
pinq_conf_structs_net—C function, 420
pinq_cur_elem_content—C function, 423
pinq_cur_elem_type_size—C function, 426
pinq_def_choice_data—C function, 428
pinq_def_choice_data3—C function, 433
pinq_def_disp_upd_st—C function, 438
pinq_def_loc_data—C function, 440
pinq_def_loc_data3—C function, 445
pinq_def_pick_data—C function, 450
pinq_def_pick_data3—C function, 455
pinq_def_string_data—C function, 460
pinq_def_string_data3—C function, 464
pinq_def_stroke_data—C function, 468
pinq_def_stroke_data3—C function, 473
pinq_def_val_data—C function, 483
pinq_disp_space_size—C function, 488
pinq_dispupd_st—C function, 494
pinq_display_space_size3—C function, 491
pinq_dyns_structs—C function, 497
pinq_dyns_ws_attrs—C function, 499
pinq_edge_facs—C function, 502
pinq_edge_rep—C function, 505
pinq_edit_mode—C function, 508
pinq_elem_content—C function, 509
pinq_elem_ptr—C function, 566
pinq_elem_type_size—C function, 567
pinq_err_hand_mode—C function, 574
pinq_gdp—C function, 575
pinq_gdp3—C function, 578
pinq_gse_facs—C function, 581
pinq_highl_filter—C function, 583
pinq_hllsr_id_facs—C function, 586
pinq_hllsr_mode—C function, 592
pinq_hllsr_mode_facs—C function, 589
pinq_in_overf—C function, 594
pinq_int_facs—C function, 596
pinq_int_rep—C function, 599
pinq_invis_filter—C function, 602
pinq_line_facs—C function, 680
pinq_line_rep—C function, 683
pinq_list_avail_gdp—C function, 605
pinq_list_avail_gdp3—C function, 607
pinq_list_avail_gse—C function, 609
pinq_list_avail_ws_types—C function, 612
pinq_list_colr_inds—C function, 614
pinq_list_edge_inds—C function, 616
pinq_list_int_inds—C function, 618
pinq_list_line_inds—C function, 622
pinq_list_marker_inds—C function, 624
pinq_list_pat_inds—C function, 620
pinq_list_text_inds—C function, 626
pinq_list_view_inds—C function, 628
pinq_loc_st—C function, 630
pinq_loc_st3—C function, 635
pinq_marker_facs—C function, 686
pinq_marker_rep—C function, 689
pinq_model_clip_facs—C function, 640
pinq_more_simult_events—C function, 642
pinq_num_avail_in—C function, 644
pinq_num_disp_pris—C function, 646
pinq_open_struct—C function, 648
pinq_open_wss—C function, 717
pinq_pat_fac—C function, 658
pinq_pat_rep—C function, 660
pinq_paths_ances—C function, 650
pinq_paths_descs—C function, 654
pinq_phigs_facs—C function, 663
pinq_pick_st—C function, 666
pinq_pick_st3—C function, 673
pinq_posted structs—C function, 692
pinq_pred_colr_rep—C function, 695
pinq_pred_edge_rep—C function, 698
pinq_pred_int_rep—C function, 701
pinq_pred_line_rep—C function, 707
pinq_pred_marker_rep—C function, 709
pinq_pred_pat_rep—C function, 704
pinq_pred_text_rep—C function, 711
pinq_pred_view_rep—C function, 714
pinq_phrase_st—C function, 721
pinq_phrase_st3—C function, 726
pinq_stroke_st—C function, 731
pinq_stroke_st3—C function, 736
pinq_struct_ids—C function, 742
pinq_struct_status—C function, 744
pinq_struct_status—C function, 745
pinq_sys_st—C function, 747
pinq_text extent—C function, 748
pinq_text_facs—C function, 753
pinq_text_rep—C function, 756
pinq_val_st—C function, 759
pinq_val_st3—C function, 763
pinq_view_facs—C function, 767
pinq_view_rep—C function, 769
pinq_ws_cat—C function, 772
pinq_ws_class—C function, 774
pinq_ws_conn_type—C function, 776
pinq_ws_st—C function, 780
pinq_ws_st_table—C function, 778
pinq_ws_tran—C function, 781
pinq_ws_tran3—C function, 784
pinq_wss_posted—C function, 719
pinski—FORTRAN function, 224
pinsk3—FORTRAN function, 232
pinst—FORTRAN function, 214
pinst3—FORTRAN function, 219
interpret_item—C function, 259
pinv1—FORTRAN function, 241
pinv13—FORTRAN function, 250
piss—FORTRAN function, 154
piss3—FORTRAN function, 160
plabel—C function, 260
plb—FORTRAN function, 260
pmessage—C function, 262
pmsg—FORTRAN function, 262
poffset_elem_ptr—C function, 264
Polygons, Filled Area Primitives
  FILL AREA, 124
  FILL AREA 3, 126
  FILL AREA SET, 129
  FILL AREA SET 3, 132
POLYLINE
  ppl—FORTRAN function, 280
  ppolyline—C function, 280
POLYLINE 3
  ppl3—FORTRAN function, 282
  ppolyline3—C function, 282
POLYMARKER
  ppm—FORTRAN function, 284
  ppolymarker—C function, 284
POLYMARKER 3
  ppm3—FORTRAN function, 286
  ppolymarker3—C function, 286
poparf—FORTRAN function, 265
popen_ar_file—C function, 265
popen_phigs—C function, 267
popen_struct—C function, 270
popen_ws—C function, 272
popfh—FORTRAN function, 267
popst—FORTRAN function, 270
popwk—FORTRAN function, 272
posep—FORTRAN function, 264
POST STRUCTURE
  ppost—FORTRAN function, 288
  ppost_struct—C function, 288
pl1—FORTRAN function, 280
pl13—FORTRAN function, 282
plm—FORTRAN function, 284
plm3—FORTRAN function, 286
plpolyline—C function, 280
plpolyline3—C function, 282
plpolymarker—C function, 284
plpolymarker3—C function, 286
plpost—FORTRAN function, 288
plpost_struct—C function, 288
pprec—FORTRAN function, 278
pqanf—FORTRAN function, 390
pqarf—FORTRAN function, 393
pqars—FORTRAN function, 395
pqceco—FORTRAN function, 423
pqcets—FORTRAN function, 426
pqcf—FORTRAN function, 406
pqchs—FORTRAN function, 396
pqchs3—FORTRAN function, 401
pqcmd—FORTRAN function, 411
pqcmdf—FORTRAN function, 413
pqcnrs—FORTRAN function, 418
pqcr—FORTRAN function, 415
pqcst—FORTRAN function, 388
pqcstn—FORTRAN function, 420
pqdch—FORTRAN function, 428
pqdch3—FORTRAN function, 433
pqddus—FORTRAN function, 438
pqdlc—FORTRAN function, 440
pqdlc3—FORTRAN function, 445
pqdp—FORTRAN function, 646
pqdpk—FORTRAN function, 450
pqdpk3—FORTRAN function, 455
pqdsk—FORTRAN function, 468
pqdsk3—FORTRAN function, 473
pqdsp—FORTRAN function, 488

