PD ISO/IEC TS 18661-1:2014

BSI Standards Publication

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

Part 1: Binary floating-point arithmetic

bsi.

...making excellence a habit.™

Document
Document Sharing Hub

Full document details: PD ISO/IEC TS 18661-1:2014

National foreword

This Published Document is the UK implementation of ISO/IEC TS 18661-1:2014.

The UK participation in its preparation was entrusted to Technical Committee IST/5, Programming languages, their environments and system software interfaces.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014. Published by BSI Standards Limited 2014

ISBN 978 0 580 85414 9

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 August 2014.

Amendments issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 18661-1:2014

TECHNICAL SPECIFICATION

ISO/IEC TS 18661-1

First edition
2014-07-15

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

Part 1: Binary floating-point arithmetic

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

Partie 1: Arithmétique binaire en virgule flottante

Reference number
ISO/IEC TS 18661-1:2014(E)

Documents
Document Sharing Hub

© ISO/IEC 2014

Full document details: PD ISO/IEC TS 18661-1:2014

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 18661-1:2014

Contents

Page

Introduction v
Background v
IEC 60559 floating-point standard v
C support for IEC 60559 vi
Purpose vii

1 Scope 1

2 Conformance 1

3 Normative references 1

4 Terms and definitions 2

5 C standard conformance 2
5.1 Freestanding implementations 2
5.2 Predefined macros 2
5.3 Standard headers 3

6 Revised floating-point standard 5

7 Types 6
7.1 Terminology 6
7.2 Canonical representation 7

8 Operation binding 8

9 Floating to integer conversion 13

10 Conversions between floating types and character sequences 13
10.1 Conversions with decimal character sequences 13
10.2 Conversions to character sequences 14

11 Constant rounding directions 15

12 NaN support 22

13 Integer width macros 27

14 Mathematics `<math.h>` 29
14.1 Nearest integer functions 29
14.1.1 Round to integer value in floating type 29
14.1.2 Convert to integer type 31
14.2 The `llogb` functions 34
14.3 Max-min magnitude functions 35
14.4 The `nextup` and `nextdown` functions 36
14.5 Functions that round result to narrower type 37
14.6 Comparison macros 40
14.7 Classification macros 41
14.8 Total order functions 43
14.9 Canonicalize functions 44
14.10 NaN functions 45

15 The floating-point environment `<fenv.h>` 47
15.1 The `fesetexcept` function 47
15.2 The `fetestexceptflag` function 48
15.3 Control modes 48

16 Type-generic math `<tgmath.h>` 50

Bibliography 52

Documents
Document Sharing Hub

© ISO/IEC 2014 – All rights reserved

Full document details: PD ISO/IEC TS 18661-1:2014

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*, 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*

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

Part 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*.

Parts 3-5 specify extensions to ISO/IEC 9899:2011 for features recommended in ISO/IEC/IEEE 60559:2011.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

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:

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:

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 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

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

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 data that may be exchanged 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 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

ISO/IEC 9899:2011 (C11) includes refinements to the C99 floating-point specification, though is still based on IEC 60559:1989. C11 upgrades 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.

Part 1, this document, 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 Part 1.

Part 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 Part 2.

Part 3 (Interchange and extended types), Part 4 (Supplementary functions), and Part 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.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

Documents
Document Sharing Hub

Full document details: PD 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

1 Scope

This part of ISO/IEC TS 18661 extends programming language C to support binary floating-point arithmetic conforming to ISO/IEC/IEEE 60559:2011. It covers all requirements of IEC 60559 as they pertain to C floating types that use IEC 60559 binary formats.

This part of ISO/IEC TS 18661 does not cover decimal floating-point arithmetic, nor does it cover most optional features of IEC 60559.

This part of ISO/IEC TS 18661 is primarily an update to IEC 9899:2011 (C11), normative Annex F (IEC 60559 floating-point arithmetic). However, it proposes that the new interfaces that are suitable for general implementations be added in the Library clauses of C11. Also it includes a few auxiliary changes in C11 where the specification is problematic for IEC 60559 support.

