

INTERNATIONAL STANDARD

ISO/IEC
9593-4

First edition
1991-12-15

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

Part 4: C

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

Partie 4: C



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

AVA **Documents**
Document Sharing Hub

Contents

1 Scope	1
2 Normative references	2
3 The C language binding of PHIGS	3
3.1 Conformance	3
3.2 Functions versus macros	3
3.3 Character strings	3
3.4 Function identifiers	3
3.5 Registration	3
3.6 Identifiers for graphics items	3
3.7 Return values	4
3.8 Header files	4
3.9 Memory management	5
3.9.1 Inquiry functions which return simple lists	5
3.9.2 Inquiry functions which return complex data structures	5
3.9.3 Meaning of the size of an element	7
3.10 Inquiries returning structure elements	7
3.11 Error handling	7
3.11.1 Application defined error handlers	7
3.11.2 Error codes	7
3.11.3 C specific PHIGS errors	8
3.12 Storage of two-dimensional data	8
3.12.1 Storage of matrices	8
3.12.2 Storage of colour arrays	9
4 Tables	10
4.1 Abbreviation policy for construction of identifiers	10
4.2 Table of abbreviations	10
4.3 Function names	14
4.3.1 List ordered alphabetically by bound name	14
4.3.2 List ordered alphabetically by PHIGS function name	20
5 Type definitions	27
5.1 Mapping of PHIGS data types	27
5.2 Environmental type definitions	27
5.3 Implementation dependent type definitions	28
5.4 Implementation independent type definitions	38
6 Macro definitions	73
6.1 Function identifiers	73
6.2 Error codes	76
6.3 Miscellaneous	83
6.3.1 Linetypes	83
6.3.2 Marker types	83
6.3.3 Annotation styles	83
6.3.4 Colour models	83
6.3.5 Prompt and Echo Types	83
6.3.6 Default parameters of OPEN PHIGS	84
6.3.7 Element enumeration	84

© ISO/IEC 1991

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Foreword

7 C PHIGS functions	85
7.1 Notational conventions	85
7.2 Control functions	85
7.3 Output primitive functions	87
7.4 Attribute specification functions	89
7.4.1 Bundled attribute selection	89
7.4.2 Individual attribute selection	90
7.4.3 Aspect source flag setting	95
7.4.4 Workstation attribute table definition	95
7.4.5 Workstation filter definition	96
7.4.6 Colour model control	97
7.4.7 HLHSR attributes	97
7.5 Transformation and clipping functions	97
7.5.1 Modelling transformations and clipping	97
7.5.2 View operation	99
7.5.3 Workstation transformation	99
7.5.4 Utility functions to support modelling	100
7.5.5 Utility functions to support viewing	104
7.6 Structure content functions	105
7.7 Structure manipulation functions	107
7.8 Structure display functions	108
7.9 Structure archiving functions	108
7.10 Input functions	111
7.10.1 Pick identifier and filter	111
7.10.2 Initialization of input devices	112
7.10.3 Setting the mode of input devices	115
7.10.4 Request input functions	116
7.10.5 Sample input functions	118
7.10.6 Event input functions	120
7.11 Metafile functions	122
7.12 Inquiry functions	123
7.12.1 Inquiry functions for operating state values	123
7.12.2 Inquiry functions for PHIGS description table	123
7.12.3 Inquiry functions for PHIGS state list	125
7.12.4 Inquiry functions for workstation state list	126
7.12.5 Inquiry functions for workstation description table	137
7.12.6 Inquiry function for structure state list	149
7.12.7 Inquiry functions for structure content	150
7.12.8 Inquiry functions for error state list	153
7.13 Error control	153
7.14 Special interfaces	154
7.15 Binding defined utility functions	154
Annexes	
A Data types in compilation order and external functions	155
A.1 Macro definitions	155
A.2 Types in compilation order	166
A.3 External functions	196
B Example Programs	248
B.1 star	248
B.2 iron	251
B.3 dyna_star	258
B.4 show_line	264
B.5 xform_pline	269
C Macros for short function identifiers	277