Index—24
pqdsp3—FORTRAN function, 491
pqdst—FORTRAN function, 460
pqdst3—FORTRAN function, 464
pqdstr—FORTRAN function, 497
pqdswa—FORTRAN function, 499
pqdus—FORTRAN function, 494
pqdvl—FORTRAN function, 478
pqdvl3—FORTRAN function, 483
pqeci—FORTRAN function, 614
pqeco—FORTRAN function, 509
pqedf—FORTRAN function, 502
pqedm—FORTRAN function, 508
pqedr—FORTRAN function, 505
pqedi—FORTRAN function, 616
pqeg3—FORTRAN function, 607
pqegdp—FORTRAN function, 605
pqegse—FORTRAN function, 609
pqeii—FORTRAN function, 618
pqep—FORTRAN function, 566
pqepai—FORTRAN function, 620
pqepli—FORTRAN function, 622
pqepmi—FORTRAN function, 624
pqerhm—FORTRAN function, 574
pqets—FORTRAN function, 567
pqetxi—FORTRAN function, 626
pqevwi—FORTRAN function, 628
pqewk—FORTRAN function, 612
pqgdp—FORTRAN function, 575
pqgdp3—FORTRAN function, 578
pqgsef—FORTRAN function, 581
pqh1ft—FORTRAN function, 583
pqhrif—FORTRAN function, 586
pqhrm—FORTRAN function, 592
pqhrmf—FORTRAN function, 589
pqif—FORTRAN function, 596
pqiqov—FORTRAN function, 594
pqir—FORTRAN function, 599
pqivf—FORTRAN function, 602
pqlcs—FORTRAN function, 630
pqlcs3—FORTRAN function, 635
pqli—FORTRAN function, 644
pqmclf—FORTRAN function, 640
pqopst—FORTRAN function, 648
pqopwk—FORTRAN function, 717
pqpa—FORTRAN function, 658
pqpar—FORTRAN function, 650
pqpar—FORTRAN function, 660
pqcr—FORTRAN function, 695
pqd—FORTRAN function, 654
pqedr—FORTRAN function, 698
pqhp—FORTRAN function, 663
pqpir—FORTRAN function, 701
pqks—FORTRAN function, 666
pqks3—FORTRAN function, 673
pqplf—FORTRAN function, 680
pqplr—FORTRAN function, 683
pqpmf—FORTRAN function, 686
pqpmr—FORTRAN function, 689
pqpost—FORTRAN function, 692
pqpar—FORTRAN function, 704
pqplr—FORTRAN function, 707
pqppm—FORTRAN function, 709
pqpt—FORTRAN function, 711
pqpvwr—FORTRAN function, 714
pqsys—FORTRAN function, 712
pqsys—FORTRAN function, 742
pqsim—FORTRAN function, 642
pqsk—FORTRAN function, 731
pqsk—FORTRAN function, 736
pqstrs—FORTRAN function, 744
pqsts—FORTRAN function, 721
pqsts—FORTRAN function, 726
pqstst—FORTRAN function, 745
pqsys—FORTRAN function, 747
pqtf—FORTRAN function, 753
pqtx—FORTRAN function, 756
pqtx—FORTRAN function, 748
pqvls—FORTRAN function, 759
pqvls3—FORTRAN function, 763
pqw—FORTRAN function, 767
pqwvr—FORTRAN function, 769
pqwic—FORTRAN function, 776
pqwkc—FORTRAN function, 772
pqwkc—FORTRAN function, 774
pqwkpo—FORTRAN function, 719
pqwks1—FORTRAN function, 778
pqwkst—FORTRAN function, 780
pqwkt—FORTRAN function, 781
pqwkt3—FORTRAN function, 784
prast—FORTRAN function, 318
prdtm—FORTRAN function, 290
pread_item—C function, 290
predraw_all_structs—C function, 291
premove_names_set—C function, 294
prepan—FORTRAN function, 320
prepde—FORTRAN function, 324
preq_choice—C function, 296
preq_loc—C function, 298
preq_loc3—C function, 301
preq_pick—C function, 304
preq_string—C function, 307
preq_stroke—C function, 309
preq_stroke3—C function, 312
preq_val—C function, 315
pres—FORTRAN function, 294
presn—FORTRAN function, 330
prest—FORTRAN function, 332
pret_all_structs_clip_vol—C function, 317
pret_paths_ances—C function, 318
pret_paths_desc—C function, 324
pret_struct_ids—C function, 328
pret_struct_nets—C function, 330
pret Structures—C function, 332
Primitives, Filled Area Primitives
FILL AREA, 124
FILL AREA 3, 126
FILL AREA SET, 129
FILL AREA SET 3, 132
Primitives, Line Primitives
POLYLINE, 280
POLYLINE 3, 282
Primitives, Marker Primitives
POLYMARKER, 284
POLYMARKER 3, 286
Primitives, Special Primitives
CELL ARRAY, 49
Primitives, Special Primitives, continued
CELL ARRAY 3, 52
Primitives, Text Primitives
ANNOTATION TEXT RELATIVE, 24
ANNOTATION TEXT RELATIVE 3, 28
TEXT, 365
TEXT 3, 368
Priority, Display Priority
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED, 646
INQUIRE POSTED STRUCTURES, 692
POST STRUCTURE, 288
prmcv—FORTRAN function, 317
pro—FORTRAN function, 334
protate—C function, 334
protate_x—C function, 336
protate_y—C function, 338
protate_z—C function, 340
prox—FORTRAN function, 336
proy—FORTRAN function, 338
proz—FORTRAN function, 340
prqch—FORTRAN function, 296
prqlc—FORTRAN function, 298
prqlc3—FORTRAN function, 301
prqpk—FORTRAN function, 304
prqsk—FORTRAN function, 309
prqsk3—FORTRAN function, 312
prqst—FORTRAN function, 307
prqvl—FORTRAN function, 315
prsid—FORTRAN function, 328
prst—FORTRAN function, 291
psample_choice—C function, 342
psample_loc—C function, 344
psample_loc3—C function, 346
psample_pick—C function, 348
psample_string—C function, 351
psample_stroke—C function, 353
psample_stroke3—C function, 356
psample_val—C function, 359
psans—FORTRAN function, 787
psatal—FORTRAN function, 789
psatch—FORTRAN function, 792
Index—26
<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>psatcu</td>
<td>FORTRAN</td>
<td>794</td>
</tr>
<tr>
<td>psatp</td>
<td>FORTRAN</td>
<td>796</td>
</tr>
<tr>
<td>psc</td>
<td>FORTRAN</td>
<td>361</td>
</tr>
<tr>
<td>psc3</td>
<td>FORTRAN</td>
<td>363</td>
</tr>
<tr>
<td>pscale</td>
<td>C</td>
<td>361</td>
</tr>
<tr>
<td>scale3</td>
<td>C</td>
<td>363</td>
</tr>
<tr>
<td>pschh</td>
<td>FORTRAN</td>
<td>800</td>
</tr>
<tr>
<td>pschm</td>
<td>FORTRAN</td>
<td>806</td>
</tr>
<tr>
<td>pschup</td>
<td>FORTRAN</td>
<td>804</td>
</tr>
<tr>
<td>pschxp</td>
<td>FORTRAN</td>
<td>798</td>
</tr>
<tr>
<td>pscmd</td>
<td>FORTRAN</td>
<td>809</td>
</tr>
<tr>
<td>pscnrs</td>
<td>FORTRAN</td>
<td>814</td>
</tr>
<tr>
<td>pscr</td>
<td>FORTRAN</td>
<td>811</td>
</tr>
<tr>
<td>pseco</td>
<td>FORTRAN</td>
<td>816</td>
</tr>
<tr>
<td>psedci</td>
<td>FORTRAN</td>
<td>821</td>
</tr>
<tr>
<td>psedfg</td>
<td>FORTRAN</td>
<td>823</td>
</tr>
<tr>
<td>psedi</td>
<td>FORTRAN</td>
<td>825</td>
</tr>
<tr>
<td>psedm</td>
<td>FORTRAN</td>
<td>835</td>
</tr>
<tr>
<td>psedr</td>
<td>FORTRAN</td>
<td>827</td>
</tr>
<tr>
<td>psedt</td>
<td>FORTRAN</td>
<td>830</td>
</tr>
<tr>
<td>psep</td>
<td>FORTRAN</td>
<td>837</td>
</tr>
<tr>
<td>pseplb</td>
<td>FORTRAN</td>
<td>839</td>
</tr>
<tr>
<td>pserhm</td>
<td>FORTRAN</td>
<td>841</td>
</tr>
<tr>
<td>pset_anno_align</td>
<td>C</td>
<td>789</td>
</tr>
<tr>
<td>pset_anno_char_ht</td>
