INTERNATIONAL STANDARD

ISO
8651-1

First edition
1988-04-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ

Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings —

Part 1 : FORTRAN

Systèmes de traitement de l'information — Infographie — Système graphique de base (GKS) — Interface langage —

Partie 1 : FORTRAN

Reference number
ISO 8651-1 : 1988 (E)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 8651-1 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

Contents

Annexes

This page intentionally left blank

Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings —

Part 1 : FORTRAN

0 Introduction

The Graphical Kernel System (GKS), the functional description of which is given in ISO 7942, 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 8651 is to define a standard binding for the FORTRAN computer programming language.

1 Scope and field of application

ISO 7942 (GKS) specifies a language independent nucleus of a graphics system. For integration into a programming language, GKS is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO 8651 specifies such a language dependent layer for the FORTRAN language.

2 References

ISO 7942, *Information Processing - Computer graphics - Graphical Kernel System (GKS) functional description.*

ISO 1539, *Programming Languages - FORTRAN.*

3 The FORTRAN language binding of GKS

3.1 Specification

The GKS language binding interface for ISO FORTRAN 77 (ISO 1539) shall be described as in clauses 3, 4, 5, 6, 7, 8, and 9.

3.2 Mapping of GKS function names to FORTRAN subroutine names

The function names of GKS are all mapped to FORTRAN subroutine names which start with the letter G. The mapping is generally done in a one-to-one correspondence to ISO 7942. However, some inquiry 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., ACTIVATE becomes AC, WORKSTATION becomes WK. Hence, the FORTRAN subroutine name of GKS function ACTIVATE WORKSTATION is GACWK. For a list of all abbreviations, see clause 4. Names used internally which may be known outside GKS, e.g., during linking, start with some easily recognized and documented form such as GK (subroutine, function, and common block names). Therefore, no external names starting with this construct should be chosen when using GKS, in order to avoid name conflicts. Globally used GKS names may be renamed if necessary.

3.3 Parameters

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

Values of input parameters are unaltered by any GKS function, by PACK DATA RECORD, or by UNPACK DATA RECORD.

In order that the application program may inquire any element of a list (member of a set), such as the set of segment names, 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 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 GKS 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 the two versions can coexist in the same implementation. In other cases the INTEGER parameter is already present and the FORTRAN 77 Subset version has the same name as the full FORTRAN 77 version.

A full FORTRAN 77 implementation shall include both subroutines in the case 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 for the Subset by replacing the PARAMETER statements with corresponding DATA statements.

3.5 Error handling

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

4 Generating FORTRAN subroutine names

For the binding of the GKS functions which inquire lists (sets), the word element (member) is added to the GKS function name before the subroutine name is generated from the resulting terms.

The derivation of the abbreviation for the subroutine names is performed in several steps. First, plurals are reduced to their singular form, and grammatical derivations are unified. Next, some compound terms are reduced. Finally, each remaining word is replaced by the null string or by an abbreviation.

Plurals

ATTRIBUTES	→	ATTRIBUTE	NUMBERS	→	NUMBER
DEVICES	→	DEVICE	PRIMITIVES	→	PRIMITIVE
EVENTS	→	EVENT	PRIORITIES	→	PRIORITY
FACILITIES	→	FACILITY	SEGMENTS	→	SEGMENT
FLAGS	→	FLAG	TYPES	→	TYPE
INDICES	→	INDEX	VALUES	→	VALUE
NAMES	→	NAME	WORKSTATIONS	→	WORKSTATION

Keeping Uniqueness

ACTIVE	→	ACTIVATE
DRAWING	→	DRAW
IDENTIFIER	→	IDENTIFICATION
SPACING	→	SPACE

Reduce Compound Terms:

STATE TABLES	→	TABLES
TRANSFORMATION NUMBER	→	TRANSFORMATION N
SET member	→	member
CURRENT NORMALISATION	→	CN
MAXIMUM LENGTH	→	LENGTH

Deletions

ALL	FACTOR	LIST	OF	TABLES
AND	FROM	member	ON	TO
AVAILABLE	GKSM	MODIFICATION	POINT	TYPE
CURRENT	IN	MORE	SIZE	VALUE
DATA	INDICATOR	NAME	STATES	VECTOR
DEVICE	LENGTH	NUMBER	SUPPORTED	WITH
EVENT				

