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

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

THIS PUBLICATION IS COPYRIGHT PROTECTED
Copyright © 2019 IEEE

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from ISO, IEC or IEEE at the respective address given below.

ISO copyright office
Case postale 56
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 749 01 11
copyright@iso.org
www.iso.org

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue
New York, NY 10016-5997
United States of America
stds.info@ieee.org
www.ieee.org

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 It also gives information on projects, replaced and withdrawn publications.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.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

ISO/IEC 60559

Edition 2.0 2020-05

IEEE Std 754™

INTERNATIONAL STANDARD

Floating-point arithmetic

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 35.200
IEEE STD24165 (PDF)

IEC ISBN 978-2-8322-8178-9
IEEE ISBN 978-1-5044-6641-7

Warning! Make sure that you obtained this publication from an authorized distributor.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

Contents

1. Overview....11
1.1 Scope....11
1.2 Purpose....11
1.3 Inclusions....11
1.4 Exclusions....11
1.5 Programming environment considerations....12
1.6 Word usage....12

2. Definitions, abbreviations, and acronyms....13
2.1 Definitions....13
2.2 Abbreviations and acronyms....15

3. Floating-point formats....16
3.1 Overview....16
3.2 Specification levels....17
3.3 Sets of floating-point data....17
3.4 Binary interchange format encodings....19
3.5 Decimal interchange format encodings....20
3.6 Interchange format parameters....23
3.7 Extended and extendable precisions....25

4. Attributes and rounding....26
4.1 Attribute specification....26
4.2 Dynamic modes for attributes....26
4.3 Rounding-direction attributes....27

5. Operations....29
5.1 Overview....29
5.2 Decimal exponent calculation....30
5.3 Homogeneous general-computational operations....31
5.4 formatOf general-computational operations....33
5.5 Quiet-computational operations....35
5.6 Signaling-computational operations....37
5.7 Non-computational operations....37
5.8 Details of conversions from floating-point to integer formats....39
5.9 Details of operations to round a floating-point datum to integral value....41
5.10 Details of totalOrder predicate....42
5.11 Details of comparison predicates....43
5.12 Details of conversion between floating-point data and external character sequences....44

6. Infinity, NaNs, and sign bit....48
6.1 Infinity arithmetic....48
6.2 Operations with NaNs....48
6.3 The sign bit....50

7. Exceptions and default exception handling....51
7.1 Overview: exceptions and flags....51
7.2 Invalid operation....52
7.3 Division by zero....53
7.4 Overflow....53
7.5 Underflow....53
7.6 Inexact....54

8. Alternate exception handling attributes....55
8.1 Overview....55
8.2 Resuming alternate exception handling attributes....55
8.3 Immediate and delayed alternate exception handling attributes....56

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

9. Recommended operations....58
9.1 Conforming language- and implementation-defined operations....58
9.2 Additional mathematical operations....58
9.3 Dynamic mode operations....65
9.4 Reduction operations....66
9.5 Augmented arithmetic operations....68
9.6 Minimum and maximum operations....69
9.7 NaN payload operations....71

10. Expression evaluation....72
10.1 Expression evaluation rules....72
10.2 Assignments, parameters, and function values....72
10.3 preferredWidth attributes for expression evaluation....73
10.4 Literal meaning and value-changing optimizations....74

11. Reproducible floating-point results....75

Annex A (informative) Bibliography....77

Annex B (informative) Program debugging support....79

Annex C (informative) List of operations....81

Annex D (informative) IEEE list of participants....83

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

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).

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

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.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

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

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

they might be represented by operators, or by families of format-specific functions, or by operations or functions whose names might differ from those in this standard.

payload: The information, which might be diagnostic, contained in a NaN.

precision: The maximum number p of significant digits that can be represented in a format, or the number of digits to which a result is rounded.

preferred exponent: For the result of a decimal operation, the value of the exponent q which best reflects the quanta of the operands when the result is exact.

preferredWidth method: A method used by a programming language to determine the destination formats for generic operations and functions. Some **preferredWidth** methods take advantage of the extra range and precision of wide formats without requiring the program to be written with explicit conversions.

quantum: The quantum of a finite floating-point representation is the value of a unit in the last position of its significand. This is equal to the radix raised to the exponent q, which is used when the significand is regarded as an integer.

quiet operation: An operation that never signals any floating-point exception.

radix: The base for the representation of binary or decimal floating-point numbers, two or ten.

result: The floating-point representation or encoding that is delivered to the destination.

signal: When an operation on some particular operands has no outcome suitable for every reasonable application, that operation might signal one or more exceptions by invoking the default handling or, if explicitly requested, a language-defined alternate handling selected by the user.

significand: A component of a finite floating-point number containing its significant digits. The significand can be thought of as an integer, a fraction, or some other fixed-point form, by choosing an appropriate exponent offset. A decimal or subnormal binary significand can also contain leading zeros, which are not significant.