<td>C</td>
<td>792</td>
</tr>
<tr>
<td>pset_anno_char_up_vec</td>
<td>C</td>
<td>794</td>
</tr>
<tr>
<td>pset_anno_path</td>
<td>C</td>
<td>796</td>
</tr>
<tr>
<td>pset_anno_style</td>
<td>C</td>
<td>787</td>
</tr>
<tr>
<td>pset_char_expan</td>
<td>C</td>
<td>798</td>
</tr>
<tr>
<td>pset_char_ht</td>
<td>C</td>
<td>800</td>
</tr>
<tr>
<td>pset_char_space</td>
<td>C</td>
<td>802</td>
</tr>
<tr>
<td>pset_char_up_vec</td>
<td>C</td>
<td>804</td>
</tr>
<tr>
<td>pset_choice_mode</td>
<td>C</td>
<td>806</td>
</tr>
<tr>
<td>pset_colr_model</td>
<td>C</td>
<td>809</td>
</tr>
<tr>
<td>pset_colr_rep</td>
<td>C</td>
<td>811</td>
</tr>
<tr>
<td>pset_conf_res</td>
<td>C</td>
<td>814</td>
</tr>
<tr>
<td>pset_disp_upd_st</td>
<td>C</td>
<td>816</td>
</tr>
<tr>
<td>pset_edge_colr_ind</td>
<td>C</td>
<td>821</td>
</tr>
<tr>
<td>pset_edge_flag</td>
<td>C</td>
<td>823</td>
</tr>
<tr>
<td>pset_edge_ind</td>
<td>C</td>
<td>825</td>
</tr>
<tr>
<td>pset_edge_rep</td>
<td>C</td>
<td>827</td>
</tr>
<tr>
<td>pset_edgetype</td>
<td>C</td>
<td>830</td>
</tr>
<tr>
<td>pset_edgewidth</td>
<td>C</td>
<td>833</td>
</tr>
<tr>
<td>pset_edit_mode</td>
<td>C</td>
<td>835</td>
</tr>
<tr>
<td>pset_elem_ptr</td>
<td>C</td>
<td>837</td>
</tr>
<tr>
<td>pset_elem_ptr_label</td>
<td>C</td>
<td>839</td>
</tr>
<tr>
<td>pset_err_hand</td>
<td>C</td>
<td>840</td>
</tr>
<tr>
<td>pset_err_hand_mode</td>
<td>C</td>
<td>841</td>
</tr>
<tr>
<td>pset_global_tran</td>
<td>C</td>
<td>843</td>
</tr>
<tr>
<td>pset_global_tran3</td>
<td>C</td>
<td>845</td>
</tr>
<tr>
<td>pset_highl_filter</td>
<td>C</td>
<td>847</td>
</tr>
<tr>
<td>pset_hihr Id</td>
<td>C</td>
<td>849</td>
</tr>
<tr>
<td>pset_hihr_mode</td>
<td>C</td>
<td>851</td>
</tr>
<tr>
<td>pset_indiv_asf</td>
<td>C</td>
<td>853</td>
</tr>
<tr>
<td>pset_int_colr_ind</td>
<td>C</td>
<td>857</td>
</tr>
<tr>
<td>pset_int_ind</td>
<td>C</td>
<td>859</td>
</tr>
<tr>
<td>pset_int_rep</td>
<td>C</td>
<td>861</td>
</tr>
<tr>
<td>pset_int_style</td>
<td>C</td>
<td>864</td>
</tr>
<tr>
<td>pset_int_style_ind</td>
<td>C</td>
<td>867</td>
</tr>
<tr>
<td>pset_invis_filter</td>
<td>C</td>
<td>869</td>
</tr>
<tr>
<td>pset_line_colr_ind</td>
<td>C</td>
<td>911</td>
</tr>
<tr>
<td>pset_line_ind</td>
<td>C</td>
<td>913</td>
</tr>
<tr>
<td>pset_line_rep</td>
<td>C</td>
<td>915</td>
</tr>
<tr>
<td>pset_linetype</td>
<td>C</td>
<td>871</td>
</tr>
<tr>
<td>pset_linewidth</td>
<td>C</td>
<td>873</td>
</tr>
<tr>
<td>pset_loc_mode</td>
<td>C</td>
<td>881</td>
</tr>
<tr>
<td>pset_local_tran</td>
<td>C</td>
<td>875</td>
</tr>
<tr>
<td>pset_local_tran3</td>
<td>C</td>
<td>878</td>
</tr>
<tr>
<td>pset_marker_colr_ind</td>
<td>C</td>
<td>918</td>
</tr>
<tr>
<td>pset_marker_ind</td>
<td>C</td>
<td>920</td>
</tr>
<tr>
<td>pset_marker_rep</td>
<td>C</td>
<td>922</td>
</tr>
<tr>
<td>pset_marker_size</td>
<td>C</td>
<td>884</td>
</tr>
<tr>
<td>pset_marker_type</td>
<td>C</td>
<td>886</td>
</tr>
<tr>
<td>pset_model_clip_ind</td>
<td>C</td>
<td>888</td>
</tr>
<tr>
<td>pset_model_clip_vol</td>
<td>C</td>
<td>890</td>
</tr>
<tr>
<td>pset_model_clip_vol3</td>
<td>C</td>
<td>893</td>
</tr>
<tr>
<td>pset_pat_ref_point</td>
<td>C</td>
<td>896, 898</td>
</tr>
<tr>
<td>pset_pat_rep</td>
<td>C</td>
<td>900</td>
</tr>
<tr>
<td>pset_pat_size</td>
<td>C</td>
<td>902</td>
</tr>
<tr>
<td>pset_pick_filter</td>
<td>C</td>
<td>904</td>
</tr>
<tr>
<td>pset_pick_id</td>
<td>C</td>
<td>906</td>
</tr>
<tr>
<td>pset_pick_mode</td>
<td>C</td>
<td>908</td>
</tr>
<tr>
<td>pset_rendering_colr_model</td>
<td>C</td>
<td>925</td>
</tr>
</tbody>
</table>
pset_string_mode—C function, 926
pset_stroke_mode—C function, 929
pset_text_align—C function, 932
pset_text_colr_ind—C function, 935
pset_text_font—C function, 937
pset_text_ind—C function, 939
pset_text_path—C function, 941
pset_text_prec—C function, 943
pset_text_rep—C function, 945
pset_view_ind—C function, 951
pset_view_rep—C function, 953
pset_view_rep3—C function, 957
pset_view_tran_in_pri—C function, 962
pset_ws_vp—C function, 964
pset_ws_vp3—C function, 967
pset_ws_win—C function, 970
pset_ws_win3—C function, 973
psewsc—FORTRAN function, 833
psgmt—FORTRAN function, 843
psgmt3—FORTRAN function, 845
pshlft—FORTRAN function, 847
pshrid—FORTRAN function, 849
pshrm—FORTRAN function, 851
psiasf—FORTRAN function, 853
psici—FORTRAN function, 857
psii—FORTRAN function, 859
psir—FORTRAN function, 861
psis—FORTRAN function, 864
psisi—FORTRAN function, 867
psivft—FORTRAN function, 869
pslcm—FORTRAN function, 881
pslmt—FORTRAN function, 875
pslmt3—FORTRAN function, 878
psln—FORTRAN function, 871
pslwsc—FORTRAN function, 873
psmch—FORTRAN function, 342
psmcli—FORTRAN function, 888
psmcv—FORTRAN function, 890
psmcv3—FORTRAN function, 893
psmk—FORTRAN function, 886
psmksc—FORTRAN function, 884
psmlc—FORTRAN function, 344
psmlc3—FORTRAN function, 346
psmpk—FORTRAN function, 348
psmsk—FORTRAN function, 353
psmsk3—FORTRAN function, 356
psmst—FORTRAN function, 351
psmv1—FORTRAN function, 359
pspa—FORTRAN function, 902
pspar—FORTRAN function, 900
psparf—FORTRAN function, 896
pspkft—FORTRAN function, 904
pspkid—FORTRAN function, 906
pspkm—FORTRAN function, 908
psplci—FORTRAN function, 911
pspli—FORTRAN function, 913
psplr—FORTRAN function, 915
pspmci—FORTRAN function, 918
pspmi—FORTRAN function, 920
pspmr—FORTRAN function, 922
psprpv—FORTRAN function, 898
psrcm—FORTRAN function, 925
psskm—FORTRAN function, 929
psstm—FORTRAN function, 926
pstxal—FORTRAN function, 932
pstxic—FORTRAN function, 935
pstxfn—FORTRAN function, 937
psti—FORTRAN function, 939
pstxp—FORTRAN function, 941
pstxpr—FORTRAN function, 943
pstxr—FORTRAN function, 945
psvlm—FORTRAN function, 948
psvtip—FORTRAN function, 962
psvwi—FORTRAN function, 951
psvwr—FORTRAN function, 953
psvwr3—FORTRAN function, 957
pswkv—FORTRAN function, 964
pswkv3—FORTRAN function, 967
pswk—FORTRAN function, 970
pswk3—FORTRAN function, 973
ptext—C function, 365
ptext3—C function, 368
ptp—FORTRAN function, 371
Q
Quick Update Methods
INQUIRE DEFAULT DISPLAY UPDATE STATE, 438
INQUIRE DISPLAY UPDATE STATE, 494
SET DISPLAY UPDATE STATE, 816