Abbreviations

ACCUMULATE	→	AC
ACTIVATE	→	AC
ALIGNMENT	→	AL
AREA	→	A
ARRAY	→	A
ASPECT	→	A
ASSOCIATE	→	A
ASSOCIATED	→	AS
ATTRIBUTE	→	A
AWAIT	→	WAIT
BASE	→	B
CATEGORY	→	CA
CELL	→	C
CHARACTER	→	CH
CHOICE	→	CH
CLASSIFICATION	→	CL
CLEAR	→	CLR
CLIPPING	→	CLIP
CLOSE	→	CL
COLOUR	→	C
CONNECTION	→	C
COPY	→	C
CREATE	→	CR
DEACTIVATE	→	DA
DEFAULT	→	D
DEFERRAL	→	D
DELETE	→	D
DETECTABILITY	→	DTEC
DIMENSIONS	→	D
DISPLAY	→	D
DRAW	→	D
DYNAMIC	→	D
element	→	E
EMERGENCY	→	E
ERROR	→	ER
ESCAPE	→	ESC
EVALUATE	→	EV
EXPANSION	→	XP
EXTENT	→	X
FACILITY	→	F
FILL	→	F
FLAG	→	F
FONT	→	F
GENERALISED	→	G
GET	→	GT
GKS	→	KS
HANDLING	→	HND
HEIGHT	→	H
HIGHLIGHTING	→	HLIT
IDENTIFICATION	→	ID
INDEX	→	I
INITIALISE	→	IN
INPUT	→	I
INQUIRE	→	Q
INSERT	→	IN
INTERIOR	→	I
INTERPRET	→	I
ITEM	→	ITM
LINE	→	LN
LINETYPE	→	LN
LINEWIDTH	→	LW
LOCATOR	→	LC
LOGGING	→	LOG
LOGICAL	→	L
MARKER	→	MK
MATRIX	→	M
MAXIMUM	→	M
MESSAGE	→	MSG
MODE	→	M
NORMALIZATION	→	N
OPEN	→	OP
OPERATING	→	OP
OVERFLOW	→	OV
PACK	→	P
PATH	→	P
PATTERN	→	PA
PICK	→	PK
PIXEL	→	PX
POLYLINE	→	PL
POLYMARKER	→	PM
PRECISION	→	P
PREDEFINED	→	P
PRIMITIVE	→	P
PRIORITY	→	P
QUEUE	→	Q
READ	→	RD
RECORD	→	REC
REDRAW	→	R
REFERENCE	→	RF
RENAME	→	REN
REPRESENTATION	→	R
REQUEST	→	RQ
SAMPLE	→	SM
SCALE	→	SC
SEGMENT	→	SG
SELECT	→	SEL
SET	→	S
SIMULTANEOUS	→	SIM
SOURCE	→	S
SPACE	→	SP
STATE	→	S
STRING	→	ST
STROKE	→	SK
STYLE	→	S
SURFACE	→	S
TEXT	→	TX
TRANSFORMATION	→	T
UNPACK	→	U
UPDATE	→	U
USE	→	US
VALUATOR	→	VL
VIEWPORT	→	VP
VISIBILITY	→	VIS
WIDTH	→	W
WINDOW	→	WN
WORKSTATION	→	WK
WRITE	→	W

5 Data types

In ISO 7942, parameters of several types are used. The following shows the correspondence between the types used in ISO 7942 and their realisation in a FORTRAN implementation.

GKS Data Type	FORTRAN Data Types
I integer	INTEGER
R real	REAL

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 which 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).
   c) CHARACTER*80 containing the string.

P point REAL, REAL containing the X- and Y-values

N name INTEGER

1) Workstation Identifier, Segment Name, Pick Identifier: An implementation may restrict the range but must at least provide all non-negative integers which are available at that implementation.

   NOTE - the default value for pick identifier is zero.

2) Workstation Type, Connection Identifier, Error File: The set of valid values is implementation dependent. The Connection Identifier and Error File may be logical unit numbers.
3) GDP Identifier, Escape Identifier: The set of legal values is described in ISO 7942.
4) Identification of GKS procedure: The range is shown under 'Enumeration Types'.

E enumeration INTEGER

NOTE - All values are mapped to the range zero to N-1, where N is the number of enumeration alternatives. Except for null values, the order of the enumeration alternatives is the same as in ISO 7942: null values always appear in the first position. If the integer value given by the application program is not in the range 0 to N-1, there is a language binding error condition (error 2000).

const x simple_type where simple_type is I or R (vector of values, for example 2xR)

1) In non-inquiry functions, separate simple_type parameters are used.

   NOTE - in GKS, const ≤4

2) In inquiry functions, if const ≤3, separate simple_type parameters are used; if const ≥4, a simple_type array of dimension const is used.

const x P (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.

const x E (only occurrence in GKS is const = 13)

An array of INTEGER elements of dimension const is used, each element being an enumeration alternative.

const 1 x const 2 x R (matrix of values, for example 2x3xR)

REAL array (const 1, const 2)

list of n values of one simple_type (for example nxI)

1) For input parameter:

   a) INTEGER (input parameter) containing length n of the list (unless the length is already present as a separate GKS parameter, in which case it is not duplicated)

   b) array of dimension n, whose elements are of the appropriate simple_type.

