INTERNATIONAL STANDARD

ISO/IEC 9593-1

First edition
1990-06-01

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

Part 1: FORTRAN

Systèmes de traitement de l'information — Infographie — Interfaces langage entre un programme d'application et son support graphique — Partie 1: FORTRAN

Reference number
ISO/IEC 9593-1:1990(E)

Contents

Page

Annexes

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 9593-1 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 processing systems — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings* :

— *Part 1: FORTRAN*

— *Part 2: Extended Pascal*

— *Part 3: ADA*

— *Part 4: C*

Annex B forms an integral part of this part of ISO/IEC 9593. Annex A is 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 FORTRAN computer programming language.

This page intentionally left blank

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

Part 1: FORTRAN

1 Scope

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 FORTRAN 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 listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 1539 : 1980, *Information processing systems - Programming Languages - FORTRAN.*

ISO/IEC 9592-1 : 1989, *Information processing systems - Computer graphics - Programmer's Hierarhical Interactive Graphics System (PHIGS) - Part 1 - functional description.*

ISO/IEC TR 9973 : 1988, *Information processing - Procedures for registration of graphical items.*

3 Principles

3.1 Specification

This part of ISO/IEC 9593 defines the PHIGS language binding interface for FORTRAN 77, as described in ISO 1539 : 1980. With some minor modifications, application programs can be transported between full FORTRAN 77 and FORTRAN 77 Subset PHIGS installations.

3.2 Mapping of PHIGS function names to FORTRAN subroutine names

The function names of PHIGS are all mapped to FORTRAN subroutine names that start with the letter 'P'. The mapping is generally done in a one-to-one correspondence to functions defined in ISO/IEC 9592-1. However, some functions are split into more than one subroutine in this binding, due to the number of parameters required. The remaining letters after the first one are obtained by deriving a unique acronym from the words of the function name; e.g., OPEN becomes OP, WORKSTATION becomes WK. Hence, the FORTRAN subroutine name of PHIGS function OPEN WORKSTATION is POPWK. For a list of all abbreviations, see clause 4. Names used internally that may be known outside PHIGS, e.g., during linking, start with some easily recognized and documented form such as 'PH' (subroutine, function, and common block names). Therefore, no external names starting with this construct should be chosen when using PHIGS, in order to avoid name conflicts.

3.3 Parameters

In general, the order of PHIGS function parameters is preserved. For some subroutines, however, there are additional parameters that have been inserted in the normal parameter sequence (e.g., array length for arrays).

Values of input parameters are unaltered by any PHIGS function as well as PACK DATA RECORD and UNPACK DATA RECORD.

In order that any element of a list (member of a set), such as the set of structure names, can be inquired, in this binding the inquiry functions return only a single element of a list (member of a set). In addition, the total number of elements of the list (members of the set) is always returned. The elements (members) are numbered starting from 1; each invocation of the inquiry function requires the desired element (member) number as an input parameter and returns the corresponding element (member). When the list (set) is empty, a zero is returned as the number of elements (members) and the parameter representing the single element (member) in the list is undefined.

3.4 The FORTRAN subset

The binding for FORTRAN 77 Subset is different from that for full FORTRAN 77 in order to accommodate the FORTRAN 77 Subset restrictions.

Those PHIGS subroutines in the full FORTRAN 77 binding that have arguments of type CHARACTER*(*) have alternative subroutine definitions that include fixed length character strings, CHARACTER*80, for the Subset.

In some cases an additional INTEGER parameter (the number of characters) appears in the parameter list and the Subset version is distinguished by the addition of a final 'S', so that they can coexist in the same implementation. In other cases the INTEGER is already present and the FORTRAN 77 Subset version has the same name as the full FORTRAN 77 version.

Principles **The FORTRAN subset**

A full FORTRAN 77 implementation shall include both subroutines when the names are distinct and only the full FORTRAN 77 version when the names are the same.

The enumeration values in this binding may be redefined by replacing the PARAMETER statements with corresponding DATA statements.