R
READ ITEM FROM METAFILE
prditm—FORTRAN function, 290
pread_item—C function, 290
REDRAW ALL STRUCTURES
predraw_all_structs—C function, 291
prst—FORTRAN function, 291
Regeneration
SET DISPLAY UPDATE STATE, 816
UPDATE WORKSTATION, 385
REMOVE NAMES FROM SET
premove_names_set—C function, 294
pres—FORTRAN function, 294
Replace Mode
SET EDIT MODE, 835
REQUEST CHOICE
REQUEST CHOICE, continued
preq_choice—C function, 296
prqch—FORTRAN function, 296
Request Input Mode
REQUEST CHOICE, 296
REQUEST LOCATOR 3, 301
REQUEST PICK, 304
REQUEST STRING, 307
REQUEST STROKE 3, 312
REQUEST VALUATOR, 315
REQUEST LOCATOR
preq_loc—C function, 298
prqlc—FORTRAN function, 298
REQUEST LOCATOR 3
preq_loc3—C function, 301
prqlc3—FORTRAN function, 301
REQUEST PICK
preq_pick—C function, 304
prqpk—FORTRAN function, 304
REQUEST STRING
preq_string—C function, 307
prqst—FORTRAN function, 307
REQUEST STROKE
preq_stroke—C function, 309
prqsk—FORTRAN function, 309
REQUEST STROKE 3
preq_stroke3—C function, 312
prqsk3—FORTRAN function, 312
REQUEST VALUATOR
preq_val—C function, 315
prqvl—FORTRAN function, 315
Reserved Names, description, 1
RESTORE MODELLING CLIPPING VOLUME
prestore_model_clip_vol—C function, 317
prmcv—FORTRAN function, 317
RETRIEVE ALL STRUCTURES
prast—FORTRAN function, 318
pret_all structs—C function, 318
RETRIEVE PATHS TO ANCESTORS
prepan—FORTRAN function, 320
pret_paths_ances—C function, 320
RETRIEVE PATHS TO DESCENDANTS
prepde—FORTRAN function, 324

