INTERNATIONAL STANDARD

ISO/IEC 9593-3

First edition
1990-04-15

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

Part 3 : Ada

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

Partie 3 : Ada

Reference number
ISO/IEC 9593-3 : 1990 (E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) together form a system for worldwide standardization as a whole. 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 approval before their acceptance as International Standards. They are approved in accordance with procedures requiring at least 75 % approval by the national bodies voting.

International Standard ISO/IEC 9593-3 was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*.

Introduction

ISO/IEC 9592 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 of PHIGS to the Ada computer programming language.

This page intentionally left blank

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

Part 3 : Ada

1 Scope

ISO/IEC 9592 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 Ada computer programming 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 8652 : 1987, *Programming languages - Ada (Endorsement of ANSI Standard 1815A-1983).*

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

3 Principles

The PHIGS Binding to Ada is intended to be implementation independent except as it can be adapted by the PHIGS_CONFIGURATION package. The PHIGS Binding to Ada makes no assumptions concerning the availability of implementation dependent facilities defined by the Ada language. It does, however, limit the use of multi-tasking as described in 3.2.5. The Ada compiler shall be able to support the number of declarations contained in this PHIGS Binding to Ada.

This binding does not make any assumptions regarding the machine representation of the predefined Ada numeric types.

This binding assumes that the application programmer will supply an error file name, archive file names, and connection identifiers that are in an acceptable format for the Ada implementation.

This binding makes no assumptions regarding the format of a string specifying an error file name, archive file names, or connection identifiers for devices or metafiles.

3.1 Conformance

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

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

- The semantics of an implementation shall be those stated in ISO/IEC 9592 as modified or extended for Ada as stated in this part of ISO/IEC 9593.
- The package corresponding to PHIGS shall be an available Ada library unit, with all names as specified by this part of ISO/IEC 9593.

3.2 Implications of the Language

3.2.1 Functional Mapping

The functions of PHIGS are all mapped to Ada procedures. The mapping utilizes a one-to-one correspondence between the PHIGS functions and Ada procedures except for the PHIGS function INQUIRE TEXT EXTENT. This function is mapped to two overloaded Ada functions each named INQ_TEXT_EXTENT one which

supports modelling text and one which supports annotation text. Certain functions required by the binding but not defined by PHIGS are mapped to Ada functions and procedures.

3.2.2 Implementation and Host Dependencies

There are a number of implementation and host dependencies associated with the Ada compiler and runtime system used. These will affect the portability of application programs and their use of PHIGS. The application programmer should follow accepted practices for ensuring portability of Ada programs to avoid introducing problems when rehosting the application on another system. Implementation dependencies include runtime storage management and processor management.

3.2.3 Error Handling

The inquiry functions utilize error indicator parameters for the error returns, and do not raise Ada exceptions. The application program shall ensure that these error indicators are checked before attempting to access other parameters, since Ada implementations are not required to raise an exception if an undefined value is accessed.

The error handling requirements of PHIGS can be summarized as follows:

1. By default, a procedure named ERROR_HANDLING will be provided that simply reports the error by calling ERROR_LOGGING. This is called from the PHIGS function that detects the error.

2. The ERROR_HANDLING procedure may be replaced by one defined by the user.

The procedure ERROR_HANDLING is defined as a library subprogram:


```
with PHIGS_TYPES;
use PHIGS_TYPES;
procedure ERROR_HANDLING
  (ERROR_INDICATOR : in ERROR_NUMBER;
   PHIGS_FUNCTION  : in STRING;
   ERROR_FILE      : in FILE_ID := DEFAULT_ERROR_FILE);

-- The procedure ERROR_HANDLING is defined as a library subprogram,
-- and is not declared within package PHIGS.
```

This binding defines two different bodies for this subprogram; each shall be supplied by the implementation. The default body is the one required by PHIGS semantics. It simply calls ERROR_LOGGING and returns. The second body calls ERROR_LOGGING and then raises the exception PHIGS_ERROR. The PHIGS function shall be written so as not to handle PHIGS_ERROR (this is a requirement of the implementation). Thus, by Ada rules, the exception will be propagated back to the application program that called the PHIGS function in which the error was detected.