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 specified in this part of ISO/IEC TS 18661; and

b) It defines **`__STDC_IEC_60559_BFP__`** to **`201404L`**.

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 9899:2011/Cor.1:2012, *Technical Corrigendum 1*

ISO/IEC/IEEE 60559:2011, *Information technology — Microprocessor Systems — Floating-point arithmetic* (with identical content to IEEE 754-2008, *IEEE Standard for Floating-Point Arithmetic*. The Institute of Electrical and Electronic Engineers, Inc., New York, 2008)

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 18661-1:2014

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 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 following change to C11 expands the conformance requirements for freestanding implementations so that they might conform to this part of ISO/IEC TS 18661.

Change to C11:

Insert after the third sentence of 4#6:

> The strictly conforming programs that shall be accepted by a conforming freestanding implementation that defines **`__STDC_IEC_60559_BFP__`** may also use features in the contents of the standard headers **`<fenv.h>`** and **`<math.h>`** and the numeric conversion functions (7.22.1) of the standard header **`<stdlib.h>`**. All identifiers that are reserved when **`<stdlib.h>`** is included in a hosted implementation are reserved when it is included in a freestanding implementation.

5.2 Predefined macros

The following changes to C11 obsolesce **`__STDC_IEC_559__`**, the current conformance macro for Annex F, in favor of **`__STDC_IEC_60559_BFP__`**, for consistency with other conformance macros and to distinguish its application to binary floating-point arithmetic. The macro **`__STDC_IEC_559__`** is retained as obsolescent, for compatibility with existing programs.

Changes to C11:

In 6.10.8.3#1, before:

> **`__STDC_IEC_559__`** The integer constant **`1`**, intended to indicate conformance to Annex F (IEC 60559 binary floating-point arithmetic).

insert:

> **`__STDC_IEC_60559_BFP__`** The integer constant **`201404L`**, intended to indicate conformance to Annex F (IEC 60559 binary floating-point arithmetic).

In 6.10.8.3#1, append to the **`__STDC_IEC_559__`** item:

> Use of this macro is an obsolescent feature.

The following changes to C11 obsolesce **`__STDC_IEC_559_COMPLEX__`**, the current conformance macro for Annex G, in favor of **`__STDC_IEC_60559_COMPLEX__`**, for consistency with other conformance macros.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

Changes to C11:

In 6.10.8.3#1, after the **__STDC_IEC_559__** item, insert the item:

> **__STDC_IEC_60559_COMPLEX__** The integer constant **201404L**, intended to indicate conformance to the specifications in annex G (IEC 60559 compatible complex arithmetic).

In 6.10.8.3#1, append to the **__STDC_IEC_559_COMPLEX__** item:

> Use of this macro is an obsolescent feature.

5.3 Standard headers

The new identifiers added to C11 library headers by this part of ISO/IEC TS 18661 are defined or declared by their respective headers only if **__STDC_WANT_IEC_60559_BFP_EXT__** is defined as a macro at the point in the source file where the appropriate header is first included. The following changes to C11 list these identifiers in each applicable library subclause.

Changes to C11:

After 5.2.4.2.1#1, insert the paragraph:

> [1a] The following identifiers are defined only if **__STDC_WANT_IEC_60559_BFP_EXT__** is defined as a macro at the point in the source file where **<limits.h>** is first included:

CHAR_WIDTH	**USHRT_WIDTH**	**ULONG_WIDTH**
SCHAR_WIDTH	**INT_WIDTH**	**LLONG_WIDTH**
UCHAR_WIDTH	**UINT_WIDTH**	**ULLONG_WIDTH**
SHRT_WIDTH	**LONG_WIDTH**	

After 5.2.4.2.2#6, insert the paragraph:

> [6a] The following identifier is defined only if **__STDC_WANT_IEC_60559_BFP_EXT__** is defined as a macro at the point in the source file where **<float.h>** is first included:

CR_DECIMAL_DIG

After 7.6#3, insert the paragraph:

> [3a] The following identifiers are defined or declared only if **__STDC_WANT_IEC_60559_BFP_EXT__** is defined as a macro at the point in the source file where **<fenv.h>** is first included:

femode_t	**fetestexceptflag**
FE_DFL_MODE	**fegetmode**
FE_SNANS_ALWAYS_SIGNAL	**fesetmode**
fesetexcept	

After 7.12#1, insert the paragraph:

> [1a] The following identifiers are defined or declared only if **__STDC_WANT_IEC_60559_BFP_EXT__** is defined as a macro at the point in the source file where **<math.h>** is first included:

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

FP_INT_UPWARD	FP_FAST_FSUBL
FP_INT_DOWNWARD	FP_FAST_DSUBL
FP_INT_TOWARDZERO	FP_FAST_FMUL
FP_INT_TONEARESTFROMZERO	FP_FAST_FMULL
FP_INT_TONEAREST	FP_FAST_DMULL
FP_LLOGB0	FP_FAST_FDIV
FP_LLOGBNAN	FP_FAST_FDIVL
SNANF	FP_FAST_DDIVL
SNAN	FP_FAST_FFMA
SNANL	FP_FAST_FFMAL
FP_FAST_FADD	FP_FAST_DFMAL
FP_FAST_FADDL	FP_FAST_FSQRT
FP_FAST_DADDL	FP_FAST_FSQRTL
FP_FAST_FSUB	FP_FAST_DSQRTL

iseqsig	fmaxmagf	ffmal
iscanonical	fmaxmagl	dfmal
issignaling	fminmag	fsqrt
issubnormal	fminmagf	fsqrtl
iszero	fminmagl	dsqrtl
fromfp	nextup	totalorder
fromfpf	nextupf	totalorderf
fromfpl	nextupl	totalorderl
ufromfp	nextdown	totalordermag
ufromfpf	nextdownf	totalordermagf
ufromfpl	nextdownl	totalordermagl
fromfpx	fadd	canonicalize
fromfpxf	faddl	canonicalizef
fromfpxl	daddl	canonicalizel
ufromfpx	fsub	getpayload
ufromfpxf	fsubl	getpayloadf
ufromfpxl	dsubl	getpayloadl
roundeven	fmul	setpayload
roundevenf	fmull	setpayloadf
roundevenl	dmull	setpayloadl
llogb	fdiv	setpayloadsig
llogbf	fdivl	setpayloadsigf
llogbl	ddivl	setpayloadsigl
fmaxmag	ffma	

After 7.20#4, insert the paragraph:

[4a] The following identifiers are defined only if `__STDC_WANT_IEC_60559_BFP_EXT__` is defined as a macro at the point in the source file where `<stdint.h>` is first included:

INT*N*_WIDTH	UINT_FAST*N*_WIDTH	PTRDIFF_WIDTH
UINT*N*_WIDTH	INTPTR_WIDTH	SIG_ATOMIC_WIDTH
INT_LEAST*N*_WIDTH	UINTPTR_WIDTH	SIZE_WIDTH
UINT_LEAST*N*_WIDTH	INTMAX_WIDTH	WCHAR_WIDTH
INT_FAST*N*_WIDTH	UINTMAX_WIDTH	WINT_WIDTH

After 7.22#1, insert the paragraph:

[1a] The following identifiers are declared only if `__STDC_WANT_IEC_60559_BFP_EXT__` is defined as a macro at the point in the source file where `<stdlib.h>` is first included:

strfromd	strfromf	strfroml

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

After 7.25#1, insert the paragraph:

[1a] The following identifiers are defined as type-generic macros only if **`__STDC_WANT_IEC_60559_BFP_EXT__`** is defined as a macro at the point in the source file where **`<tgmath.h>`** is first included:

`roundeven`	**`fromfpx`**	**`fmul`**
`llogb`	**`ufromfpx`**	**`dmul`**
`fmaxmag`	**`totalorder`**	**`fdiv`**
`fminmag`	**`totalordermag`**	**`ddiv`**
`nextup`	**`fadd`**	**`ffma`**
`nextdown`	**`dadd`**	**`dfma`**
`fromfp`	**`fsub`**	**`fsqrt`**
`ufromfp`	**`dsub`**	**`dsqrt`**

6 Revised floating-point standard

C11 Annex F specifies C language support for the floating-point arithmetic of IEC 60559:1989. This document proposes changes to C11 to bring Annex F into alignment with ISO/IEC/IEEE 60559:2011. The changes to C11 below update the introduction to Annex F to acknowledge the revision to IEC 60559.