Index–29
RETRIEVE PATHS TO DESCENDANTS, continued
pret_paths_desc—C function, 324
RETRIEVE STRUCTURE IDENTIFIERS
pret_struct_ids—C function, 328
prsid—FORTRAN function, 328
RETRIEVE STRUCTURE NETWORKS
presn—FORTRAN function, 330
pret_struct_nets—C function, 330
RETRIEVE STRUCTURES
prest—FORTRAN function, 332
pret_structs—C function, 332
ROTATE
pro—FORTRAN function, 334
protate—C function, 334
ROTATE X
protate_x—C function, 336
prox—FORTRAN function, 336
ROTATE Y
protate_y—C function, 338
proy—FORTRAN function, 338
ROTATE Z
protate_z—C function, 340
proz—FORTRAN function, 340
SAMPLE CHOICE
psample_choice—C function, 342
psmch—FORTRAN function, 342
Sample Input Mode
SAMPLE CHOICE, 342
SAMPLE LOCATOR 3, 346
SAMPLE PICK, 348
SAMPLE STRING, 351
SAMPLE STROKE 3, 356
SAMPLE VALUATOR, 359
SAMPLE LOCATOR
psample_loc—C function, 344
psmlc—FORTRAN function, 344
SAMPLE LOCATOR 3
psample_loc3—C function, 346
psmlc3—FORTRAN function, 346
SAMPLE PICK
psample_pick—C function, 348
psmpk—FORTRAN function, 348
SAMPLE STRING
psample_string—C function, 351
psmst—FORTRAN function, 351
SAMPLE STROKE
psample_stroke—C function, 353
psmsk—FORTRAN function, 353
SAMPLE STROKE 3
psample_stroke3—C function, 356
psmsk3—FORTRAN function, 356
SAMPLE VALUATOR
psample_val—C function, 359
psmvl—FORTRAN function, 359
SCALE
psc—FORTRAN function, 361
pscale—C function, 361
SCALE 3
psc3—FORTRAN function, 363
pscale3—C function, 363
Searching
ELEMENT SEARCH, 95
INCREMENTAL SPATIAL SEARCH, 154
INCREMENTAL SPATIAL SEARCH 3, 160
SET ANNOTATION STYLE
psans—FORTRAN function, 787
pset_anno_style—C function, 787
SET ANNOTATION TEXT ALIGNMENT
psatal—FORTRAN function, 789
pset_anno_align—C function, 789
SET ANNOTATION TEXT CHARACTER HEIGHT
psatch—FORTRAN function, 792
pset_anno_char_ht—C function, 792
SET ANNOTATION TEXT CHARACTER UP VECTOR
psatcu—FORTRAN function, 794
pset_anno_char_up_vec—C function, 794
SET ANNOTATION TEXT PATH
psatp—FORTRAN function, 796
pset_anno_path—C function, 796
SET CHARACTER EXPANSION FACTOR
pschxp—FORTRAN function, 798
pset_char_expan—C function, 798
SET CHARACTER HEIGHT
pschh—FORTRAN function, 800
pset_char_ht—C function, 800
Index—30
SET CHARACTER SPACING
  pset_char_space—C function, 802
  pschsp—FORTRAN function, 802
SET CHARACTER UP VECTOR
  pset_char_up_vec—C function, 804
  pschup—FORTRAN function, 804
SET CHOICE MODE
  pset_choice_mode—C function, 806
  pschm—FORTRAN function, 806
SET COLOUR MODEL
  pset_colr_model—C function, 809
  pscmd—FORTRAN function, 809
SET COLOUR REPRESENTATION
  pset_colr_rep—C function, 811
  pscr—FORTRAN function, 811
SET CONFLICT RESOLUTION
  pset_conf_res—C function, 814
  pschnrs—FORTRAN function, 814
SET DISPLAY UPDATE STATE
  pset_disp_upd_st—C function, 816
  pseco—FORTRAN function, 816
SET EDGE COLOUR INDEX
  pset_edge_colr_ind—C function, 821
  psedci—FORTRAN function, 821
SET EDGE FLAG
  pset_edge_flag—C function, 823
  psedfg—FORTRAN function, 823
SET EDGE INDEX
  pset_edge_ind—C function, 825
  psedi—FORTRAN function, 825
SET EDGE REPRESENTATION
  pset_edge_rep—C function, 827
  psedr—FORTRAN function, 827
SET EDGETYPE
  pset_edgetype—C function, 830
  psedt—FORTRAN function, 830
SET EDGEWIDTH SCALE FACTOR
  pset_edgewidth—C function, 833
  psewsc—FORTRAN function, 833
SET EDIT MODE
  pset_edit_mode—C function, 835
  psedm—FORTRAN function, 835
SET ELEMENT POINTER
  pset_elem_ptr—C function, 837
  psepb—FORTRAN function, 839
SET ELEMENT POINTER AT LABEL
  pset_elem_ptr_label—C function, 839
SET ERROR HANDLING
  pset_err_hand—C function, 840
SET ERROR HANDLING MODE
  pset_err_hand_mode—C function, 841
  pserhm—FORTRAN function, 841
SET GLOBAL TRANSFORMATION
  pset_global_tran—C function, 843
  psgmt—FORTRAN function, 843
SET GLOBAL TRANSFORMATION 3
  pset_global_tran3—C function, 845
  psgmt3—FORTRAN function, 845
SET HIGHLIGHTING FILTER
  pset_highl_filter—C function, 847
  pshlft—FORTRAN function, 847
SET HLHSR IDENTIFIER
  pset_hlhlsr_id—C function, 849
  pshrid—FORTRAN function, 849
SET HLHSR MODE
  pset_hlhlsr_mode—C function, 851
  pshrm—FORTRAN function, 851
SET INDIVIDUAL ASF
  pset_indiv_asf—C function, 853
  psiasf—FORTRAN function, 853
SET INTERIOR COLOUR INDEX
  pset_int_colr_ind—C function, 857
  psici—FORTRAN function, 857
SET INTERIOR INDEX
  pset_int_ind—C function, 859
  psii—FORTRAN function, 859
SET INTERIOR REPRESENTATION
  pset_int_rep—C function, 861
  psir—FORTRAN function, 861
SET INTERIOR STYLE
  pset_int_style—C function, 864
  psis—FORTRAN function, 864
SET INTERIOR STYLE INDEX
  pset_int_style_ind—C function, 867
  psisi—FORTRAN function, 867
SET INVISIBILITY FILTER

