TECHNICAL SPECIFICATION

ISO/IEC TS 18661-5

First edition
2016-08-15

Information Technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C —

Part 5: Supplementary attributes

Technologies de l'information — Langages de programmation, leurs environnements et interfaces du logiciel système — Extensions à virgule flottante pour C —

Partie 5: Attributs supplémentaires

Reference number
ISO/IEC TS 18661-5:2016(E)

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments, and system software interfaces*.

ISO/IEC TS 18661 consists of the following parts, under the general title *Information technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C*:

— *Part 1: Binary floating-point arithmetic*

— *Part 2: Decimal floating-point arithmetic*

— *Part 3: Interchange and extended types*

— *Part 4: Supplementary functions*

— *Part 5: Supplementary attributes*

ISO/IEC TS 18661-1 updates ISO/IEC 9899:2011, *Information technology — Programming Language C*, annex F in particular, to support all required features of ISO/IEC/IEEE 60559:2011, *Information technology — Microprocessor Systems — Floating-point arithmetic*.

ISO/IEC TS 18661-2 supersedes ISO/IEC TR 24732:2009, *Information technology — Programming languages, their environments and system software interfaces — Extension for the programming language C to support decimal floating-point arithmetic*.

ISO/IEC TS 18661-3, ISO/IEC TS 18661-4, and ISO/IEC TS 18661-5 specify extensions to ISO/IEC 9899:2011 for features recommended in ISO/IEC/IEEE 60559:2011.

Introduction

Background

IEC 60559 floating-point standard

The IEEE 754-1985 standard for binary floating-point arithmetic was motivated by an expanding diversity in floating-point data representation and arithmetic, which made writing robust programs, debugging, and moving programs between systems exceedingly difficult. Now the great majority of systems provide data formats and arithmetic operations according to this standard. The IEC 60559:1989 international standard was equivalent to the IEEE 754-1985 standard. Its stated goals were the following:

1 Facilitate movement of existing programs from diverse computers to those that adhere to this standard.

2 Enhance the capabilities and safety available to programmers who, though not expert in numerical methods, may well be attempting to produce numerically sophisticated programs. However, we recognize that utility and safety are sometimes antagonists.

3 Encourage experts to develop and distribute robust and efficient numerical programs that are portable, by way of minor editing and recompilation, onto any computer that conforms to this standard and possesses adequate capacity. When restricted to a declared subset of the standard, these programs should produce identical results on all conforming systems.

4 Provide direct support for

a. Execution-time diagnosis of anomalies

b. Smoother handling of exceptions

c. Interval arithmetic at a reasonable cost

5 Provide for development of

a. Standard elementary functions such as exp and cos

b. Very high precision (multiword) arithmetic

c. Coupling of numerical and symbolic algebraic computation

6 Enable rather than preclude further refinements and extensions.

To these ends, the standard specified a floating-point model comprising the following:

— *formats* – for binary floating-point data, including representations for Not-a-Number (NaN) and signed infinities and zeros

— *operations* – basic arithmetic operations (addition, multiplication, etc.) on the format data to compose a well-defined, closed arithmetic system; also specified conversions between floating-point formats and decimal character sequences, and a few auxiliary operations

— *context* – status flags for detecting exceptional conditions (invalid operation, division by zero, overflow, underflow, and inexact) and controls for choosing different rounding methods

The ISO/IEC/IEEE 60559:2011 international standard is equivalent to the IEEE 754-2008 standard for floating-point arithmetic, which is a major revision to IEEE 754-1985.

The revised standard specifies more formats, including decimal as well as binary. It adds a 128-bit binary format to its basic formats. It defines extended formats for all of its basic formats. It specifies data interchange formats (which may or may not be arithmetic), including a 16-bit binary format and an unbounded tower of wider formats. To conform to the floating-point standard, an implementation must provide at least one of the basic formats, along with the required operations.

The revised standard specifies more operations. New requirements include – among others – arithmetic operations that round their result to a narrower format than the operands (with just one rounding), more conversions with integer types, more classifications and comparisons, and more operations for managing flags and modes. New recommendations include an extensive set of mathematical functions and seven reduction functions for sums and scaled products.

The revised standard places more emphasis on reproducible results, which is reflected in its standardization of more operations. For the most part, behaviors are completely specified. The standard requires conversions between floating-point formats and decimal character sequences to be correctly rounded for at least three more decimal digits than is required to distinguish all numbers in the widest supported binary format; it fully specifies conversions involving any number of decimal digits. It recommends that transcendental functions be correctly rounded.