The means by which the user replaces the default body by either the exception-raising version or another one of his or her choosing is dependent upon the Ada library manager. Some implementations support multiple versions of a body with a single specification or otherwise allow hierarchical libraries with the sharing of common units. In other implementations, it may be necessary to duplicate the PHIGS library for each version of ERROR_HANDLING.

3.2.4 Data mapping

The simple and compound data types of PHIGS are bound to a variety of Ada scalar and compound types. Constraints on permitted values are reflected where possible in the type definitions. The general correspondence between the PHIGS data types and Ada binding data types is summarized below:

- PHIGS integer types (I) are mapped to Ada integer types.
- PHIGS real types (R) are mapped to Ada floating-point types.
- PHIGS string types (S) are mapped to the predefined Ada type STRING, or to a type providing for variable length strings.
- PHIGS point types (P2, P3) are mapped to Ada record types.
- PHIGS vector types (V2, V3) are mapped to Ada record types.
- PHIGS enumeration types (E) are mapped to Ada enumeration types.
- PHIGS name types (NM) are mapped to an Ada integer type. The PHIGS name set composite type SET(NM) is mapped to an Ada private type. A set of subprograms for operating on objects of this private type is explicitly defined by this binding.
- PHIGS filter types (FR) are mapped to an Ada record type.
- PHIGS pick path item type (PP) is mapped to an Ada record type. PHIGS pick paths are mapped to an Ada array type.
- PHIGS element reference type (ER) is mapped to an Ada record type.
- PHIGS half-space types (HS2, HS3) are mapped to Ada record types.
- PHIGS font/precision pair type (FP) is mapped to an Ada record type.
- PHIGS structure element type (SE) is mapped to an Ada record type.
- PHIGS posted structure type (PS) is mapped to an Ada record type.
- PHIGS bounding range types (B) are mapped to Ada record types.
- PHIGS colour specification type (CLR) is mapped to an Ada record type.
- PHIGS chromaticity coefficient type (CC) is mapped to an Ada record type.
- PHIGS connection identifier type (C) is mapped to the Ada STRING type.
- PHIGS file types (F) are mapped to Ada STRING types.
- PHIGS workstation type (W) is mapped to an Ada integer type.
- PHIGS modelling clipping volume type (MCV) is not used by the binding. Implementations can map this type to an implementation specific private type.
- PHIGS generalized drawing primitive identifier types (G2, G3) are mapped to Ada integer types.
- PHIGS generalized structure element identifier type (GS) is mapped to an Ada integer type.
- PHIGS archive file identifier type (AI) is mapped to an Ada integer type.
- PHIGS pick identifier type (PI) is mapped to an Ada integer type.
- PHIGS escape identifier type (EI) is mapped to an Ada integer type.
- PHIGS function name type (FN) is mapped to an Ada type providing for variable length strings.

- PHIG workstation identifier type (WI) is mapped to an Ada integer type.
- PHIGS tuples are mapped to Ada record types.
- PHIGS matrices are mapped to Ada array types.
- PHIGS arrays are mapped to either an unconstrained Ada array type, or to a record type providing for variable length arrays.
- PHIGS lists are mapped to an Ada private type declared in the generic PHIGS_LIST_UTILITIES package.
- PHIGS set types are mapped to Ada private types with generic accessing functions provided as necessary.
- PHIGS data records are mapped to Ada private types. In some cases, a set of subprograms for operating on the data records is explicitly defined by this binding. This is because the content and structure of the data record is implementation dependent. An implementation of PHIGS may provide other subprograms for manipulating implementation dependent data records.

Additional types used by the binding are declared as needed in a manner compatible with the PHIGS types.

3.2.5 Multi-tasking

The Ada language definition provides explicit support for concurrency. The Ada tasking model includes facilities for declaring and allocating tasks, and operations allowing intertask communication and synchronization.

The PHIGS standard, and hence this binding standard, neither requires nor prohibits an implementation from protecting against problems which could arise from asynchronous access to the PHIGS data structures from concurrent tasks. Implementors of PHIGS should provide information in the user's documentation regarding whether protection against such problems is implemented.

Annex D contains guidelines for implementors who want to support multi-tasking application programs. This annex does not form an integral part of the binding standard, but provides additional information.