Index—31
SET INVISIBILITY FILTER, continued
pset_invis_filter—C function, 869
psivft—FORTRAN function, 869
SET LINETYPE
pset_linetype—C function, 871
psln—FORTRAN function, 871
SET LINEWIDTH SCALE FACTOR
pset_linewidth—C function, 873
pslwsc—FORTRAN function, 873
SET LOCAL TRANSFORMATION
pset_local_tran—C function, 875
pslmt—FORTRAN function, 875
SET LOCAL TRANSFORMATION 3
pset_local_tran3—C function, 878
pslmt3—FORTRAN function, 878
SET LOCATOR MODE
pset_loc_mode—C function, 881
pslcm—FORTRAN function, 881
SET MARKER SIZE SCALE FACTOR
pset_marker_size—C function, 884
psmksc—FORTRAN function, 884
SET MARKER TYPE
pset_marker_type—C function, 886
psmk—FORTRAN function, 886
SET MODELLING CLIPPING INDICATOR
pset_model_clip_ind—C function, 888
psmcli—FORTRAN function, 888
SET MODELLING CLIPPING VOLUME
pset_model_clip_vol—C function, 890
psmcv—FORTRAN function, 890
SET MODELLING CLIPPING VOLUME 3
pset_model_clip_vol3—C function, 893
psmcv3—FORTRAN function, 893
SET PATTERN REFERENCE POINT
pset_pat_ref_point—C function, 896
psparf—FORTRAN function, 896
SET PATTERN REFERENCE POINT AND VECTORS
pset_pat_ref_point—C function, 898
psprpv—FORTRAN function, 898
SET PATTERN REPRESENTATION
pset_pat_rep—C function, 900
pspar—FORTRAN function, 900
SET PATTERN SIZE
pset_pat_size—C function, 902
pspa—FORTRAN function, 902
SET PATTERN SIZE, continued
pset_pat_size—C function, 902
pspa—FORTRAN function, 902
SET PICK FILTER
pset_pick_filter—C function, 904
pspkft—FORTRAN function, 904
SET PICK IDENTIFIER, See Generalized Structure Elements
pset_pick_id—C function, 906
pspkid—FORTRAN function, 906
SET PICK MODE
pset_pick_mode—C function, 908
pspkm—FORTRAN function, 908
SET POLYLINE COLOUR INDEX
pset_line_colr_ind—C function, 911
psplci—FORTRAN function, 911
SET POLYLINE INDEX
pset_line_ind—C function, 913
pspli—FORTRAN function, 913
SET POLYLINE REPRESENTATION
pset_line_rep—C function, 915
psplr—FORTRAN function, 915
SET POLYMARKER COLOUR INDEX
pset_marker_colr_ind—C function, 918
pspmci—FORTRAN function, 918
SET POLYMARKER INDEX
pset_marker_ind—C function, 920
pspmi—FORTRAN function, 920
SET POLYMARKER REPRESENTATION
pset_marker_rep—C function, 922
pspmr—FORTRAN function, 922
SET RENDERING COLOUR MODEL
pset_rendering_colr_model—C function, 925
psrcm—FORTRAN function, 925
SET STRING MODE
pset_string_mode—C function, 926
psstm—FORTRAN function, 926
SET STROKE MODE
pset_stroke_mode—C function, 929
psskm—FORTRAN function, 929
SET TEXT ALIGNMENT
pset_text_align—C function, 932
psstxal—FORTRAN function, 932