   When the length could legally be zero within GKS, the binding indicates the array dimension by *. The implementation checks that the given length is ≥0.

2) For output parameter in non-inquiry functions:

   a) INTEGER (input parameter) containing the dimension of the array

   b) INTEGER (output parameter) containing the number of elements of the array actually used.

   c) an array whose elements are of the appropriate simple_type. The input dimension being too small is a language binding error condition (error 2001).

   In both cases (input or output), where the simple_type is a point, there is a REAL array for the X-coordinates and another for the Y-coordinates.

3) For inquiry functions, a single call returns a single element of the list. For a complete list of length n,

   a) INTEGER (input parameter) containing the sequence number of required list element (in the range 0...n).

   b) INTEGER (output parameter) containing the number of items in the list n.

   c) a parameter of the appropriate simple_type containing the requested element.

   If the sequence number given is 0, the requested element returned is undefined, but an error is not indicated thereby; the number of items in the list n is returned. If the sequence number given is <0 or >n, then error 2002 is indicated, the number of items in the list is returned, but the requested element is undefined; the exception to this is when the list size is 0, and in that case an error is not indicated thereby.

4) A complete inquired list is returned from a single call when the maximum size of the list is a small constant m:

   a) INTEGER (output parameter) containing the number of elements of the array actually used.

b) an array of dimension m, whose elements are of the appropriate simple_type.

list of n values of a compound type (for example, nx4xR)
: This only occurs in an inquiry function. A single call returns a single element of the list exactly as for the list of values of one simple_type, except that here the requested element is several FORTRAN parameters.

array of integers (for example, nxnxI)
: This is described more fully below, where the representations of CELL ARRAY, PIXEL ARRAY and PATTERN ARRAY are described.

an ordered pair of different types (for example I;E)
: The different types are represented in turn in the FORTRAN parameter list.

DATA RECORD
: Represented as a set of scalar values and an array of type CHARACTER*80 containing the data. In addition, an INTEGER input parameter is used to dimension the array. Where the data record is an output parameter, an additional argument 'number of array elements of data record occupied' is needed. There are no scalar values except where the data record contains values which are compulsory in GKS.

: Although data can be read from and written into the data record with the FORTRAN READ and WRITE statements, special utility functions are defined to pack INTEGER, REAL, and CHARACTER data into the data record and to unpack the data record to the individual data items (GPREC, GUREC). The content of the packed data records is implementation dependent, but GPREC must perform the inverse function to GUREC and vice versa.

The representation of CELL ARRAY, PIXEL ARRAY, and PATTERN allows the user of the routines requiring a cell array parameter to pass any portion of the array as an argument. Two examples should make this clear.

The user can pass an entire two-dimensional array. In this case the number of columns of the cell array is the same as the first dimension of the FORTRAN array:

```
INTEGER DIMX, DIMY, CELLS (DIMX,DIMY)
CALL GCA (X1, Y1, X2, Y2, DIMX, DIMY, 1, 1, DIMX, DIMY, CELLS)
```

(1,1)	(2,1)	(3,1)	. . .	(DIMX,1)
(1,2)	(2,2)	(3,2)	. . .	(DIMX,2)
:	:	:		:
(1,DIMY)	(2,DIMY)	(3,DIMY)	. . .	(DIMX,DIMY)

To use an arbitrary portion of an array the user passes the upper left corner of the portion as the starting address and the dimensions of the entire array for the proper treatment of addresses. The area inside the small box is the cell array being passed:

```
INTEGER STARTX, STARTY, DX, DY, DIMX, DIMY, CELLS (DIMX,DIMY)
DATA STARTX/3/, STARTY/6/, DX/2/, DY/3/
CALL GCA (X1,Y1,X2,Y2,DIMX,DIMY,STARTX,STARTY,DX,DY,CELLS)
```

(1,1)	(2,1)	(3,1)	(4,1)	. . .	(DIMX,1)
(1,2)	(2,2)	(3,2)	(4,2)	. . .	(DIMX,2)
:	:	:	:		:
:	:				:
(1,6)	(2,6)	(3,6)	(4,6)	. . .	(DIMX,6)
(1,7)	(2,7)	(3,7)	(4,7)	. . .	(DIMX,7)
(1,8)	(2,8)	(3,8)	(4,8)	. . .	(DIMX,8)
:	:				:
:	:	:	:		:
(1,DIMY)	(2,DIMY)	(3,DIMY)	(4,DIMY)	. . .	(DIMX,DIMY)