The revised standard requires a way to specify a constant rounding direction for a static portion of code, with details left to programming language standards. This feature potentially allows rounding control without incurring the overhead of runtime access to a global (or thread) rounding mode.

Other features recommended by the revised standard include alternate methods for exception handling, controls for expression evaluation (allowing or disallowing various optimizations), support for fully reproducible results, and support for program debugging.

The revised standard, like its predecessor, defines its model of floating-point arithmetic in the abstract. It neither defines the way in which operations are expressed (which might vary depending on the computer language or other interface being used), nor does it define the concrete representation (specific layout in storage, or in a processor's register, for example) of data or context, except that it does define specific encodings that are to be used for the exchange of floating-point data between different implementations that conform to the specification.

IEC 60559 does not include bindings of its floating-point model for particular programming languages. However, the revised standard does include guidance for programming language standards, in recognition of the fact that features of the floating-point standard, even if well supported in the hardware, are not available to users unless the programming language provides a commensurate level of support. The implementation's combination of both hardware and software determines conformance to the floating-point standard.

C support for IEC 60559

The C standard specifies floating-point arithmetic using an abstract model. The representation of a floating-point number is specified in an abstract form where the constituent components (sign, exponent, significand) of the representation are defined but not the internals of these components. In particular, the exponent range, significand size, and the base (or radix) are implementation-defined. This allows flexibility for an implementation to take advantage of its underlying hardware architecture. Furthermore, certain behaviors of operations are also implementation-defined, for example in the area of handling of special numbers and in exceptions.

The reason for this approach is historical. At the time when C was first standardized, before the floating-point standard was established, there were various hardware implementations of floating-point arithmetic in common use. Specifying the exact details of a representation would have made most of the existing implementations at the time not conforming.

Beginning with ISO/IEC 9899:1999 (C99), C has included an optional second level of specification for implementations supporting the floating-point standard. C99, in conditionally normative annex F, introduced nearly complete support for the IEC 60559:1989 standard for binary floating-point arithmetic. Also, C99's informative annex G offered a specification of complex arithmetic that is compatible with IEC 60559:1989.

ISO/IEC 9899:2011 (C11) includes refinements to the C99 floating-point specification, though it is still based on IEC 60559:1989. C11 upgraded annex G from "informative" to "conditionally normative".

ISO/IEC TR 24732:2009 introduced partial C support for the decimal floating-point arithmetic in ISO/IEC/IEEE 60559:2011. ISO/IEC TR 24732, for which technical content was completed while IEEE 754-2008 was still in the later stages of development, specifies decimal types based on ISO/IEC/IEEE 60559:2011 decimal formats, though it does not include all of the operations required by ISO/IEC/IEEE 60559:2011.

Purpose

The purpose of ISO/IEC TS 18661 is to provide a C language binding for ISO/IEC/IEEE 60559:2011, based on the C11 standard, that delivers the goals of ISO/IEC/IEEE 60559 to users and is feasible to implement. It is organized into five parts.

ISO/IEC TS 18661-1 provides changes to C11 that cover all the requirements, plus some basic recommendations, of ISO/IEC/IEEE 60559:2011 for binary floating-point arithmetic. C implementations intending to support ISO/IEC/IEEE 60559:2011 are expected to conform to conditionally normative annex F as enhanced by the changes in ISO/IEC TS 18661-1.

ISO/IEC TS 18661-2 enhances ISO/IEC TR 24732 to cover all the requirements, plus some basic recommendations, of ISO/IEC/IEEE 60559:2011 for decimal floating-point arithmetic. C implementations intending to provide an extension for decimal floating-point arithmetic supporting ISO/IEC/IEEE 60559:2011 are expected to conform to ISO/IEC TS 18661-2.

ISO/IEC TS 18661-3 (Interchange and extended types), ISO/IEC TS 18661-4 (Supplementary functions), and ISO/IEC TS 18661-5 (Supplementary attributes) cover recommended features of ISO/IEC/IEEE 60559:2011. C implementations intending to provide extensions for these features are expected to conform to the corresponding parts.

Additional background on supplementary attributes

ISO/IEC/IEEE 60559:2011 defines alternatives for certain attributes of floating-point semantics and intends that programming languages provide means by which a program can specify which of the alternative semantics apply to a given block of code. The program specification of attributes is to be constant (fixed at translation time), not dynamic (changeable at execution time).