3.5 Error handling

There are two error routines in every PHIGS system, named PERLOG and PERHND. The user may replace the latter with his/her own subroutine using the same name, PERHND, and calling sequence. Furthermore, this user-defined error routine may call the system-defined error logging procedure PERLOG.

4 Generating FORTRAN subroutine names

For the binding of the PHIGS functions that inquire lists (sets), the word 'element' ('member') is added to the PHIGS name.

The derivation of the abbreviation for the subroutine names is performed in several steps. First, plurals are reduced to their singular form, and then compound terms are reduced to maintain uniqueness and appropriate name length. Finally, each remaining word is replaced by the null string or by an abbreviation.

Table 1 - Reduction of plurals to singulars

DEVICES	→	DEVICE
ELEMENTS	→	ELEMENT
EVENTS	→	EVENT
FILES	→	FILE
IDENTIFIERS	→	IDENTIFIER
INDICES	→	INDEX
LABELS	→	LABEL
LENGTHS	→	LENGTH
NETWORKS	→	NETWORK
PATHS	→	PATH
PRIMITIVES	→	PRIMITIVE
PRIORITIES	→	PRIORITY
REFERENCES	→	REFERENCE
TRANSFORMATIONS	→	TRANSFORMATION
STRUCTURES	→	STRUCTURE
TYPES	→	TYPE
WORKSTATIONS	→	WORKSTATION

Generating FORTRAN subroutine names

Table 2 - Reduce compound terms for uniqueness

ANNOTATION TEXT CHARACTER HEIGHT	→	ATCH
ANNOTATION TEXT CHARACTER UP	→	ATCU
INQUIRE ANNOTATION TEXT	→	QAT
SET ANNOTATION TEXT	→	SAT
ANNOTATION TEXT RELATIVE	→	ATR
ARCHIVE ALL	→	ARA
ARCHIVE STRUCTURE IDENTIFIER	→	ASID
IDENTIFIER AND REFERENCE	→	IR
CHANGE STRUCTURE IDENTIFIER	→	CSTID
DELETE ALL STRUCTURE	→	DAS
DYNAMICS OF STRUCTURE	→	DSTR
DYNAMICS OF WORKSTATION ATTRIBUTES	→	DSWA
EDGE FLAG	→	EDFG
ELEMENT POINTER	→	EP
ELEMENT CONTENT	→	ECO
ELEMENT TYPE AND SIZE	→	ETS
ERROR HANDLING MODE	→	ERHM
EVALUATE VIEW MAPPING MATRIX	→	EVMM
EVALUATE VIEW ORIENTATION MATRIX	→	EVOM
GENERALIZED STRUCTURE ELEMENT	→	GSE
INDIVIDUAL ASF	→	IASF
LIST OF	→	E
E AVAILABLE GENERALIZED DRAWING PRIMITIVE 3	→	EGD3
MAXIMUM LENGTH	→	L
MODELLING CLIPPING VOLUME	→	MCV
MODELLING CLIPPING	→	MCL
PATTERN REFERENCE POINT AND VECTORS	→	PRPV
PATTERN REFERENCE POINT	→	PARF
RETRIEVE ALL	→	RA
RETRIEVE STRUCTURE IDENTIFIER	→	RSID
SET OF	→	element
STRUCTURE IDENTIFIER	→	SID
STRUCTURE PATH	→	STPA
STRUCTURE NETWORK	→	SN
STRUCTURE STATE	→	STRS
TEXT FONT	→	TXFN
TRANSFORM POINT	→	TP
UNPOST ALL	→	UPA
VIEW TRANSFORMATION	→	VT
WORKSTATION STATE VALUE	→	WKST
(FORTRAN 77 subset)	→	S

Table 3 - Deletions

element	ALL	AND	AT	AVAILABLE
BETWEEN	DATA	DEVICE	EVENT	FACTOR
FROM	IN	LIST	METAFILE	MORE
NAMES	NUMBER	OF	SIZE	SUPPORTED
TABLE	TO	TYPE	VALUE	VECTOR
WHICH				