status flag: A variable that can take two states, raised or lowered. When raised, a status flag might convey additional system-dependent information, possibly inaccessible to some users. The operations of this standard, when exceptional, can as a side effect raise some of the following status flags: inexact, underflow, overflow, divideByZero, and invalid operation.

subnormal number: In a particular format, a non-zero floating-point number with magnitude less than the magnitude of that format's smallest normal number. A subnormal number does not use the full precision available to normal numbers of the same format.

supported format: A floating-point format provided in the programming environment and implemented in conformance with the requirements of this standard. Thus, a programming environment might provide more formats than it supports, as only those implemented in accordance with the standard are said to be supported. Also, an integer format is said to be supported if conversions between that format and supported floating-point formats are provided in conformance with this standard.

trailing significand field: A component of an encoded binary or decimal floating-point format containing all the significand digits except the leading digit. In these formats, the biased exponent or combination field encodes or implies the leading significand digit.

user: Any person, hardware, or program not itself specified by this standard, having access to and controlling those operations of the programming environment specified in this standard.

width of an operation: The format of the destination of an operation specified by this standard; it will be one of the supported formats provided by an implementation in conformance to this standard.

2.2 Abbreviations and acronyms

LSB	least significant bit
MSB	most significant bit
NaN	not a number
qNaN	quiet NaN
sNaN	signaling NaN

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

3. Floating-point formats

3.1 Overview

3.1.1 Formats

This clause defines floating-point formats, which are used to represent a finite subset of real numbers (see 3.2). Formats are characterized by their radix, precision, and exponent range, and each format can represent a unique set of floating-point data (see 3.3).

All formats can be supported as **arithmetic formats**; that is, they may be used to represent floating-point operands or results for the operations described in later clauses of this standard.

Specific fixed-width encodings for binary and decimal formats are defined in this clause for a subset of the formats (see 3.4 and 3.5). These **interchange formats** are identified by their size (see 3.6) and can be used for the exchange of floating-point data between implementations.

Five **basic formats** are defined in this clause:

— Three binary formats, with encodings in lengths of 32, 64, and 128 bits.
— Two decimal formats, with encodings in lengths of 64 and 128 bits.

Additional arithmetic formats are recommended for extending these basic formats (see 3.7).

The choice of which of this standard's formats to support is language-defined or, if the relevant language standard is silent or defers to the implementation, implementation-defined. The names used for formats in this standard are not necessarily those used in programming environments.

3.1.2 Conformance

A conforming implementation of any supported format shall provide means to initialize that format and shall provide conversions between that format and all other supported formats.

A conforming implementation of a supported arithmetic format shall provide all the operations of this standard defined in Clause 5, for that format.

A conforming implementation of a supported interchange format shall provide means to read and write that format using a specific encoding defined in this clause, for that format.

A programming environment conforms to this standard, in a particular radix, by implementing one or more of the basic formats of that radix as both a supported arithmetic format and a supported interchange format.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

3.2 Specification levels

Floating-point arithmetic is a systematic approximation of real arithmetic, as illustrated in Table 3.1. Floating-point arithmetic can only represent a finite subset of the continuum of real numbers. Consequently certain properties of real arithmetic, such as associativity of addition, do not always hold for floating-point arithmetic.

Table 3.1—Relationships between different specification levels for a particular format

Level 1	**{−∞ … 0 … +∞}**	Extended real numbers.
many-to-one ↓	*rounding*	↑ projection (except for NaN)
Level 2	**{−∞ … −0}** ∪ **{+0 … +∞}** ∪ **NaN**	Floating-point data—an algebraically closed system.
one-to-many ↓	*representation specification*	↑ many-to-one
Level 3	**(*sign*, *exponent*, *significand*)** ∪ **{−∞, +∞}** ∪ **qNaN** ∪ **sNaN**	Representations of floating-point data.
one-to-many ↓	*encoding for representations of floating-point data*	↑ many-to-one
Level 4	**0111000…**	Bit strings.

The mathematical structure underpinning the arithmetic in this standard is the extended reals, that is, the set of real numbers together with positive and negative infinity. For a given format, the process of *rounding* (see Clause 4) maps an extended real number to a *floating-point number* included in that format. A *floating-point datum*, which can be a signed zero, finite non-zero number, signed infinity, or a NaN (not-a-number), can be mapped to one or more *representations of floating-point data* in a format.

The representations of floating-point data in a format consist of:

— triples (*sign*, *exponent*, *significand*); in radix *b*, the floating-point number represented by a triple is $(-1)^{sign} \times b^{exponent} \times significand$

— +∞, −∞

— qNaN (quiet), sNaN (signaling).

An *encoding* maps a representation of a floating-point datum to a bit string. An encoding might map some representations of floating-point data to more than one bit string. A NaN encoding should be used to store retrospective diagnostic information (see 6.2).