Changes to C11:

Change F.1 from:

F.1 Introduction

[1] This annex specifies C language support for the IEC 60559 floating-point standard. The *IEC 60559 floating-point standard* is specifically *Binary floating-point arithmetic for microprocessor systems, second edition* (IEC 60559:1989), previously designated IEC 559:1989 and as *IEEE Standard for Binary Floating-Point Arithmetic* (ANSI/IEEE 754−1985). *IEEE Standard for Radix-Independent Floating-Point Arithmetic* (ANSI/IEEE 854−1987) generalizes the binary standard to remove dependencies on radix and word length. *IEC 60559* generally refers to the floating-point standard, as in IEC 60559 operation, IEC 60559 format, etc. An implementation that defines **`__STDC_IEC_559__`** shall conform to the specifications in this annex.356) Where a binding between the C language and IEC60559 is indicated, the IEC 60559-specified behavior is adopted by reference, unless stated otherwise. Since negative and positive infinity are representable in IEC 60559 formats, all real numbers lie within the range of representable values.

to:

F.1 Introduction

[1] This annex specifies C language support for the IEC 60559 floating-point standard. The *IEC 60559 floating-point standard* is specifically *Floating-point arithmetic* (ISO/IEC/IEEE 60559:2011), also designated as *IEEE Standard for Floating-Point Arithmetic* (IEEE 754−2008). The IEC 60559 floating-point standard supersedes the IEC 60559:1989 binary arithmetic standard, also designated as *IEEE Standard for Binary Floating-Point Arithmetic* (IEEE 754−1985). *IEC 60559* generally refers to the floating-point standard, as in IEC 60559 operation, IEC 60559 format, etc.

[2] The IEC 60559 floating-point standard specifies decimal, as well as binary, floating-point arithmetic. It supersedes *IEEE Standard for Radix-Independent Floating-Point Arithmetic* (ANSI/IEEE 854−1987), which generalized the binary arithmetic standard (IEEE 754-1985) to remove dependencies on radix and word length.

[3] An implementation that defines **`__STDC_IEC_60559_BFP__`** to **`201404L`** shall conform to the specifications in this annex.356) Where a binding between the C language and IEC 60559 is indicated, the IEC 60559-specified behavior is adopted by reference, unless stated otherwise.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

In footnote 356), change "`__STDC_IEC_559__`" to "`__STDC_IEC_60559_BFP__`".

Note that the last sentence of F.1 which is removed above is inserted into a more appropriate place by a later change (see 12 below).

7 Types

7.1 Terminology

IEC 60559 now includes a 128-bit binary format as one of its three binary basic formats: *binary32*, *binary64*, and *binary128*. The binary128 format continues to meet the less specific requirements for a binary64-extended format, as in the previous IEC 60559. The changes to C11 below reflect the new terminology in IEC 60559; these changes are not substantive.

Changes to C11:

In F.2#1, change the three bullets from:

— The `float` type matches the IEC 60559 single format.

— The `double` type matches the IEC 60559 double format,

— The `long double` type matches an IEC 60559 extended format,357) else a non-IEC 60559 extended format, else the IEC 60559 `double` format.

to:

— The `float` type matches the IEC 60559 binary32 format.

— The `double` type matches the IEC 60559 binary64 format.

— The `long double` type matches the IEC 60559 binary128 format, else an IEC 60559 binary64-extended format,357) else a non-IEC 60559 extended format, else the IEC 60559 binary64 format.