C.1 Short function identifiers	277
D Memory management	284
D.1 Introduction	284
D.2 Functions that return simple lists	284
D.2.1 Operation of ping_list_line_inds	285
D.3 Functions that return complex data structures	288
D.3.1 Operation of pcreate_store	289
D.3.2 Operation of ping_stroke_st and ping_pat_rep	291
D.3.3 Operation of pdel_store	295
E Function Lists	297
E.1 List of functions ordered alphabetically by function name	297
E.2 List of functions ordered alphabetically by bound name	303

Foreword

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

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

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

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

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

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

Introduction

The Programmer's Hierarchical Interactive Graphics System (PHIGS), the functional description of which is given in ISO/IEC 9592-1, is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this part of ISO/IEC 9593 is to define a standard binding for the C computer programming language.

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

Part 4:

C

1 Scope

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

2 Normative references

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

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

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

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

3 The C language binding of PHIGS

3.1 Conformance

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

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

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

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

3.2 Functions versus macros

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

3.3 Character strings

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

3.4 Function identifiers

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

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

3.5 Registration

ISO/IEC 9592-1 reserves certain value ranges for registration¹ as graphical items. The registered graphical items will be bound to the C programming language (and other programming languages). The binding of registered items will be consistent with the bindings presented in this part of ISO/IEC 9593.

3.6 Identifiers for graphics items

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

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

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

The format of the identifiers for graphical items is

"PFFF_Tn"

where:

FFF is the abbreviation of the PHIGS function name.

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

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

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

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

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

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

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

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

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

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

3.7 Return values

All PHIGS C functions return `void`.

3.8 Header files

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

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

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

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

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

The C language binding of PHIGS

3.9 Memory management

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

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

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

3.9.1 Inquiry functions which return simple lists

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

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

- a) `num_elem_appl_list`: An integer input parameter which is the size of the application's list. The value of `num_elem_appl_list` indicates the number of list elements which will fit into the application's list. A value of 0 is valid and allows the application to determine the size of the implementation's list (which is returned via `num_elem_impl_list`) without having the implementation return any of the elements of its list.
- b) `start_ind`: An integer input parameter which is the starting index into the implementation's list. (Index 0 is the first element of both the implementation's and application's list.) `start_ind` indicates the element in the implementation's list that is copied into index 0 of the application's list. Elements are copied sequentially from the implementation's list into the application's list until the application's list is full or there are no more elements in the implementation's list.
If `start_ind` is out of range, error number 2200 (PE_START_IND_INVALID) is returned as the value of the error indicator parameter.
- c) `num_elem_impl_list`: An output parameter which is a pointer to an integer. The implementation stores into this parameter the number of elements that are in the implementation's list.

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

3.9.2 Inquiry functions which return complex data structures

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

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

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

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

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

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

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

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

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

CREATE STORE (PHOP, *, *, *)

Parameters:

OUT	error indicator	I
OUT	store	STORE

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

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

2203 Ignoring function, error allocating *Store*.

Errors: none

DELETE STORE (PHOP, *, *, *)

Parameters:

OUT	error indicator	I
IN/OUT	store	STORE

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

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

Errors: none

The C language binding of PHIGS

3.9.3 Meaning of the size of an element

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

3.10 Inquiries returning structure elements

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

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

3.11 Error handling

3.11.1 Application defined error handlers

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

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

Parameters:

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

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

Errors: none

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

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

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

3.11.2 Error codes

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

3.11.3 C specific PHIGS errors

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

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

3.12 Storage of two-dimensional data

3.12.1 Storage of matrices

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

The elements of a PHIGS 3×3 matrix are:

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

and these elements are stored such that:

$$\begin{array}{lll} m[0][0] = a; & m[0][1] = b; & m[0][2] = c; \\ m[1][0] = d; & m[1][1] = e; & m[1][2] = f; \\ m[2][0] = g; & m[2][1] = h; & m[2][2] = i; \end{array}$$

where `m` is of type `Pmatrix`.

The elements of a PHIGS 4×4 matrix are:

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

and these elements are stored such that:

$$\begin{array}{llll} m[0][0] = a; & m[0][1] = b; & m[0][2] = c; & m[0][3] = d; \\ m[1][0] = e; & m[1][1] = f; & m[1][2] = g; & m[1][3] = h; \\ m[2][0] = i; & m[2][1] = j; & m[2][2] = k; & m[2][3] = l; \\ m[3][0] = m; & m[3][1] = n; & m[3][2] = o; & m[3][3] = p; \end{array}$$

The C language binding of PHIGS

where m is of type `Pmatrix3`.