3.2.6 Packaging

The PHIGS standard defines all of its graphic functionality as a cohesive whole. An implementation of PHIGS shall implement the entire functionality of PHIGS. To support this concept, this binding defines a single Ada package which corresponds to all of the PHIGS functionality. This package is named

```
package PHIGS is ... end PHIGS;
```

Associated with this package is a data type package which provides the type declarations as defined in 4.2 and the exception defined in 4.3. This package is

```
package PHIGS_TYPES is ... end PHIGS_TYPES;
```

A minimal program referencing PHIGS is shown below.

```
with PHIGS,
     PHIGS_TYPES;

procedure APPLICATION is
begin
```

```
    null;
  end APPLICATION;
```

Several additional Ada packages are defined in this binding. These packages are

- generic package PHIGS_COORDINATE_SYSTEM
- generic package PHIGS_LIST_UTILITIES
- package PHIGS_NAME_SET_FACILITY
- package PHIGS_CONFIGURATION

These packages support the declaration types in the PHIGS_TYPES package described above. PHIGS_COORDINATE_SYSTEM is a generic package which defines an assortment of types supporting each of the PHIGS coordinate systems. PHIGS_LIST_UTILITIES is also a generic package which provides type declarations and operations for list types which correspond to the PHIGS list types. The PHIGS_NAME_SET_FACILITY package defines private types for name sets and provides functions for building and manipulating name sets. The PHIGS_CONFIGURATION package sets implementation dependent limit values for various types defined in this binding. The PHIGS_CONFIGURATION package also provides for possible application modification of these limits.

3.2.7 Application Program Environment

An application program utilizing an Ada implementation of PHIGS will need to be aware of the environment in which both PHIGS and the application program(s) reside.

One aspect of the environment is the Ada program library. The Ada language requires that the application program have access to the program library in which the PHIGS software resides. The Ada standard ISO 8652 does not specify whether there is a single library or multiple libraries, or how access to the libraries is granted or managed. The user's documentation for the PHIGS implementation should specify where the PHIGS library exists in the system, and how access to the library is acquired.

Input/Output interfaces are also implementation-dependent, and are required to be described in the user's documentation. Besides the obvious graphics device interface information, interfaces to the file system shall be included in the documentation. Specifically, this includes the interface to the PHIGS error file, archive files, and metafile storage.

Subclause 3.2.2 discusses implications about the application program environment which may also apply to application programs.

3.2.8 Registration[1]

PHIGS reserves certain value ranges for registration as graphical items. The registered graphical items will be bound to Ada (and other programming languages). The registered item bindings will be consistent with the binding presented in this document.

---

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 JTC 1 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.

4 Tables

4.1 Abbreviations used in procedure names

ASF	aspect source flag
CHAR	character
ESC	escape
GDP	generalized drawing primitive
GSE	generalized structure element
HLHSR	hidden line/hidden surface removal
INQ	inquire
PHIGS	Programmer's Hierarchical Interactive Graphics System
U	unregistered
WS	workstation

4.1.1 List of procedures using the abbreviations

ASF	SET_INDIVIDUAL_ASF
CHAR	SET_ANNOTATION_TEXT_CHAR_HEIGHT SET_ANNOTATION_TEXT_CHAR_UP_VECTOR SET_CHAR_EXPANSION_FACTOR SET_CHAR_HEIGHT SET_CHAR_SPACING SET_CHAR_UP_VECTOR
ESC	PHIGS_ESCAPE.GENERALIZED_ESC PHIGS_ESC_<name of the escape procedure>.ESC PHIGS_UESC_<name of the escape procedure>.ESC
GDP	INQ_GDP (2D) INQ_GDP (3D) INQ_LIST_OF_AVAILABLE_GDP (2D) INQ_LIST_OF_AVAILABLE_GDP (3D) PHIGS_GDP.GENERALIZED_GDP PHIGS_GDP_<name of the GDP procedure>.GDP PHIGS_UGDP_<name of the GDP procedure>.GDP
GSE	INQ_GSE_FACILITIES INQ_LIST_OF_AVAILABLE_GSE PHIGS_GSE.GENERALIZED_GSE PHIGS_GSE_<name of the GSE procedure>.GSE PHIGS_UGSE_<name of the GSE procedure>.GSE
HLHSR	INQ_HLHSR_FACILITIES INQ_HLHSR_MODE SET_HLHSR_IDENTIFIER SET_HLHSR_MODE
INQ	INQ_ALL_CONFLICTING_STRUCTURES

