Technical Specification

ISO/IEC TS 18661-5

Second edition
2025-03

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

Part 5: Supplementary attributes

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 18661-5:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

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.

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 or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This second edition cancels and replaces the first edition (ISO/IEC TS 18661-5:2016), which has been technically revised.

The main changes are as follows:

— The specification has been updated to extend ISO/IEC 9899:2024.

— Conformance macros have been added to allow conformance to each of the four feature sets (evaluation formats, optimization controls, reproducibility, and alternate exception handling) independently.

A list of all parts in the ISO/IEC 18661 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

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 reliable programs, debugging and moving programs between systems exceedingly difficult. Now the great majority of systems provide data formats and arithmetic operations according to IEEE 754. Corresponding versions of IEEE 754 and ISO/IEC 60559 have equivalent content.

Support for IEEE 754-1985 was added in ISO/IEC 9899:1999 (also referred to as C99), and ISO/IEC 9899:2018 is still based on IEEE 754-1985. However, IEEE 754 underwent a major revision in 2008 and a minor revision in 2019, which added several new features.

The purpose of the ISO/IEC 18661 series (first published 2014 through 2016) has been to specify C language support for the new features introduced into IEEE 754 since 1985. Most of the ISO/IEC 18661 series has been incorporated into ISO/IEC 9899:2024 (also referred to as C23 because major work on this revision was completed in 2023), which supports all required and most recommended features in IEEE 754-2019.

IEEE 754 defines alternatives for certain attributes of floating-point semantics, and aims to provide, through programming languages, a 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 constant (fixed at translation time), not dynamic (changeable at execution time).

The `FENV_ROUND` and `FENV_DEC_ROUND` pragmas in C23 provide the rounding direction attributes required by IEEE 754.

IEEE 754 also recommends other attributes that are not supported in C23, including:

— preferredWidth: evaluation formats for floating-point operations;

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

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

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

To supplement the IEEE 754 support in C23, this document provides these recommended attributes by means of standard pragmas. The pragma parameters represent the alternative semantics. The pragmas are similar in form to the floating-point pragmas (`FENV_ACCESS`, `FP_CONTRACT`, `CX_LIMITED_RANGE`) that have been in C since 1999.

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

Part 5: Supplementary attributes

1 Scope

This document specifies extensions to programming language C to include pragmas corresponding to attributes specified and recommended in ISO/IEC 60559 but not supported in ISO/IEC 9899:2024 (also referred to as C23).

2 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 9899:2024, *Information technology — Programming languages — C*

ISO/IEC 60559:2020, *Information technology — Microprocessor Systems — Floating-Point arithmetic*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9899:2024 and ISO/IEC 60559:2020 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

4 Conformance

An implementation that meets the requirements for a conforming implementation of C23 may conform to any or all of the four feature sets in this document. The implementation conforms to the feature sets if

a) it defines `__STDC_IEC_60559_BFP__` or `__STDC_IEC_60559_DFP__` or both, indicating support for ISO/IEC 60559 binary or decimal floating-point arithmetic, as specified in C23, Annex F;

and one or more of the following are true:

b) it defines `__STDC_IEC_60559_ATTRIB_EVALUATION_FORMAT__` to `202401L` and provides the features for evaluation formats as specified in this document (Clause 7);

c) it defines `__STDC_IEC_60559_ATTRIB_OPTIMIZATION__` to `202401L` and provides the features for optimization as specified in this document (Clause 8);

d) it defines `__STDC_IEC_60559_ATTRIB_REPRODUCIBLE__` to `202401L` and provides the features for reproducibility as specified in this document (Clause 9);

e) it defines `__STDC_IEC_60559_ATTRIB_ALTERNATE_EXCEPTION_HANDLING__` to `202401L` and provides the features for alternate exception handling as specified in this document (Clause 10).

5 C standard conformance

5.1 Freestanding implementations

C23, Clause 4 allows freestanding implementations to conform to this document.

5.2 Predefined macros

The implementation defines one or more of the following macros to indicate conformance to the specification in this document for support of the corresponding attributes specified and recommended in ISO/IEC 60559.

`__STDC_IEC_60559_ATTRIB_EVALUATION_FORMAT__` The integer constant `202401L`.

`__STDC_IEC_60559_ATTRIB_OPTIMIZATION__` The integer constant `202401L`.

`__STDC_IEC_60559_ATTRIB_REPRODUCIBLE__` The integer constant `202401L`.

`__STDC_IEC_60559_ATTRIB_ALTERNATE_EXCEPTION_HANDLING__` The integer constant `202401L`.

5.3 Standard headers

The identifiers specified in this document are defined or declared by the associated header if and only if the implementation defines the relevant feature macros (5.2) and

`__STDC_WANT_IEC_60559_ATTRIB_EXT__`

is defined as a macro at the point in the source file where the header is first included.