Index—32
SET TEXT COLOUR INDEX
   pset_text_colr_ind—C function, 935
   pstxci—FORTRAN function, 935

SET TEXT FONT
   pset_text_font—C function, 937
   pstxfn—FORTRAN function, 937

SET TEXT INDEX
   pset_text_ind—C function, 939
   pstxi—FORTRAN function, 939

SET TEXT PATH
   pset_text_path—C function, 941
   pstxp—FORTRAN function, 941

SET TEXT PRECISION
   pset_textPrec—C function, 943
   pstxpr—FORTRAN function, 943

SET TEXT REPRESENTATION
   pset_text_rep—C function, 945
   pstxr—FORTRAN function, 945

SET VALUATOR MODE
   pset_val_mode—C function, 948
   psvlm—FORTRAN function, 948

SET VIEW INDEX
   pset_view_ind—C function, 951
   psvwi—FORTRAN function, 951

SET VIEW REPRESENTATION
   pset_view_rep—C function, 953
   psvwr—FORTRAN function, 953

SET VIEW REPRESENTATION 3
   pset_view_rep3—C function, 957
   psvwr3—FORTRAN function, 957

SET VIEW TRANSFORMATION INPUT PRIORITY
   pset_view_tran_in_pri—C function, 962
   psvtip—FORTRAN function, 962

SET WORKSTATION VIEWPORT
   pset_ws_vp—C function, 964
   pswkv—FORTRAN function, 964

SET WORKSTATION VIEWPORT 3
   pset_ws_vp3—C function, 967
   pswkv3—FORTRAN function, 967

SET WORKSTATION WINDOW
   pset_ws_win—C function, 970
   pswkw—FORTRAN function, 970

SET WORKSTATION WINDOW 3
   pset_ws_win3—C function, 973

SET WORKSTATION WINDOW 3, continued
   pswkw3—FORTRAN function, 973

Simulations
   INQUIRE DEFAULT DISPLAY UPDATE
     STATE, 438
   INQUIRE DISPLAY UPDATE STATE, 494
   SET DISPLAY UPDATE STATE, 816

Special Attributes
   ADD NAMES TO SET, 22
   REMOVE NAMES FROM SET, 294
   SET HLHSR IDENTIFIER, 849
   SET PICK IDENTIFIER, 906

Special Primitives
   CELL ARRAY, 49
   CELL ARRAY 3, 52

String Input Devices
   GET STRING, 147
   INITIALIZE STRING, 214
   INITIALIZE STRING 3, 219
   INQUIRE DEFAULT STRING DEVICE DATA, 460
   INQUIRE DEFAULT STRING DEVICE DATA
     3, 464
   INQUIRE STRING DEVICE STATE, 721
   INQUIRE STRING DEVICE STATE 3, 726
   REQUEST STRING, 307
   SAMPLE STRING, 351
   SET STRING MODE, 926

Stroke Input Devices
   GET STROKE, 149
   GET STROKE 3, 151
   INITIALIZE STROKE, 224
   INITIALIZE STROKE 3, 232
   INQUIRE DEFAULT STROKE DEVICE DATA, 468
   INQUIRE DEFAULT STROKE DEVICE DATA
     3, 473
   INQUIRE STROKE DEVICE STATE, 731
   INQUIRE STROKE DEVICE STATE 3, 736
   REQUEST STROKE, 309
   REQUEST STROKE 3, 312
   SAMPLE STROKE, 353
   SAMPLE STROKE 3, 356
   SET STROKE MODE, 929

Structure Networks

Index—33
Structure Networks, continued
  ARCHIVE STRUCTURE NETWORKS, 36
  CHANGE STRUCTURE IDENTIFIER AND REFERENCES, 57
  CHANGE STRUCTURE REFERENCES, 60
  DELETE STRUCTURE NETWORK, 89
  EXECUTE STRUCTURE, 122
  INQUIRE PATHS TO ANCESTORS, 650
  INQUIRE PATHS TO DESCENDANTS, 654
  POST STRUCTURE, 288
  RETRIEVE PATHS TO ANCESTORS, 320
  RETRIEVE PATHS TO DESCENDANTS, 324
  RETRIEVE STRUCTURE NETWORKS, 330

Structures
  CHANGE STRUCTURE IDENTIFIER, 55
  CLOSE STRUCTURE, 65
  DELETE ALL STRUCTURES, 80
  DELETE STRUCTURE, 87
  EMPTY STRUCTURE, 104
  INQUIRE STRUCTURE IDENTIFIERS, 742
  INQUIRE STRUCTURE STATE VALUE, 744
  INQUIRE STRUCTURE STATUS, 745
  OPEN STRUCTURE, 270
  RETRIEVE ALL STRUCTURES, 318
  RETRIEVE STRUCTURES, 332

Text
  ptext—C function, 365
  ptx—FORTRAN function, 365

Text 3
  ptext3—C function, 368
  ptx3—FORTRAN function, 368

Text Attributes
  SET ANNOTATION STYLE, 787
  SET ANNOTATION TEXT ALIGNMENT, 789
  SET ANNOTATION TEXT CHARACTER HEIGHT, 792
  SET ANNOTATION TEXT CHARACTER UP VECTOR, 794
  SET ANNOTATION TEXT PATH, 796
  SET CHARACTER EXPANSION FACTOR, 798
  SET CHARACTER HEIGHT, 800
  SET CHARACTER SPACING, 802
  SET CHARACTER UP VECTOR, 804

Transformation Matrix
  BUILD TRANSFORMATION MATRIX 3, 46
  COMPOSE MATRIX 3, 70
  COMPOSE TRANSFORMATION MATRIX 3, 75
  ROTATE, 334
  ROTATE X, 336
  ROTATE Y, 338
  ROTATE Z, 340
  SCALE 3, 363
  SET GLOBAL TRANSFORMATION 3, 845
  SET LOCAL TRANSFORMATION 3, 878
  SET VIEW REPRESENTATION 3, 957
  TRANSFORM POINT 3, 373
  TRANSLATE 3, 377

