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)

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

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. 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.

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

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 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.

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)

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.

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:

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

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


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:

```
INTN_WIDTH
UINTN_WIDTH
INT_LEASTN_WIDTH
UINT_LEASTN_WIDTH
INT_FASTN_WIDTH
UINT_FASTN_WIDTH
INTPTR_WIDTH
UINTPTR_WIDTH
INTMAX_WIDTH
UINTMAX_WIDTH
PTRDIFF_WIDTH
SIG_ATOMIC_WIDTH
SIZE_WIDTH
WCHAR_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
```

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.

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.

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.