6 Standard pragmas

C23 provides standard pragmas (C23, 6.10.8) for specifying certain attributes pertaining to floating-point behavior within a compound statement or file. This document extends this practice by introducing additional standard pragmas to support attributes recommended by ISO/IEC 60559:

```
#pragma STDC FP_FLT_EVAL_METHOD width
#pragma STDC FP_DEC_EVAL_METHOD width
#pragma STDC FP_ALLOW_VALUE_CHANGING_OPTIMIZATION on-off-switch
#pragma STDC FP_ALLOW_ASSOCIATIVE_LAW on-off-switch
#pragma STDC FP_ALLOW_DISTRIBUTIVE_LAW on-off-switch
#pragma STDC FP_ALLOW_MULTIPLY_BY_RECIPROCAL on-off-switch
#pragma STDC FP_ALLOW_ZERO_SUBNORMAL on-off-switch
#pragma STDC FP_ALLOW_CONTRACT_FMA on-off-switch
#pragma STDC FP_ALLOW_CONTRACT_OPERATION_CONVERSION on-off-switch
#pragma STDC FP_ALLOW_CONTRACT on-off-switch
#pragma STDC FP_REPRODUCIBLE on-off-switch
#pragma STDC FENV_EXCEPT action except-list
```

width: specified with the pragmas (7.2, 7.3)

on-off-switch: specified in C23, 6.10.8

action, *except-list*: specified with the pragma (10.1)

7 Evaluation formats

7.1 General

This clause applies to implementations that define:

```
__STDC_IEC_60559_ATTRIB_EVALUATION_FORMAT__
```

C23 gives implementations the flexibility to evaluate operations to the format of the wider operand or to a still wider evaluation format. The values of the macros `FLT_EVAL_METHOD` (C23, 5.3.5.3.3 and H.3) and `DEC_EVAL_METHOD` (C23, 5.3.5.3.4 and H.3) characterize these evaluation methods. Though C23 does not provide means for the user to control the evaluation method, some implementations provide such controls as extensions. ISO/IEC 60559 recommends an attribute for this purpose. The following subclauses (7.2 and 7.3) specify pragmas in `<math.h>` to control the evaluation method. These evaluation method pragmas, like the `FENV_ROUND` (C23, 7.6.3) and `FENV_DEC_ROUND` (C23, 7.6.4) pragmas, affect translation-time expression evaluation and constants.

NOTE As specified in C23, 6.7.2, the value of the initializer for an object declared with storage-class specifier `constexpr` is constrained to be exactly representable in the target type. Thus, an evaluation method pragma whose scope includes the declaration can affect the validity of the declaration.

7.2 Evaluation method pragma

Synopsis

```
#define __STDC_WANT_IEC_60559_ATTRIB_EXT__
#include <math.h>
#pragma STDC FP_FLT_EVAL_METHOD width
```

Constraints

The *width* parameter shall be `-1`, `0`, `DEFAULT`, or another value supported by the implementation.

Description

The `FP_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
DEFAULT
```

where the value of the expression is a possible value of the `FLT_EVAL_METHOD` macro, as specified in C23, 5.3.5.3.3 and H.3. An expression represents the evaluation method corresponding to its value (C23, 5.3.5.3.3 and H.3) and `DEFAULT` designates the implementation's default evaluation method (characterized by the `FLT_EVAL_METHOD` macro). The *width* parameter may be `-1`, `0`, or `DEFAULT`. Which, if any, other values of *width* are supported is implementation-defined. 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 `FP_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 `FP_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.

7.3 Evaluation method pragma for decimal floating types

Synopsis

```
#define __STDC_WANT_IEC_60559_ATTRIB_EXT__
#include <math.h>
#pragma STDC FP_DEC_EVAL_METHOD width
```

Constraints

The *width* parameter shall be `-1`, `1`, `DEFAULT`, or another value supported by the implementation.

Description

The `FP_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
DEFAULT
```

where the value of the expression is a possible value of the `DEC_EVAL_METHOD` macro, as specified in C23, 5.3.5.3.4 and H.3. An expression represents the evaluation method corresponding to its value (C23, 5.3.5.3.4 and H.3) and `DEFAULT` designates the implementation's default evaluation method (characterized by the `DEC_EVAL_METHOD` macro). The *width* parameter may be `-1`, `1`, or `DEFAULT`. Which, if any, other values of *width* are supported is implementation-defined. 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 `FP_DEC_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 `FP_DEC_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.

7.4 Effective evaluation method macros

The `<float.h>` macros `FLT_EVAL_METHOD` (C23, 5.3.5.3.3 and H.3) and `DEC_EVAL_METHOD` (C23, 5.3.5.3.4 and H.3) characterize the default evaluation method. Their values are constant expressions, suitable for use in conditional expression inclusion preprocessing directives. They are not affected by the evaluation method pragmas, so it is possible they do not reflect the effective evaluation method.

The `<math.h>` header defines macros `FLT_EVAL_METHOD_EFFECTIVE` and `DEC_EVAL_METHOD_EFFECTIVE` that are similar to the `<float.h>` macros `FLT_EVAL_METHOD` and `DEC_EVAL_METHOD`, except that they characterize the effective evaluation method at the point in the program where the macro is used. Thus, they reflect the state of any evaluation method pragmas (7.2, 7.3) that are in effect. These macros shall not be used in conditional expression inclusion preprocessing directives.

7.5 Evaluation type macros

The `<math.h>` types with an `_t` suffix (e.g. `float_t`) (C23, 7.12.1 and H.11), which are defined to match evaluation formats, reflect the evaluation method where no evaluation method pragma is in effect. For each of these types, there is a type-like macro in `<math.h>` with the same name which expands to a designation for the type whose range and precision are used for evaluating operations and constants of the corresponding standard, binary, or decimal floating type. The macro reflects the actual evaluation method, which can be determined by an evaluation method pragma. Use of `#undef` to remove the macro definition will ensure that the actual type is referred to (as though no evaluation method pragma was in effect).