Table 4 - Abbreviations

3	→	3	EMPTY	→	EM
ADD	→	AD	ERROR	→	ER
ALIGNMENT	→	AL	ESCAPE	→	ESC
ANCESTORS	→	AN	EVALUATE	→	EV
ANNOTATION	→	AN	EXECUTE	→	EX
APPLICATION	→	AP	EXPANSION	→	XP
ARCHIVE	→	AR	EXTENT	→	X
AREA	→	A	FACILITIES	→	F
ARRAY	→	A	FILE	→	F
ASF	→	ASF	FILL	→	F
AWAIT	→	WAIT	FILTER	→	FT
BUILD	→	BL	FLAG	→	F
CATEGORY	→	CA	FLUSH	→	FLUSH
CELL	→	C	FONT	→	F
CHANGE	→	C	GENERALIZED	→	G
CHARACTER	→	CH	GET	→	GT
CHOICE	→	CH	GLOBAL	→	GM
CLASSIFICATION	→	CL	HANDLING	→	HND
CLOSE	→	CL	HEIGHT	→	H
COLOUR	→	C	HIGHLIGHTING	→	HL
COMPOSE	→	CO	HLHSR	→	HR
CONFLICT	→	CN	IDENTIFIER	→	ID
CONFLICTING	→	C	INCREMENTAL	→	I
CONNECTION	→	C	INDEX	→	I
CONTENT	→	CT	INDICATOR	→	I
COPY	→	C	INDIVIDUAL	→	I
CURRENT	→	C	INITIALIZE	→	IN
DEFAULT	→	D	INPUT	→	I
DESCENDANTS	→	DE	INQUIRE	→	Q
DELETE	→	D	INTERPRET	→	I
DISPLAY	→	D	INTERIOR	→	I
DRAWING	→	D	INVISIBILITY	→	IV
EDGE	→	ED	ITEM	→	ITM
EDGETYPE	→	EDT	LABEL	→	LB
EDGEWIDTH	→	EW	LENGTH	→	L
EDIT	→	ED	LINETYPE	→	LN
ELEMENT	→	EL	LINEWIDTH	→	LW
EMERGENCY	→	E	LOCAL	→	LM

Generating FORTRAN subroutine names

Table 4 (continued) - Abbreviations

LOCATOR	→	LC	RESOLUTION	→	RS
LOGGING	→	LOG	RESTORE	→	R
LOGICAL	→	L	RETRIEVE	→	RE
MAPPING	→	MP	ROTATE	→	RO
MARKER	→	MK	SAMPLE	→	SM
MATRIX	→	M	SCALE	→	SC
MESSAGE	→	MSG	SEARCH	→	S
MODE	→	M	SET	→	S
MODEL	→	MD	SIMULTANEOUS	→	SIM
NETWORK	→	N	SPACE	→	SP
OFFSET	→	OS	SPACING	→	SP
OPEN	→	OP	SPATIAL	→	S
ORIENTATION	→	OR	STATE	→	S
OVERFLOW	→	OV	STATUS	→	ST
PACK	→	P	STRING	→	ST
PATH	→	P	STROKE	→	SK
PATTERN	→	PA	STRUCTURE	→	ST
PHIGS	→	PH	STYLE	→	S
PICK	→	PK	SYSTEM	→	SY
POINTER	→	PT	TEXT	→	TX
POLYLINE	→	PL	TRANSFORM	→	T
POLYMARKER	→	PM	TRANSFORMATION	→	T
POST	→	PO	TRANSLATE	→	TR
POSTED	→	PO	UNPACK	→	U
PRECISION	→	PR	UNPOST	→	UPO
PREDEFINED	→	P	UP	→	UP
PRIMITIVE	→	P	UPDATE	→	U
PRIORITY	→	P	VALUATOR	→	VL
QUEUE	→	Q	VIEW	→	VW
RANGE	→	RA	VIEWPORT	→	V
READ	→	RD	VOLUME	→	VOL
RECORD	→	REC	WINDOW	→	W
REDRAW	→	R	WRITE	→	W
REFERENCE	→	RF	WORKSTATION	→	WK
REMOVE	→	RE	X	→	X
REPRESENTATION	→	R	Y	→	Y
REQUEST	→	RQ	Z	→	Z

5 Data types

In ISO/IEC 9592-1 parameters of several types are used. The following shows the correspondence between the types used in ISO/IEC 9592-1 and their realization in a FORTRAN implementation.

