ISO/IEC 60559

Edition 2.0 2020-05

IEEE Std 754™

INTERNATIONAL STANDARD

Floating-point arithmetic

ISO/IEC 60559:2020-05(en) IEEE Std 754-2019

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
67 000 electrotechnical terminology entries in English and French extracted from the Terms and definitions clause of IEC publications issued between 2002 and 2015. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

ISO/IEC 60559

Edition 2.0 2020-05

IEEE Std 754™

INTERNATIONAL STANDARD

Floating-point arithmetic

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 35.200

ISBN 978-2-8322-8178-9

Contents

FLOATING-POINT ARITHMETIC

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

   IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see* http://standards.ieee.org/ipr/disclaimers.html *for more information).*

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE-SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

International Standard ISO/IEC 60559/IEEE Std 754 has been processed through ISO/IEC subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology, under the IEC/IEEE Dual Logo Agreement.

The text of this standard is based on the following documents:

IEEE Std	FDIS	Report on voting
754 (2019)	JTC1-SC25/2933/FDIS	JTC1-SC25/2936/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IEEE Std 754™-2019
(Revision of IEEE Std 754-2008)

IEEE Standard for Floating-Point Arithmetic

Sponsor

Microprocessor Standards Committee
of the
IEEE Computer Society

Approved 13 June 2019

IEEE-SA Standards Board

Abstract: This standard specifies interchange and arithmetic formats and methods for binary and decimal floating-point arithmetic in computer programming environments. This standard specifies exception conditions and their default handling. An implementation of a floating-point system conforming to this standard may be realized entirely in software, entirely in hardware, or in any combination of software and hardware. For operations specified in the normative part of this standard, numerical results and exceptions are uniquely determined by the values of the input data, sequence of operations, and destination formats, all under user control.

Keywords: arithmetic, binary, computer, decimal, exponent, floating-point, format, IEEE 754™, interchange, NaN, number, rounding, significand, subnormal.

IEEE Introduction

This introduction is not part of IEEE Std 754-2019, IEEE Standard for Floating-Point Arithmetic.

This standard is a product of the Floating-Point Working Group of, and sponsored by, the Microprocessor Standards Committee of the IEEE Computer Society.

This standard provides a discipline for performing floating-point computation that yields results independent of whether the processing is done in hardware, software, or a combination of the two. For operations specified in the normative part of this standard, numerical results and exceptions are uniquely determined by the values of the input data, the operation, and the destination, all under user control.

This standard defines a family of commercially feasible ways for systems to perform binary and decimal floating-point arithmetic. Among the desiderata that guided the formulation of this standard were:

a) Facilitate movement of existing programs from diverse computers to those that adhere to this standard as well as among those that adhere to this standard.

b) Enhance the capabilities and safety available to users and programmers who, although not expert in numerical methods, might well be attempting to produce numerically sophisticated programs.

c) 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. Together with language controls it should be possible to write programs that produce identical results on all conforming systems.

d) Provide direct support for
   - execution-time diagnosis of anomalies
   - smoother handling of exceptions
   - interval arithmetic at a reasonable cost.

e) Provide for development of
   - common elementary functions such as *exp* or *cos*
   - high precision (multiword) arithmetic
   - coupled numerical and symbolic algebraic computation.

f) Enable rather than preclude further refinements and extensions.

In programming environments, this standard is also intended to form the basis for a dialog between the numerical community and programming language designers. It is hoped that language-defined methods for the control of expression evaluation and exceptions might be defined in coming years, so that it will be possible to write programs that produce identical results on all conforming systems. However, it is recognized that utility and safety in languages are sometimes antagonists, as are efficiency and portability.

Therefore, it is hoped that language designers will look on the full set of operation, precision, and exception controls described here as a guide to providing the programmer with the ability to portably control expressions and exceptions. It is also hoped that designers will be guided by this standard to provide extensions in a completely portable way.

Informative annexes provide additional information – Annex A lists bibliographical resources, Annex B suggests programming environment features for debugging support, and Annex C lists all references to the operations of the standard.

Floating-Point Arithmetic

1. Overview

1.1 Scope

This standard specifies formats and operations for floating-point arithmetic in computer systems. Exception conditions are defined and handling of these conditions is specified.