In F.2#1, change the sentence after the bullet from:

Any non-IEC 60559 extended format used for the `long double` type shall have more precision than IEC 60559 double and at least the range of IEC 60559 double.358)

to:

Any non-IEC 60559 extended format used for the `long double` type shall have more precision than IEC 60559 binary64 and at least the range of IEC 60559 binary64.358)

Change footnote 357) from:

357) ''Extended'' is IEC 60559's double-extended data format. Extended refers to both the common 80-bit and quadruple 128-bit IEC 60559 formats.

to:

357) IEC 60559 binary64-extended formats include the common 80-bit IEC 60559 format.

In F.2, change the recommended practice from:

Recommended practice

[2] The `long double` type should match an IEC 60559 extended format.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

to:

Recommended practice

[2] The `long double` type should match the IEC 60559 binary128 format, else an IEC 60559 binary64-extended format.

Change footnote 361) from:

361) If the minimum-width IEC60559 extended format (64 bits of precision) is supported, `DECIMAL_DIG` shall be at least 21. If IEC 60559 double (53 bits of precision) is the widest IEC 60559 format supported, then `DECIMAL_DIG` shall be at least 17. (By contrast, `LDBL_DIG` and `DBL_DIG` are 18 and 15, respectively, for these formats.)

to:

361) If the minimum-width IEC 60559 binary64-extended format (64 bits of precision) is supported, `DECIMAL_DIG` shall be at least 21. If IEC 60559 binary64 (53 bits of precision) is the widest IEC 60559 format supported, then `DECIMAL_DIG` shall be at least 17. (By contrast, `LDBL_DIG` and `DBL_DIG` are 18 and 15, respectively, for these formats.)

7.2 Canonical representation

IEC 60559 refers to preferred encodings in a format – or, in C terminology, preferred representations of a type – as *canonical*. Some types also contain redundant or ill-specified representations, which are *non-canonical*. All representations of types with IEC 60559 binary interchange formats are canonical; however, types with IEC 60559 extended formats may have non-canonical encodings. (Types with IEC 60559 decimal interchange formats, covered in ISO/IEC TS 18661-2, contain non-canonical redundant representations.)

Changes to C11:

In 5.2.4.2.2#3, change the sentence:

A *NaN* is an encoding signifying Not-a-Number.

to:

A *NaN* is a value signifying Not-a-Number.

In 5.2.4.2.2 footnote 22, change:

… the terms quiet NaN and signaling NaN are intended to apply to encodings with similar behavior.

to:

… the terms quiet NaN and signaling NaN are intended to apply to values with similar behavior.

After 5.2.4.2.2#5, add:

[5a] An implementation may prefer particular representations of values that have multiple representations in a floating type, 6.2.6.1 not withstanding. The preferred representations of a floating type, including unique representations of values in the type, are called *canonical*. A floating type may also contain *non-canonical* representations, for example, redundant representations of some or all of its values, or representations that are extraneous to the floating-point model. Typically, floating-point operations deliver results with canonical representations.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

In 5.2.4.2.2#5a, attach a footnote to the wording:

> An implementation may prefer particular representations of values that have multiple representations in a floating type, 6.2.6.1 not withstanding.

where the footnote is:

> *) The library operations **`iscanonical`** and **`canonicalize`** distinguish canonical (preferred) representations, but this distinction alone does not imply that canonical and non-canonical representations are of different values.

In 5.2.4.2.2#5a, attach a footnote to the wording:

> A floating type may also contain *non-canonical* representations, for example, redundant representations of some or all of its values, or representations that are extraneous to the floating-point model.

where the footnote is:

> *) Some of the values in the IEC 60559 decimal formats have non-canonical representations (as well as a canonical representation).

8 Operation binding

IEC 60559 includes several new required operations. The change to C11 below shows the complete mapping of IEC 60559 operations to C operators, functions, and function-like macros. The new IEC 60559 operations map to C functions and function-like macros; no new C operators are proposed.