Translate
  ptp—FORTRAN function, 371
  ptran_point—C function, 371

TRANSFORM POINT 3
  ptp3—FORTRAN function, 373
  ptran_point3—C function, 373

Text Attributes, continued
  SET TEXT ALIGNMENT, 932
  SET TEXT COLOUR INDEX, 935
  SET TEXT FONT, 937
  SET TEXT INDEX, 939
  SET TEXT PATH, 941
  SET TEXT PRECISION, 943

Text Primitives
  ANNOTATION TEXT RELATIVE, 24
  ANNOTATION TEXT RELATIVE 3, 28
  TEXT, 365
  TEXT 3, 368

Text Attributes
  SET EDGETYPE, 830
  SET LINETYPE, 871

TRANSFORM POINT
  ptp—FORTRAN function, 371
  ptran_point—C function, 371

TRANSFORM POINT 3
  ptp3—FORTRAN function, 373
  ptran_point3—C function, 373

Transformation Matrix
  BUILD TRANSFORMATION MATRIX 3, 46
  COMPOSE MATRIX 3, 70
  COMPOSE TRANSFORMATION MATRIX 3, 75
  ROTATE, 334
  ROTATE X, 336
  ROTATE Y, 338
  ROTATE Z, 340
  SCALE 3, 363
  SET GLOBAL TRANSFORMATION 3, 845
  SET LOCAL TRANSFORMATION 3, 878
  SET VIEW REPRESENTATION 3, 957
  TRANSFORM POINT 3, 373
  TRANSLATE 3, 377

Translate
  ptp—FORTRAN function, 371
  ptran_point—C function, 371

TRANSFORM POINT 3
  ptp3—FORTRAN function, 373
  ptran_point3—C function, 373

Transformation Matrix
  BUILD TRANSFORMATION MATRIX 3, 46
  COMPOSE MATRIX 3, 70
  COMPOSE TRANSFORMATION MATRIX 3, 75
  ROTATE, 334
  ROTATE X, 336
  ROTATE Y, 338
  ROTATE Z, 340
  SCALE 3, 363
  SET GLOBAL TRANSFORMATION 3, 845
  SET LOCAL TRANSFORMATION 3, 878
  SET VIEW REPRESENTATION 3, 957
  TRANSFORM POINT 3, 373
  TRANSLATE 3, 377

Translate
  ptp—FORTRAN function, 371
  ptran_point—C function, 371

Transformation Matrix
  BUILD TRANSFORMATION MATRIX 3, 46
  COMPOSE MATRIX 3, 70
  COMPOSE TRANSFORMATION MATRIX 3, 75
  ROTATE, 334
  ROTATE X, 336
  ROTATE Y, 338
  ROTATE Z, 340
  SCALE 3, 363
  SET GLOBAL TRANSFORMATION 3, 845
  SET LOCAL TRANSFORMATION 3, 878
  SET VIEW REPRESENTATION 3, 957
  TRANSFORM POINT 3, 373
  TRANSLATE 3, 377

Translate
  ptp—FORTRAN function, 371
  ptran_point—C function, 371
UNPACK DATA RECORD
purec—FORTRAN function, 379
UNPOST ALL STRUCTURES
punpost_all_structs—C function, 382
punpost—all—FORTRAN function, 382
UNPOST STRUCTURE
punpost_struct—C function, 383
punpost—all—FORTRAN function, 383
UPDATE WORKSTATION
pupd_ws—C function, 385
puwk—FORTRAN function, 385

V
Valuator Input Devices
GET VALUATOR, 153
INITIALIZE VALUATOR, 241
INITIALIZE VALUATOR 3, 250
INQUIRE DEFAULT VALUATOR DEVICE DATA, 478
INQUIRE DEFAULT VALUATOR DEVICE DATA 3, 483
INQUIRE VALUATOR DEVICE STATE, 759
INQUIRE VALUATOR DEVICE STATE 3, 763
REQUEST VALUATOR, 315
SAMPLE VALUATOR, 359
SET VALUATOR MODE, 948
Viewing
EVALUATE VIEW MAPPING MATRIX, 109
EVALUATE VIEW ORIENTATION MATRIX, 117
INQUIRE PREDEFINED VIEW REPRESENTATION, 714
INQUIRE VIEW FACILITIES, 767
INQUIRE VIEW REPRESENTATION, 769
SET VIEW INDEX, 951
SET VIEW REPRESENTATION, 953
SET VIEW REPRESENTATION 3, 957
SET VIEW TRANSFORMATION INPUT PRIORITY, 962

W
Wide Lines
SET EDGEWIDTH SCALE FACTOR, 833
SET LINEWIDTH SCALE FACTOR, 873
Workstation State Table
INQUIRE WORKSTATION STATE TABLE LENGTHS, 778
Workstation Transformation
INQUIRE WORKSTATION TRANSFORMATION, 781
INQUIRE WORKSTATION TRANSFORMATION 3, 784
SET WORKSTATION VIEWPORT, 964
SET WORKSTATION VIEWPORT 3, 967
SET WORKSTATION WINDOW, 970
SET WORKSTATION WINDOW 3, 973
Workstation Types
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES, 612
OPEN WORKSTATION, 272
Workstations
CLOSE WORKSTATION, 66
INQUIRE SET OF OPEN WORKSTATIONS, 717
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED, 719
INQUIRE WORKSTATION CONNECTION AND TYPE, 776
INQUIRE WORKSTATION STATE VALUE, 780
OPEN WORKSTATION, 272
UPDATE WORKSTATION, 385
WRITE ITEM TO METAFILE
pwitm—FORTRAN function, 387
pwrite_item—C function, 387

Z
Z-buffering
INQUIRE HLHSR IDENTIFIER FACILITIES, 586
INQUIRE HLHSR MODE, 592
INQUIRE HLHSR MODE FACILITIES, 589
SET HLHSR IDENTIFIER, 849
SET HLHSR MODE, 851