INTERNATIONAL STANDARD

ISO/IEC 12088-4

First edition
1995-12-15

Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings —

Part 4: C

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/st

Technologies de l'information — Infographie et traitement de l'image — Traitement et échange de l'image — Liants de langage d'interface de programme d'application —

Partie 4: C

Reference number
ISO/IEC 12088-4:1995(E)

Contents

© ISO/IEC 1995
All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including protocopying and microfilm, without permission in writing from the publisher.

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

Printed in Switzerland

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995

Annexes

iTeh STANDARD PREVIEW
(https://standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360e3-6aa8-4843-94fe-afaf0aa0371/iso-iec-12088-4-1995

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

Contents

List of figures

List of tables

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

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 12088-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

ISO/IEC 12088 consists of the following part, under the general title *Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings*:

— *Part 4: C*

Other parts may follow.

Annexes A to D of this part of ISO/IEC 12088 are for information only.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

Introduction

The Image Processing and Interchange (IPI) functional specification, ISO/IEC 12087, upon which this binding is based, emerged as an International Standard in 1994. It consists of three parts: Part 1: Common Architecture for Imaging (IPI-CAI). Part 2: Programmer's Imaging Kernel System (IPI-PIKS) Application Program Interface and Part 3: Image Interchange Facility (IPI-IIF).

The functional description of ISO/IEC 12088 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 12088 is to define a standard binding for the Image Processing and Interchange Standard in the C programming language.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings —

Part 4:
C

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

1 Scope

ISO/IEC 12087 consists of the three parts which define the functional aspects of this part of ISO/IEC 12088. The Common Architecture for Imaging (IPI-CAI) defines the overall architecture. The Programmer's Imaging Kernel System (IPI-PIKS) and the Image Interchange Facility (IPI-IIF) each specify a language independent, image processing Application Program Interface (API) within the Image Processing and Interchange Standard. Either API may be implemented independently or both may be combined in one implementation. For integration into a programming language, IPI-PIKS and IPI-IIF APIs are embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 12088 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 12088. 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 12088 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.

[1] ISO/IEC 12087-1:1995, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 1: Common architecture for imaging.*

[2] ISO/IEC 12087-2:1994, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 2: Programmer's imaging kernel system application programme interface.*

[3] ISO/IEC 12087-3:1995, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 3: Imaging Interchange Facility (IIF).*

[4] ISO/IEC 9899:1990, *Programming Languages - C.*

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

3 The C language binding of the Image Processing and Interchange Standard

3.1 Classification and designation

This part of ISO/IEC 12088 incorporates the rules of conformance defined in the Image Processing and Interchange Standard (ISO/IEC 12087) for implementations. with those additional requirements specifically defined for C language implementations of the Standard.

The following criteria shall determine conformance of an implementation of this part of ISO/IEC 12088:

In order to conform, development of a C language implementation of the Image Processing and Interchange Standard shall implement one or more specific profiles as specified in the functional specification of ISO/IEC 12087. Independent implementation of the Programmer's Imaging Kernel System (Part 2 of ISO/IEC 12087 including relevant portions of Part 1) or the Image Interchange Facility (Part 3 of ISO/IEC 12087 including relevant portions of Part 1) is allowed.

The implementation shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 12088 for that same level of ISO/IEC 12087 and all lower levels and for a specific level of the C language. Thus. for example. the syntax of the function names shall be precisely as specified in the binding and parameters shall be of the data types stated in the binding.

This part of ISO/IEC 12088 supports the conformance profiles defined in the functional specification of ISO/IEC 12087.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
https://standards.iteh.ai/catalog/standards/sist/16a360c5-6aa8-4843-94fe-affaf0aa0371/iso-iec-12088-4-1995

3.2 Functions versus macros

An implementation may substitute macros for functions. Implementation by macro should result in the same effect as implementation by function including any side effects.