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

$$\begin{aligned}x &\rightarrow x'/w' \\ y &\rightarrow y'/w' \\ z &\rightarrow z'/w'\end{aligned}$$

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

$$\begin{aligned}x' &= p[0][0]*x + p[0][1]*y + p[0][2]*z + p[0][3]; \\ y' &= p[1][0]*x + p[1][1]*y + p[1][2]*z + p[1][3]; \\ z' &= p[2][0]*x + p[2][1]*y + p[2][2]*z + p[2][3]; \\ w' &= p[3][0]*x + p[3][1]*y + p[3][2]*z + p[3][3];\end{aligned}$$

where p is of type `Pmatrix3`.

3.12.2 Storage of colour arrays

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

$$\text{colr_ind}_{(i,j)} = \text{colr_rect.colr_array}[i + DX*j];$$

where `colr_rect` is of type `Ppat_rep` and

$$\begin{aligned}\text{colr_rect.dims.size_x} &= DX; \\ \text{colr_rect.dims.size_y} &= DY; \\ i &= 0, \dots, DX-1; \\ j &= 0, \dots, DY-1;\end{aligned}$$

4 Tables

4.1 Abbreviation policy for construction of identifiers

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

- a) All identifiers in this part of ISO/IEC 9593 are abbreviated using the same abbreviations for every component and using underscores to denote blanks.
- b) The plural of an expression is constructed by adding an "s" after its abbreviation; so, for example, "vector" is abbreviated to "vec" and "vectors" is abbreviated to "vecs"; if an expression is mapped to NULL then its plural is also abbreviated to NULL.
- c) Digits are preceded by underscores, except when the 3D version of a function or type is denoted, then no underscore is used.
- d) The following are exceptions to the preceding rules:
 - 1) The phrases "polyline", "polymarker", "fill area", "fill area set", and "annotation text" are not abbreviated when they are used in the primitive functions POLYLINE(3), POLYMARKER(3), FILL AREA(3), FILL AREA SET(3), and ANNOTATION TEXT RELATIVE(3).
 - 2) The phrase "generalized drawing primitives" is abbreviated "gdp".
 - 3) The phrase "generalized structure elements" is abbreviated "gsc".
 - 4) The phrase "length" is not deleted when forming the macro identifiers for error numbers.
- e) Construction of identifiers:
 - 1) function names:
"p" (lower case) followed by abbreviated function name in lower case;
 - 2) data types:
"P" (upper case) followed by abbreviated data type name in lower case;
 - 3) fields of data types:
The name of a field comes from the description of the abstract parameter in PHIGS;
 - 4) Function macros:
"Pfn_" followed by abbreviation of function names in lower case;
 - 5) Error macros:
"PE_" followed by some abbreviated expression in upper case;
 - 6) Fields of enumeration types:
"P" (upper case) followed by a prefix followed by an abbreviation of the field names; this prefix is constant for each enumeration field; all the fields are in upper case.