ISO/IEC TS 18661 provides these attributes by means of standard pragmas, where the pragma parameters represent the alternative semantics.

The **`FENV_ROUND`** and **`FENV_DEC_ROUND`** pragmas, specified in ISO/IEC TS 18661-1 and ISO/IEC TS 18661-2, respectively, provide the rounding direction attributes required by ISO/IEC/IEEE 60559:2011.

ISO/IEC/IEEE 60559:2011 recommends attributes for

— alternate exception handling: methods of handling floating-point exceptions

— preferredWidth: evaluation formats for floating-point operations

— value-changing optimizations: allow/disallow program transformations that might affect floating-point result values

— reproducibility: support for getting floating-point result values and exceptions that are exactly reproducible on other systems

This part of ISO/IEC TS 18661 specifies pragmas that provide these attributes.

Note that the pragmas introduced by ISO/IEC TS 18661 are similar in form to the floating-point pragmas (**FENV_ACCESS**, **FP_CONTRACT**, **CX_LIMITED_RANGE**) that are already in ISO/IEC 9899.

Information technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C —

Part 5: Supplementary attributes

1 Scope

This part of ISO/IEC TS 18661 extends programming language C to include support for attributes specified and recommended in ISO/IEC/IEEE 60559:2011.

2 Conformance

An implementation conforms to this part of ISO/IEC TS 18661 if

a) it meets the requirements for a conforming implementation of C11 with all the changes to C11 as specified in parts 1-5 of ISO/IEC TS 18661;

b) it conforms to ISO/IEC TS 18661-1 or ISO/IEC TS 18661-2 (or both); and

c) it defines `__STDC_IEC_60559_ATTRIBS__` to `201607L`.

An implementation conforms to the optional specification for alternate exception handling in this part of ISO/IEC TS 18661 if, in addition to the above,

d) it defines `__STDC_IEC_60559_ATTRIB_ALTERNATE_EXCEPTION_HANDLING__` to `201607L`.

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9899:2011, *Information technology — Programming languages — C*

ISO/IEC/IEEE 60559:2011, *Information technology — Microprocessor Systems — Floating-point arithmetic*

ISO/IEC TS 18661-1:2014, *Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 1: Binary floating-point arithmetic*

ISO/IEC TS 18661-2:2015, *Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 2: Decimal floating-point arithmetic*

ISO/IEC TS 18661-3:2015, *Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 3: Interchange and extended types*

ISO/IEC TS 18661-4:2015, *Information Technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C — Part 4: Supplementary functions*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9899:2011, ISO/IEC/IEEE 60559:2011, ISO/IEC TS 18661-1:2014, ISO/IEC TS 18661-2:2015, ISO/IEC TS 18661-3:2015, ISO/IEC TS 18661-4:2015, and the following apply.

4.1

C11
standard ISO/IEC 9899:2011, *Information technology — Programming languages C*, including *Technical Corrigendum 1* (ISO/IEC 9899:2011/Cor. 1:2012)

5 C standard conformance

5.1 Freestanding implementations

The specification in C11 + TS18661-1 + TS18661-2 allows freestanding implementations to conform to this part of ISO/IEC TS 18661.

5.2 Predefined macros

Change to C11 + TS18661-1 + TS18661-2 + TS18661-3 + TS18661-4:

In 6.10.8.3#1, add:

> **`__STDC_IEC_60559_ATTRIBS__`** The integer constant **`201607L`**, intended to indicate support of attributes specified and recommended in IEC 60559.
>
> **`__STDC_IEC_60559_ATTRIB_ALTERNATE_EXCEPTION_HANDLING__`** The integer constant **`201607L`**, intended to indicate support of the alternate exception handling attributes specified and recommended in IEC 60559.

6 Standard pragmas

C11 provides standard pragmas (6.10.6) for specifying certain attributes pertaining to floating-point behavior within a compound statement or file. This part of ISO/IEC TS 16881 extends this practice by introducing additional standard pragmas to support the attributes recommended by IEC 60559.

Change to C11 + TS18661-1 + TS18661-2 + TS18661-3 + TS18661-4:

In 5.1.2.3#5, append the sentence:

> Behaviors affected by the standard pragmas (6.10.6) are unspecified in the handler and after the handler exits.