3.3 Implications of the language

3.3.1 Character strings

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

3.3.2 Implementation dependencies

There are a number of implementation issues which will affect the portability of application programs employing this part of ISO/IEC 12087. The application programmer should follow accepted practices for ensuring the portability of C language programs to avoid introducing problems when recompiling the application on another system.

This part of ISO/IEC 12088 attempts to avoid dependencies on computer specific types which may vary among computers. Clause 5 recommends type mappings for common numeric types, data object identifiers, and precision to be used throughout this C language binding.

3.3.3 Data object repository

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 12088-4:1995
12088-4-1995

The Image Processing and Interchange Standard provides a repository of data objects for routinely used objects. These objects are defined in ISO/IEC 12087. Access to these objects for their assignment to IPI-PIKS operators is by IPI-PIKS-internal identifiers which may be acquired by the execution of a *return_repository_id* function presented in 7.2 of this part of ISO/IEC 12088.

The repository object identifiers are defined in 6.5. Each repository object name begins with the characters "IR_".

3.4 Identifier mapping

The IPI-PIKS-API and IPI-IIF-API function names are mapped to C identifiers beginning with the sentinel "I" followed by a lower case name prefix which identifies the return parameter type. Words and phrases used in the function names are concatenated with and beyond the prefix and may be abbreviated in the name representation. The acronyms and the first letter of name words and abbreviations are in upper case and the remainder are in lower case. For example, the identifier of the IPI-PIKS *import_image* function, which returns an IPI-PIKS internal object identifier, is *InImportImage*.

Data type element names are mapped to C identifiers beginning with the sentinel "I." The sentinel is followed by a two-part prefix, which delineates the usage and type of the element, followed by a name. These usage and data type designators, see Table 1, include "m" for external physical image pixel representation, "p" for parameter representation and "d" for internal representation. For some data types, the type designator and name may be merged, e.g., *Ipint*, the integer basic type. Acronyms, words and phrases used in the type names are in lower case, often abbreviated in the representation, and separated with the underscore character, "_". For example, the union of pixel data types used for parameters is *Ipsparameter_pixel*.

Enumerated data type constants and macro names are mapped to C identifiers beginning with the sentinel "I" followed by a word or abbreviation of the entity to which the identifier belongs and the name of the identifier in upper case with acronyms, abbreviations or words separated with the underscore character, "_". For example the macro name for Boolean false is *IBOOL_FALSE*.

Parameters are mapped to C identifiers beginning with a prefix which identifies the type and/or usage. Words and phrases used in the parameter and variable names are concatenated with and beyond the prefix and may be abbreviated in the representation. The first letter of name words and abbreviations are in upper case and the remainder are in lower case. Acronyms are presented in upper case. For example, the name of the function parameter which denotes the identifier of a destination ROI object is *nDestROI*.

The prefix symbols used in this part of ISO/IEC 12088 are listed in Table 1.

Table 1 - Data type prefixes

Prefix	Meaning
a	array
b	Boolean
c	character
d	internal data type
e	enumerated data type
f	function
i	integer
m	external image data type
n	identifier
p	parameter type
r	real
s	structure
t	pointer
u	unsigned integer
v	void
x	complex
z	zero terminated string

The C language standard 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. Implementations which must run in environments which accept less than 31 characters in external identifiers must include a set of macros (#defines) in the header file which equate the long names as defined in this document to a set of shorter names which conform to the requirements of the implementation. An example of these shorter 6-character names can be found in Annex D

3.5 Return values

Functional elements which perform a status test within the IPI-PIKS internal domain, and do not affect destination data, output the Boolean result of the test as return value.

Destination identifiers or values are output as return values from elements which include destination objects or values, respectively, as part of their parameter list, even if the identifier itself is input to the element. If more than one destination object is identified by a function, no return value is output. Return values are void from elements which do not provide destination objects or values.