1.2 Purpose

This standard provides a method for computation with floating-point numbers that will yield the same result whether the processing is done in hardware, software, or a combination of the two. The results of the computation will be identical, independent of implementation, given the same input data. Errors, and error conditions, in the mathematical processing will be reported in a consistent manner regardless of implementation.

1.3 Inclusions

This standard specifies:

- Formats for binary and decimal floating-point data, for computation and data interchange.
- Addition, subtraction, multiplication, division, fused multiply add, square root, compare, and other operations.
- Conversions between integer and floating-point formats.
- Conversions between different floating-point formats.
- Conversions between floating-point formats and external representations as character sequences.
- Floating-point exceptions and their handling, including data that are not numbers (NaNs).

1.4 Exclusions

This standard does not specify:

- Formats of integers.
- Interpretation of the sign and significand fields of NaNs.

1.5 Programming environment considerations

This standard specifies floating-point arithmetic in two radices, 2 and 10. A programming environment may conform to this standard in one radix or in both.

This standard does not define all aspects of a conforming programming environment. Such behavior should be defined by a programming language definition supporting this standard, if available, and otherwise by a particular implementation. Some programming language specifications might permit some behaviors to be defined by the implementation.

Language-defined behavior should be defined by a programming language standard supporting this standard. Then all implementations conforming both to this floating-point standard and to that language standard behave identically with respect to such language-defined behaviors. Standards for languages intended to reproduce results exactly on all platforms are expected to specify behavior more tightly than do standards for languages intended to maximize performance on every platform.

Because this standard requires facilities that are not currently available in common programming languages, the standards for such languages might not be able to fully conform to this standard if they are no longer being revised. If the language can be extended by a function library or class or package to provide a conforming environment, then that extension should define all the language-defined behaviors that would normally be defined by a language standard.

Implementation-defined behavior is defined by a specific implementation of a specific programming environment conforming to this standard. Implementations define behaviors not specified by this standard nor by any relevant programming language standard or programming language extension.

Conformance to this standard is a property of a specific implementation of a specific programming environment, rather than of a language specification.

However a language standard could also be said to conform to this standard if it were constructed so that every conforming implementation of that language also conformed automatically to this standard.

1.6 Word usage

In this standard three words are used to differentiate between different levels of requirements and optionality, as follows:

- **may** indicates a course of action permissible within the limits of the standard with no implied preference (“may” means “is permitted to”)
- **shall** indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (“shall” means “is required to”)
- **should** indicates that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (“should” means “is recommended to”).

Further:

- **might** indicates the possibility of a situation that could occur, with no implication of the likelihood of that situation (“might” means “could possibly”)
- **see** followed by a number is a cross-reference to the clause or subclause of this standard identified by that number
- **NOTE** introduces text that is informative (that is, is not a requirement of this standard).

2. Definitions, abbreviations, and acronyms

2.1 Definitions

For the purposes of this standard, the following terms and definitions apply.

applicable attribute: The value of an attribute governing a particular instance of execution of a computational operation of this standard. Languages specify how the applicable attribute is determined.

arithmetic format: A floating-point format that can be used to represent floating-point operands or results for the operations of this standard.

attribute: An implicit parameter to operations of this standard, which a user might statically set in a programming language by specifying a constant value. The term attribute might refer to the parameter (as in "rounding-direction attribute") or its value (as in "roundTowardZero attribute").

basic format: One of five floating-point representations, three binary and two decimal, whose encodings are specified by this standard, and which can be used for arithmetic. One or more of the basic formats is implemented in any conforming implementation.

biased exponent: The sum of the exponent and a constant (bias) chosen to make the biased exponent's range non-negative.

binary floating-point number: A floating-point number with radix two.

block: A language-defined syntactic unit for which a user can specify attributes. Language standards might provide means for users to specify attributes for blocks of varying scopes, even as large as an entire program and as small as a single operation.

canonical encoding: A preferred encoding of a floating-point representation in a format. "Canonical encoding" also applies to declets, significands of finite numbers, infinities, and NaNs, especially in decimal formats.

cohort: The set of all floating-point representations that represent a given floating-point number in a given floating-point format. In this context −0 and +0 are considered distinct and are in different cohorts.