Change to C11:

Replace F.3:

> **F.3 Operators and functions**
>
> [1] C operators and functions provide IEC 60559 required and recommended facilities as listed below.
>
> — The **`+`**, **`–`**, **`*`**, and **`/`** operators provide the IEC 60559 add, subtract, multiply, and divide operations.
>
> — The **`sqrt`** functions in **`<math.h>`** provide the IEC 60559 square root operation.
>
> — The **`remainder`** functions in **`<math.h>`** provide the IEC 60559 remainder operation. The **`remquo`** functions in **`<math.h>`** provide the same operation but with additional information.
>
> — The **`rint`** functions in **`<math.h>`** provide the IEC 60559 operation that rounds a floating-point number to an integer value (in the same precision). The **`nearbyint`** functions in **`<math.h>`** provide the nearbyinteger function recommended in the Appendix to ANSI/IEEE 854.
>
> — The conversions for floating types provide the IEC 60559 conversions between floating-point precisions.
>
> — The conversions from integer to floating types provide the IEC 60559 conversions from integer to floating point.
>
> — The conversions from floating to integer types provide IEC 60559-like conversions but always round toward zero.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

— The **lrint** and **llrint** functions in **<math.h>** provide the IEC 60559 conversions, which honor the directed rounding mode, from floating point to the **long int** and **long long int** integer formats. The **lrint** and **llrint** functions can be used to implement IEC 60559 conversions from floating to other integer formats.

— The translation time conversion of floating constants and the **strtod**, **strtof**, **strtold**, **fprintf**, **fscanf**, and related library functions in **<stdlib.h>**, **<stdio.h>**, and **<wchar.h>** provide IEC 60559 binary-decimal conversions. The **strtold** function in **<stdlib.h>** provides the conv function recommended in the Appendix to ANSI/IEEE 854.

— The relational and equality operators provide IEC 60559 comparisons. IEC 60559 identifies a need for additional comparison predicates to facilitate writing code that accounts for NaNs. The comparison macros (**isgreater**, **isgreaterequal**, **isless**, **islessequal**, **islessgreater**, and **isunordered**) in **<math.h>** supplement the language operators to address this need. The **islessgreater** and **isunordered** macros provide respectively a quiet version of the <> predicate and the unordered predicate recommended in the Appendix to IEC 60559.

— The **feclearexcept**, **feraiseexcept**, and **fetestexcept** functions in **<fenv.h>** provide the facility to test and alter the IEC 60559 floating-point exception status flags. The **fegetexceptflag** and **fesetexceptflag** functions in **<fenv.h>** provide the facility to save and restore all five status flags at one time. These functions are used in conjunction with the type **fexcept_t** and the floating-point exception macros (**FE_INEXACT**, **FE_DIVBYZERO**, **FE_UNDERFLOW**, **FE_OVERFLOW**, **FE_INVALID**) also in **<fenv.h>**.

— The **fegetround** and **fesetround** functions in **<fenv.h>** provide the facility to select among the IEC 60559 directed rounding modes represented by the rounding direction macros in **<fenv.h>** (**FE_TONEAREST**, **FE_UPWARD**, **FE_DOWNWARD**, **FE_TOWARDZERO**) and the values **0**, **1**, **2**, and **3** of **FLT_ROUNDS** are the IEC 60559 directed rounding modes.

— The **fegetenv**, **feholdexcept**, **fesetenv**, and **feupdateenv** functions in **<fenv.h>** provide a facility to manage the floating-point environment, comprising the IEC 60559 status flags and control modes.

— The **copysign** functions in **<math.h>** provide the copysign function recommended in the Appendix to IEC 60559.

— The **fabs** functions in **<math.h>** provide the abs function recommended in the Appendix to IEC 60559.

— The unary minus (−) operator provides the unary minus (−) operation recommended in the Appendix to IEC 60559.

— The **scalbn** and **scalbln** functions in **<math.h>** provide the scalb function recommended in the Appendix to IEC 60559.

— The **logb** functions in **<math.h>** provide the logb function recommended in the Appendix to IEC 60559, but following the newer specifications in ANSI/IEEE 854.

— The **nextafter** and **nexttoward** functions in **<math.h>** provide the nextafter function recommended in the Appendix to IEC 60559 (but with a minor change to better handle signed zeros).