If the destination identifier, output as a return value, is generated by the functional element, the parameter represented by the return value is not included in the argument list of the functional element. Conversely, if the same parameter appears in the argument list, the return value is a replication of an input of the destination identifier, though the contents of the object may be modified by the function.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

3.6 Header files

ISO/IEC 12088-4:1995

The C language provides a mechanism to allow external files to be included in a compilation. Clause 5 of this part of ISO/IEC 12088 describes the data types that shall be defined in the piks.h and iif.h which should be included in any application program that intends to use IPI-PIKS and/or IPI-IIF, respectively, via the C language binding. Additional implementation-dependent items may be placed in these files if needed and the files may be concatenated as one header file. These items should start with the sentinel "I", as far as applicable.

The files piks.h and iif.h shall also contain external prototypes for all IPI-PIKS and IPI-IIF C language functions. For example, the prototype for the IPI-PIKS function *InResize* would appear as follows:

```
extern Idnimage InResize (                  /* OUT destination image identifier          */
    Idnimage        nSourceImage,           /* source image identifier                   */
    Idnimage        nDestImage,             /* destination image identifier              */
    Idntuple        nDestSize               /* destination size ND 5-tuple identifier    */
);
```

Examples of piks.h and iif.h which utilize suggested data types are presented in Annex B.

3.7 Memory management

IPI-PIKS requires internal memory space, denoted by identifiers containing the letter "d" in their prefixes (immediately following the sentinel) as they appear in this part of ISO/IEC 12088. Internal memory requirements may exist in a memory domain independent of application memory or may be acquired from application memory as provided in a specific implementation. ISO/IEC 12087 and this part of ISO/IEC 12088 do not provide functionality to support the allocation and management of IPI-PIKS internal memory should it be acquired from the application domain.

Both IPI-IIF and IPI-PIKS require external memory space, denoted by identifiers containing the letter "p" in their prefixes (immediately following the sentinel) as they appear in this part of ISO/IEC 12088. External memory requirements must be acquired from application memory space. The application shall allocate the external memory for the data provided to and returned by the implementation of the Image Processing and Interchange Standard. The application will allocate C structures and pass the appropriate identifiers of those structures to the requiring functional element, which will then place information into or receive information from the structures. However, the application may provide data buffered in partial amounts for large data objects input to the functions. Also, some functions return variable length data, the length of which is not known a priori and can not be computed by the application.

If the memory space requirement is small, even if variable in length, the structure and sufficient space must be provided by the application. As example, a B-tuple for colour image data precision specification will contain no more than four data elements and, as such, is a small structure. A B-tuple for spectral image data precision specification may contain many elements.

Functions which may receive buffered data and functions which return variable lengths of sizable data or are considered to return potentially large amounts of fixed-length data are managed such that the application can submit or receive buffered portions of the data. The data portion, or buffer, is a subset of the implementation's internal data structure and is called the application's buffer. This allows the application to process the implementation's data, considered as a single dimension array, in a piecewise manner rather than all at once.

The application allocates the memory for its buffer and passes an identifier of that buffer to the implementation. The implementation places the subset of data into the buffer according to the specifications included among the function's parameters. In order to support this policy of memory management, three additional parameters have been added to functions which may transfer buffered data:

a) iApplBufferSize: An integer input parameter which specifies the length of the application's buffer, dimensioned in terms of the number of data elements of data type specified for the input or return. The positive value of iApplBufferSize indicates the element count of the data which can be contained in the application's buffer. A value of 0 is valid as input for this parameter and is useful to interrogate the overall data size available for output or available in the implementation for input of each buffered transfer operation. A negative value is valid for this parameter and is interpreted as a signal for complete transfer of the data, the application being responsible to provide adequate space for the complete data transfer from implementation to application or the complete data requirement for transfer from application to implementation.

b) uStartPoint: An unsigned integer input parameter which is an element index offset into the implementation's data space represented as a 1D array of data type specified for the external data being transferred. uStartPoint of 0 represents the first element of both the implementation's and application's data. Otherwise, uStartPoint is the index which indicates the element offset in the implementation's data space where data transfer is to be initiated to or from the beginning of the application's buffer. The transfer of data continues to the extent of the application's buffer or until there are no more items to be transferred, whichever is least.