computational operation: An operation that produces floating-point results or that might signal floating-point exceptions. Computational operations produce results in floating-point or other destination formats by rounding them to fit if necessary.

correct rounding: This standard's method of converting an infinitely precise result to a floating-point number, as determined by the applicable rounding direction. A floating-point number so obtained is said to be correctly rounded.

decimal floating-point number: A floating-point number with radix ten.

declet: An encoding of three decimal digits into ten bits using the densely packed decimal encoding scheme. Computational operations accept all 1024 possible declets in operands. Most computational operations produce only the 1000 canonical declets.

denormalized number: *See:* **subnormal number.**

destination: The location for the result of an operation upon one or more operands. A destination might be either explicitly designated by the user or implicitly supplied by the system (for example, intermediate results in subexpressions or arguments for procedures). Even though some languages place the results of intermediate calculations in destinations beyond the user's control, this standard defines the result of an operation in terms of that destination's format and the operands' values.

dynamic mode: An optional method of dynamically setting attributes by means of operations of this standard to set, test, save, and restore them.

exception: An event that occurs when an operation on some particular operands has no outcome suitable for every reasonable application. That operation might signal an exception by invoking default exception handling or alternate exception handling. Exception handling might signal further exceptions. Recognize that *event*, *exception*, and *signal* are defined in diverse ways in different programming environments.

exponent: The component of a finite floating-point representation that signifies the integer power to which the radix is raised in determining the value of that floating-point representation. The exponent e is used when the significand is regarded as an integer digit and fraction field, and the exponent q is used when the significand is regarded as an integer; $e=q+p-1$ where p is the precision of the format in digits.

extendable precision format: A format with precision and range that are defined under user control.

extended precision format: A format that extends a supported basic format by providing wider precision and range.

external character sequence: A representation of a floating-point datum as a sequence of characters, including the character sequences in floating-point literals in program text.

flag: *See:* **status flag.**

floating-point datum: A floating-point number or non-number (NaN) that is representable in a floating-point format. In this standard, a floating-point datum is not always distinguished from its representation or encoding.

floating-point number: A finite or infinite number that is representable in a floating-point format. A floating-point datum that is not a NaN. All floating-point numbers, including zeros and infinities, are signed.

floating-point operation: An operation where an operand or result is a floating-point datum.

floating-point representation: An unencoded member of a floating-point format, representing a finite number, a signed infinity, a quiet NaN, or a signaling NaN. A representation of a finite number has three components: a sign, an exponent, and a significand; its numerical value is the signed product of its significand and its radix raised to the power of its exponent.

format: A set of representations of numerical values and symbols, perhaps accompanied by an encoding.

fusedMultiplyAdd: The operation fusedMultiplyAdd(x, y, z) computes $(x\times y)+z$ as if with unbounded range and precision, rounding only once to the destination format.

generic operation: An operation of this standard that can take operands of various formats, for which the formats of the results might depend on the formats of the operands.

homogeneous operation: An operation of this standard that takes operands and returns results all in the same format.

implementation-defined: Behavior defined by a specific implementation of a specific programming environment conforming to this standard.

integer format: A format not defined in this standard that represents a subset of the integers and perhaps additional values representing infinities, NaNs, or negative zeros.

interchange format: A format that has a specific fixed-width encoding defined in this standard.

language-defined: Behavior defined by a programming language standard supporting this standard.

NaN: not a number—a symbolic floating-point datum. There are two kinds of NaN representations: quiet and signaling. Most operations propagate **quiet NaN**s without signaling exceptions, and signal the invalid operation exception when given a **signaling NaN** operand.

narrower/wider format: If the set of floating-point numbers of one format is a proper subset of another format, the first is called narrower and the second wider. The wider format might have greater precision, range, or (usually) both.

non-computational operation: An operation that is not computational.

normal number: For a particular format, a finite non-zero floating-point number with magnitude greater than or equal to a minimum b^{emin} value, where b is the radix. Normal numbers can use the full precision available in a format. In this standard, zero is neither normal nor subnormal.

not a number: *See:* **NaN.**

operation: this standard defines required and recommended operations which operate on zero or more operands and produce results or side effects, such as changes in dynamic modes or flags or control flow, or both. In this standard, operations are written as named functions; in a specific programming environment