4.2 Table of abbreviations

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

Table 1 - Abbreviations ordered alphabetically

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

Table 1 - Abbreviations ordered alphabetically

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

Table 1 - Abbreviations ordered alphabetically

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

Table 1 - Abbreviations ordered alphabetically

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

4.3 Function names

4.3.1 List ordered alphabetically by bound name

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

Table 2 - Function names ordered by bound name

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

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
peval_view_ori_matrix3	EVALUATE VIEW ORIENTATION MATRIX 3
pexec_struct	EXECUTE STRUCTURE
pfill_area	FILL AREA
pfill_area3	FILL AREA 3
pfill_area_set	FILL AREA SET
pfill_area_set3	FILL AREA SET 3
pflush_events	FLUSH DEVICE EVENTS
pgdp	GENERALIZED DRAWING PRIMITIVE
pgdp3	GENERALIZED DRAWING PRIMITIVE 3
pget_choice	GET CHOICE
pget_item_type	GET ITEM TYPE FROM METAFILE
pget_loc	GET LOCATOR
pget_loc3	GET LOCATOR 3
pget_pick	GET PICK
pget_string	GET STRING
pget_stroke	GET STROKE
pget_stroke3	GET STROKE 3
pget_val	GET VALUATOR
pgse	GENERALIZED STRUCTURE ELEMENT
pincr_spa_search	INCREMENTAL SPATIAL SEARCH
pincr_spa_search3	INCREMENTAL SPATIAL SEARCH 3
pinit_choice	INITIALIZE CHOICE
pinit_choice3	INITIALIZE CHOICE 3
pinit_loc	INITIALIZE LOCATOR
pinit_loc3	INITIALIZE LOCATOR 3
pinit_pick	INITIALIZE PICK
pinit_pick3	INITIALIZE PICK 3
pinit_string	INITIALIZE STRING
pinit_string3	INITIALIZE STRING 3
pinit_stroke	INITIALIZE STROKE
pinit_stroke3	INITIALIZE STROKE 3
pinit_val	INITIALIZE VALUATOR
pinit_val3	INITIALIZE VALUATOR 3
pinq_all_conf_structs	INQUIRE ALL CONFLICTING STRUCTURES
pinq_anno_fac	INQUIRE ANNOTATION FACILITIES
pinq_ar_files	INQUIRE ARCHIVE FILES
pinq_ar_st	INQUIRE ARCHIVE STATE VALUE
pinq_choice_st	INQUIRE CHOICE DEVICE STATE
pinq_choice_st3	INQUIRE CHOICE DEVICE STATE 3
pinq_colr_fac	INQUIRE COLOUR FACILITIES
pinq_colr_model	INQUIRE COLOUR MODEL
pinq_colr_model_fac	INQUIRE COLOUR MODEL FACILITIES
pinq_colr_rep	INQUIRE COLOUR REPRESENTATION
pinq_conf_res	INQUIRE CONFLICT RESOLUTION
pinq_conf_structs_net	INQUIRE CONFLICTING STRUCTURES IN NETWORK
pinq_cur_elem_content	INQUIRE CURRENT ELEMENT CONTENT
pinq_cur_elem_type_size	INQUIRE CURRENT ELEMENT TYPE AND SIZE
pinq_def_choice_data	INQUIRE DEFAULT CHOICE DEVICE DATA
pinq_dcf_choice_data3	INQUIRE DEFAULT CHOICE DEVICE DATA 3
pinq_def_disp_upd_st	INQUIRE DEFAULT DISPLAY UPDATE STATE
pinq_def_loc_data	INQUIRE DEFAULT LOCATOR DEVICE DATA

Table 2 - Function names ordered by bound name

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

Table 2 - Function names ordered by bound name

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

Table 2 - Function names ordered by bound name

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

Table 2 - Function names ordered by bound name

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

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
pset_text_font	SET TEXT FONT
pset_text_ind	SET TEXT INDEX
pset_text_path	SET TEXT PATH
pset_text_prec	SET TEXT PRECISION
pset_text_rep	SET TEXT REPRESENTATION
pset_val_mode	SET VALUATOR MODE
pset_view_ind	SET VIEW INDEX
pset_view_rep	SET VIEW REPRESENTATION
pset_view_rep3	SET VIEW REPRESENTATION 3
pset_view_tran_in_pri	SET VIEW TRANSFORMATION INPUT PRIORITY
pset_ws_vp	SET WORKSTATION VIEWPORT
pset_ws_vp3	SET WORKSTATION VIEWPORT 3
pset_ws_win	SET WORKSTATION WINDOW
pset_ws_win3	SET WORKSTATION WINDOW 3
ptext	TEXT
ptext3	TEXT 3
ptranlate	TRANSLATE
ptranlate3	TRANSLATE 3
ptran_point	TRANSFORM POINT
ptran_point3	TRANSFORM POINT 3
punpost_all_structs	UNPOST ALL STRUCTURES
punpost_struct	UNPOST STRUCTURE
pupd_ws	UPDATE WORKSTATION
pwrite_item	WRITE ITEM TO METAFILE

4.3.2 List ordered alphabetically by PHIGS function name

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

Table 3 - Function names ordered by PHIGS name

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

5 Type definitions

5.1 Mapping of PHIGS data types

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

Table 4 - Mapping of PHIGS data types to C

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

5.2 Environmental type definitions

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

Pfloat — floating point number

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

```
typedef float Pfloat;
```

Pint — integer

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

```
typedef int      Pint;
```

5.3 Implementation dependent type definitions

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

"xxx_tn"

where:

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

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

n is "n" is the item number; for unregistered items, the absolute value of the item number shall be used. The contents of the metafile item data record (Pitem_data) are not subject to registration and therefore do not follow this naming convention.