INQ_ANNOTATION_FACILITIES
INQ_ARCHIVE_FILES
INQ_ARCHIVE_STATE_VALUE
INQ_CHOICE_DEVICE_STATE (2D)
INQ_CHOICE_DEVICE_STATE (3D)
INQ_COLOUR_FACILITIES
INQ_COLOUR_MODEL
INQ_COLOUR_MODEL_FACILITIES
INQ_COLOUR_REPRESENTATION
INQ_CONFLICT_RESOLUTION
INQ_CONFLICTING_STRUCTURES_IN_NETWORK
INQ_CURRENT_ELEMENT_CONTENT
INQ_CURRENT_ELEMENT_TYPE_AND_SIZE
INQ_DEFAULT_CHOICE_DEVICE_DATA (2D)
INQ_DEFAULT_CHOICE_DEVICE_DATA (3D)
INQ_DEFAULT_DISPLAY_UPDATE_STATE
INQ_DEFAULT_LOCATOR_DEVICE_DATA (2D)
INQ_DEFAULT_LOCATOR_DEVICE_DATA (3D)
INQ_DEFAULT_PICK_DEVICE_DATA (2D)
INQ_DEFAULT_PICK_DEVICE_DATA (3D)
INQ_DEFAULT_STRING_DEVICE_DATA (2D)
INQ_DEFAULT_STRING_DEVICE_DATA (3D)
INQ_DEFAULT_STROKE_DEVICE_DATA (2D)
INQ_DEFAULT_STROKE_DEVICE_DATA (3D)
INQ_DEFAULT_VALUATOR_DEVICE_DATA (2D)
INQ_DEFAULT_VALUATOR_DEVICE_DATA (3D)
INQ_DISPLAY_SPACE_SIZE (2D)
INQ_DISPLAY_SPACE_SIZE (3D)
INQ_DISPLAY_UPDATE_STATE
INQ_DYNAMICS_OF_STRUCTURES
INQ_DYNAMICS_OF_WS_ATTRIBUTES
INQ_EDGE_FACILITIES
INQ_EDGE_REPRESENTATION
INQ_EDIT_MODE
INQ_ELEMENT_CONTENT
INQ_ELEMENT_POINTER
INQ_ELEMENT_TYPE_AND_SIZE
INQ_ERROR_HANDLING_MODE
INQ_GDP (2D)
INQ_GDP (3D)
INQ_GSE_FACILITIES
INQ_HIGHLIGHTING_FILTER
INQ_HLHSR_FACILITIES
INQ_HLHSR_MODE
INQ_INPUT_QUEUE_OVERFLOW
INQ_INTERIOR_FACILITIES
INQ_INTERIOR_REPRESENTATION
INQ_INVISIBILITY_FILTER
INQ_LIST_OF_AVAILABLE_GDP (2D)
INQ_LIST_OF_AVAILABLE_GDP (3D)
INQ_LIST_OF_AVAILABLE_GSE
INQ_LIST_OF_AVAILABLE_WS_TYPES
INQ_LIST_OF_COLOUR_INDICES
INQ_LIST_OF_EDGE_INDICES
INQ_LIST_OF_INTERIOR_INDICES
INQ_LIST_OF_PATTERN_INDICES
INQ_LIST_OF_POLYLINE_INDICES
INQ_LIST_OF_POLYMARKER_INDICES
INQ_LIST_OF_TEXT_INDICES
INQ_LIST_OF_VIEW_INDICES
INQ_LOCATOR_DEVICE_STATE (2D)
INQ_LOCATOR_DEVICE_STATE (3D)
INQ_MODELLING_CLIPPING_FACILITIES
INQ_MORE_SIMULTANEOUS_EVENTS
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
INQ_NUMBER_OF_DISPLAY_PRIORITIES_SUPPORTED
INQ_OPEN_STRUCTURE
INQ_PATHS_TO_ANCESTORS
INQ_PATHS_TO_DESCENDANTS
INQ_PATTERN_FACILITIES
INQ_PATTERN_REPRESENTATION