In 6.10.6#2, append to the list of standard pragmas:

```
#pragma STDC FENV_FLT_EVAL_METHOD width
#pragma STDC FENV_DEC_EVAL_METHOD width
#pragma STDC FENV_ALLOW_VALUE_CHANGING_OPTIMIZATION on-off-switch
#pragma STDC FENV_ALLOW_ASSOCIATIVE_LAW on-off-switch
#pragma STDC FENV_ALLOW_DISTRIBUTIVE_LAW on-off-switch
#pragma STDC FENV_ALLOW_MULTIPLY_BY_RECIPROCAL on-off-switch
#pragma STDC FENV_ALLOW_ZERO_SUBNORMAL on-off-switch
#pragma STDC FENV_ALLOW_CONTRACT_FMA on-off-switch
#pragma STDC FENV_ALLOW_CONTRACT_OPERATION_CONVERSION on-off-switch
#pragma STDC FENV_ALLOW_CONTRACT on-off-switch
#pragma STDC FENV_REPRODUCIBLE on-off-switch
#pragma STDC FENV_EXCEPT action except-list
```

width: specified with the pragmas (7.6.1c, 7.6.1d)

action, *except-list*: specified with the pragma (7.6.1g.1)

7 Evaluation formats

IEC 60559 recommends attributes for specifying a preferred width for operation results. These preferred widths correspond to the evaluation formats defined in C11, though C11 does not provide means for the user to control the evaluation format. This part of ISO/IEC TS 16881 provides pragmas in **<fenv.h>** to control the evaluation format, using constants with the values of the **FLT_EVAL_METHOD** and **DEC_EVAL_METHOD** macros (5.2.4.2.2a) to represent the evaluation formats.

The evaluation methods in C11 apply to floating-point operators, but not to math functions. Hence, they do not apply to the IEC 60559 operations that are provided as library functions. This clause specifies a macro the user can define to cause the generic macros in **<tgmath.h>** to be evaluated like floating-point operators.

Changes to C11 + TS18661-1 + TS18661-2 + TS18661-3 + TS18661-4:

After 5.2.4.2.2a#3, insert:

> [3a] The **FLT_EVAL_METHOD** and **DEC_EVAL_METHOD** macros characterize the use of evaluation formats at the point in the program where the macro is used. Thus, the values of these macros reflect the state of any evaluation method pragmas (7.6.1c, 7.6.1d) that are in effect. These macros shall not be used in a **#if** or **#elif** expression within the scope of a corresponding evaluation method pragma.

After 7.6.1b, insert:

7.6.1c Evaluation method pragma

Synopsis

[1]
```
#define __STDC_WANT_IEC_60559_ATTRIBS_EXT__
#include <fenv.h>
#pragma STDC FENV_FLT_EVAL_METHOD width
```

Description

[2] The **FENV_FLT_EVAL_METHOD** pragma sets the evaluation method for standard floating types and for binary interchange and extended floating types to the evaluation method represented by *width*. The parameter *width* is an expression in one of the forms

> **0**
> *decimal-constant*
> **–** *decimal-constant*

or

> **DEFAULT**

where the value of an expression is a possible value of the **FLT_EVAL_METHOD** macro, as specified in 5.2.4.2.2a. An expression represents the evaluation method corresponding to its value (5.2.4.2.2a) and **DEFAULT** designates the implementation's default evaluation method (characterized by the **FLT_EVAL_METHOD** macro where no **FENV_FLT_EVAL_METHOD** pragma is in effect). *width* may be **-1**, **0**, or **DEFAULT**. Which, if any, other values of *width* are supported is implementation-defined. Use of unsupported values of *width* results in undefined behavior. The pragma shall occur either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another **FENV_FLT_EVAL_METHOD** pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another **FENV_FLT_EVAL_METHOD** pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement.

7.6.1d Evaluation method pragma for decimal floating types

Synopsis

[1]
```
#define __STDC_WANT_IEC_60559_DFP_EXT__
#define __STDC_WANT_IEC_60559_ATTRIBS_EXT__
#include <fenv.h>
#pragma STDC FENV_DEC_EVAL_METHOD width
```

Description

[2] The **FENV_DEC_EVAL_METHOD** pragma sets the evaluation method for decimal interchange and extended floating types to the evaluation method represented by *width*. The parameter *width* is an expression in one of the forms

> **0**
> *decimal-constant*
> **–** *decimal-constant*

or

> **DEFAULT**

where the value of an expression is a possible value of the **DEC_EVAL_METHOD** macro, as specified in 5.2.4.2.2a. An expression represents the evaluation method corresponding to its