— The **isfinite** macro in **<math.h>** provides the finite function recommended in the Appendix to IEC 60559.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014

— The `isnan` macro in `<math.h>` provides the isnan function recommended in the Appendix to IEC 60559.

— The `signbit` macro and the `fpclassify` macro in `<math.h>`, used in conjunction with the number classification macros (`FP_NAN`, `FP_INFINITE`, `FP_NORMAL`, `FP_SUBNORMAL`, `FP_ZERO`), provide the facility of the class function recommended in the Appendix to IEC 60559 (except that the classification macros defined in 7.12.3 do not distinguish signaling from quiet NaNs).

with:

F.3 Operations

[1] C operators, functions, and function-like macros provide the operations required by IEC 60559 as shown in the following table. Specifications for the C facilities are provided in the listed clauses.

Operation binding

IEC 60559 operation	C operation	Clauses - C11
roundToIntegralTiesToEven	`roundeven`	7.12.9.7a, F.10.6.7a
roundToIntegralTiesAway	`round`	7.12.9.6, F.10.6.6
roundToIntegralTowardZero	`trunc`	7.12.9.8, F.10.6.8
roundToIntegralTowardPositive	`ceil`	7.12.9.1, F.10.6.1
roundToIntegralTowardNegative	`floor`	7.12.9.2, F.10.6.2
roundToIntegralExact	`rint`	7.12.9.4, F.10.6.4
nextUp	`nextup`	7.12.11.5, F.10.8.5
nextDown	`nextdown`	7.12.11.6, F.10.8.6
remainder	`remainder`, `remquo`	7.12.10.2, F.10.7.2, 7.12.10.3, F.10.7.3
minNum	`fmin`	7.12.12.3, F.10.9.3
maxNum	`fmax`	7.12.12.2, F.10.9.2
minNumMag	`fminmag`	7.12.12.5, F.10.9.5
maxNumMag	`fmaxmag`	7.12.12.4, F.10.9.4
scaleB	`scalbn`, `scalbln`	7.12.6.13, F.10.3.13
logB	`logb`, `ilogb`, `llogb`	7.12.6.11, F.10.3.11, 7.12.6.5, F.10.3.5 7.12.6.6a, F.10.3.6a
addition	`+`, `fadd`, `faddl`, `daddl`	6.5.6, 7.12.13a.1, F.10.10a
subtraction	`-`, `fsub`, `fsubl`, `dsubl`	6.5.6, 7.12.13a.2, F.10.10a
multiplication	`*`, `fmul`, `fmull`, `dmull`	6.5.5, 7.12.13a.3, F.10.10a
division	`/`, `fdiv`, `fdivl`, `ddivl`	6.5.5, 7.12.13a.4, F.10.10a
squareRoot	`sqrt`, `fsqrt`, `fsqrtl`, `dsqrtl`	7.12.13a.6, F.10.10a
fusedMultiplyAdd	`fma`, `ffma`, `ffmal`, `dfmal`	7.12.13.1, F.10.10.1, 7.12.13a.5, F.10.10a
convertFromInt	cast and implicit conversion	6.3.1.4, 6.5.4
convertToIntegerTiesToEven	`fromfp`, `ufromfp`	7.12.9.9, F.10.6.9
convertToIntegerTowardZero	`fromfp`, `ufromfp`	7.12.9.9, F.10.6.9
convertToIntegerTowardPositive	`fromfp`, `ufromfp`	7.12.9.9, F.10.6.9
convertToIntegerTowardNegative	`fromfp`, `ufromfp`	7.12.9.9, F.10.6.9
convertToIntegerTiesToAway	`fromfp`, `ufromfp`, `lround`, `llround`	7.12.9.9, F.10.6.9, 7.12.9.7, F.10.6.7
convertToIntegerExactTiesToEven	`fromfpx`, `ufromfpx`	7.12.9.10, F.10.6.10
convertToIntegerExactTowardZero	`fromfpx`, `ufromfpx`	7.12.9.10, F.10.6.10

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 18661-1:2014