Pchoice_data CHOICE DATA RECORD

```

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

```

Pchoice_data3 CHOICE DATA RECORD 3

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

Pcolr_rep COLOUR REPRESENTATION

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

Pescape_in_data ESCAPE INPUT DATA RECORD

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

Pescape_out_data ESCAPE OUTPUT DATA RECORD

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

Pgdp_data GDP DATA RECORD

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

Pgdp_data3 GDP DATA RECORD 3

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

Pgse_data GSE DATA RECORD

```
typedef union {  
    struct Pgse_rl {  
        . . .      /* registration dependent      */  
    } gse_rl;  
    Pdata_unsupp;   /* unsupported GSE data record */  
    . . .          /* implementation defined      */  
} Pgse_data;
```

Pitem_data ITEM DATA RECORD

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

Ploc_data LOCATOR DATA RECORD

```

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

```

Ploc_data3 LOCATOR DATA RECORD 3

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

Ppick_data PICK DATA RECORD

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

Ppick_data3 PICK DATA RECORD 3

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

Pstring_data STRING DATA RECORD

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

Pstring_data3 STRING DATA RECORD 3

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

Pstroke_data STROKE DATA RECORD

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

Pstroke_data3 STROKE DATA RECORD 3

```

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

```

Pval_data VALUATOR DATA RECORD

```

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

```

Pval_data3 VALUATOR DATA RECORD 3

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

5.4 Implementation independent type definitions

Par_file ARCHIVE FILE

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

Par_file_list ARCHIVE FILE LIST

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

Par_st ARCHIVE STATE

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

Pasf ASPECT SOURCE FLAG

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

Paspect ASPECT

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

Pattr ATTRIBUTES USED

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

Pcieluv CIE L*U*V*

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

Pclip_ind CLIPPING INDICATOR

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

Pcolr_avail COLOUR AVAILABILITY

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

Pcolr_fac3 COLOUR FACILITIES

```
typedef struct {  
    Pint      num_colrs;        /* number of colours          */  
    Pcolr_avail colr_avail;     /* colour availability         */  
    Pint      num_pred_inds;    /* number of predefined colour indices */  
    Pcieluv   prim_colrs[3];    /* primary colours            */  
} Pcolr_fac3;
```

Pcompose_type COMPOSITION TYPE

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

Pconf_res CONFLICT RESOLUTION

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

Pctrl_flag CONTROL FLAG

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

Pdata DATA

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

Pdc_units DEVICE COORDINATE UNITS

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

Pdefer_mode DEFERRAL MODE

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

Pdisp_space_size DISPLAY SPACE SIZE

```
typedef struct {
    Pdc_units    dc_units;          /* device coordinate units */
    Pfloat_size  size_dc;           /* device size in coordinate units */
    Pint_size    size_raster;       /* device size in raster units */
} Pdisp_space_size;
```

Pdisp_space_size3 DISPLAY SPACE SIZE 3

```
typedef struct {
    Pdc_units    dc_units;          /* device coordinate units */
    Pfloat_size3 size_dc;           /* device volume in coordinate units */
    Pint_size3   size_raster;       /* device volume in raster units */
} Pdisp_space_size3;
```

Pdisp_surf_empty DISPLAY SURFACE EMPTY

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

Pdyns_structs DYNAMICS OF STRUCTURES

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

Pdyns_ws_attr DYNAMICS OF WORKSTATION ATTRIBUTES

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

Pdyn_mod DYNAMIC MODIFICATION

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

Pecho_switch ECHO SWITCH

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

Pedge_attrs EDGE ATTRIBUTES

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

Pedge_bundle EDGE BUNDLE

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

Pedge_facfs EDGE FACILITIES

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