PHIGS Data Type	FORTRAN Data Types
I integer	INTEGER
A(I) array of integers	This is described more at the end of this clause, where the representations of CELL ARRAY and PATTERN are described.
R real	REAL
const × *simple_type* where *simple_type* is realized as I or R	(vector of values, for example 2×R) For input argument where const ≥6 or for output argument where const ≥4, then array of constant length is used, otherwise use separate parameters.
const 1 × const 2 × R (matrix of values, for example 2×3×R)	REAL array (const 1, const 2) For example, in order to store the projection transformation defined by:

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

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

$$x' = p[1,1]*x + p[2,1]*y + p[3,1]*z + p[4,1]$$
$$y' = p[1,2]*x + p[2,2]*y + p[3,2]*z + p[4,2]$$
$$z' = p[1,3]*x + p[2,3]*y + p[3,3]*z + p[4,3]$$
$$w' = p[1,4]*x + p[2,4]*y + p[3,4]*z + p[4,4]$$

S string

1) In a full FORTRAN 77 subroutine:
   a) INTEGER containing the number of characters returned (for output string argument only).
   b) CHARACTER*(*) containing the string. In addition, if a character string that is an input parameter may reasonably contain no characters, then an INTEGER (≥0) is used to give the number of characters to be passed to the subroutine.

2) In a FORTRAN 77 Subset subroutine:
   a) INTEGER containing the number of characters passed to the subroutine (for input string only, i.e. only one INTEGER needed for output).
   b) INTEGER containing the number of characters returned (for output string argument only). If the value is < 0 or > 80, error 2004 is

Data types

generated.
c) CHARACTER*80 containing the string.

P2 point — REAL, REAL containing the X- and Y-values

const × P2 (only occurs in non-inquiry functions)
Separate REAL parameters, with the X- and Y- coordinates of one point being followed by the X- and Y- coordinates of the next.

P3 point — REAL, REAL, REAL containing the X-, Y-, and Z-values

const × P3 — If const ≥2 REAL arrays xa(const), ya(const), za(const) are used, otherwise separate REAL arrays are used, with the X-, Y-, and Z-coordinates of one point being followed by the X-,Y-, and Z-coordinates of the next.

L(L(P2/3)) list of point lists (for fill area sets)
The following description applies to both 2D and 3D point lists except that the PZA array is not present for 2D point lists. The arguments that specify the list of point lists are as follows:

```
INTEGER NPL
INTEGER IXA(NPL)
REAL PXA(*), PYA(*), PZA(*)
```

where

NPL is the number of point lists
IXA is an array of end indices for the point lists
PXA, PYA, PZA are the coordinate arrays

The range of indices in PXA, PYA, and PZA of each point list is as follows:

1 to IXA(1) is the first point list
IXA(i-1)+1 to IXA(i) is the ith point list, for i=2, to NPL when NPL ≥2

Thus, for example:

a) 1 is the start index for the 1st point list,
b) IXA(i-1)+1 is the start index for the ith point list for all i=2 to NPL when NPL≥2, and
c) IXA(i) is the end index for all point lists i=1 to NPL.

In the actual arguments specifying the list of point lists (supplied by the application program), the following conditions shall hold true; otherwise error 2005 is generated, with the allowable exception:

d) NPL≥1,
e) PXA, PYA and PZA are dimensioned by at least IXA(NPL) (however it is allowable for the implementation not to generate error 2005 in this case),
f) IXA(1)≥3 (the first point list is at least 3 points),
g) IXA(i+1)-IXA(i)≥3 for i=1 to NPL-1, when NPL≥2 (the ith point list is at least 3 points).

V2 2D vector — REAL, REAL containing the X- and Y-values specifying an offset from some reference point in the coordinate system of the reference point.