3.3 Sets of floating-point data

This subclause specifies the sets of floating-point data representable within all floating-point formats; the encodings for specific representations of floating-point data in interchange formats are defined in 3.4 and 3.5, and the parameters for interchange formats are defined in 3.6.

The set of finite floating-point numbers representable within a particular format is determined by the following integer parameters:

— *b* = the radix, 2 or 10

— *p* = the number of digits in the significand (precision)

— *emax* = the maximum exponent *e*

— *emin* = the minimum exponent *e*

emin shall be 1 − *emax* for all formats.

The values of these parameters for each basic format are given in Table 3.2, in which each format is identified by its radix and the number of bits in its encoding. Constraints on these parameters for extended and extendable precision formats are given in 3.7.

Within each format, the following floating-point data shall be represented:

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

— Signed zero and non-zero floating-point numbers of the form $(-1)^s \times b^e \times m$, where
  — s is 0 or 1.
  — e is any integer $emin \le e \le emax$.
  — m is a number represented by a digit string of the form
    $d_0 \bullet d_1 d_2 \ldots d_{p-1}$ where d_i is an integer digit $0 \le d_i < b$ (therefore $0 \le m < b$).
— Two infinities, $+\infty$ and $-\infty$.
— Two NaNs, qNaN (quiet) and sNaN (signaling).

These are the only floating-point data represented.

In the foregoing description, the significand m is viewed in a scientific form, with the radix point immediately following the first digit. It is also convenient for some purposes to view the significand as an integer; in which case the finite floating-point numbers are described thus:

— Signed zero and non-zero floating-point numbers of the form $(-1)^s \times b^q \times c$, where
  — s is 0 or 1.
  — q is any integer $emin \le q+p-1 \le emax$.
  — c is a number represented by a digit string of the form
    $d_0 d_1 d_2 \ldots d_{p-1}$ where d_i is an integer digit $0 \le d_i < b$ (c is therefore an integer with $0 \le c < b^p$).

This view of the significand as an integer c, with its corresponding exponent q, describes exactly the same set of zero and non-zero floating-point numbers as the view in scientific form. (For finite floating-point numbers, $e=q+p-1$ and $m=c \times b^{1-p}$.)

The smallest positive *normal* floating-point number is b^{emin} and the largest is $b^{emax} \times (b - b^{1-p})$. The non-zero floating-point numbers for a format with magnitude less than b^{emin} are called *subnormal* because their magnitudes lie between zero and the smallest normal magnitude. They always have fewer than p significant digits. Every finite floating-point number is an integral multiple of the smallest subnormal magnitude $b^{emin} \times b^{1-p}$.

For a floating-point number that has the value zero, the sign s provides extra information. Although all formats have distinct representations for +0 and −0, the sign of a zero is significant in some circumstances, such as division by zero, but not in others (see 6.3). Binary interchange formats have just one representation each for +0 and −0, but decimal formats have many. In this standard, 0 and ∞ are written without a sign when the sign is not important.

Table 3.2—Parameters defining basic format floating-point numbers

	Binary format (b=2)			**Decimal format** (b=10)	
parameter	**binary32**	**binary64**	**binary128**	**decimal64**	**decimal128**
p, digits	24	53	113	16	34
$emax$	+127	+1023	+16383	+384	+6144

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

3.4 Binary interchange format encodings

Each floating-point number has just one encoding in a binary interchange format. To make the encoding unique, in terms of the parameters in 3.3, the value of the significand m is maximized by decreasing e until either $e=emin$ or $m \geq 1$. After this process is done, if $e=emin$ and $0<m<1$, the floating-point number is subnormal. Subnormal numbers (and zero) are encoded with a reserved biased exponent value.

Representations of floating-point data in the binary interchange formats are uniquely encoded in k bits in the following three fields ordered as shown in Figure 3.1:

a) 1-bit sign S
b) w-bit biased exponent $E = e + bias$
c) $(t=p-1)$-bit trailing significand field digit string $T=d_1 d_2 \ldots d_{p-1}$; the leading bit of the significand, d_0, is implicitly encoded in the biased exponent E.

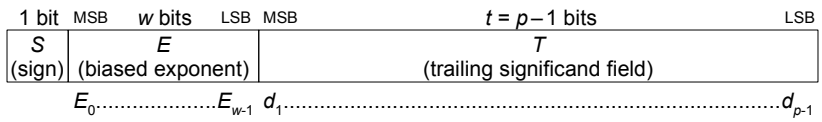

Figure 3.1—Binary interchange floating-point format

The values of k, p, t, w, and $bias$ for binary interchange formats are listed in Table 3.5 (see 3.6).

The range of the encoding's biased exponent E shall include:

— every integer between 1 and 2^w-2, inclusive, to encode normal numbers
— the reserved value 0 to encode ±0 and subnormal numbers
— the reserved value 2^w-1 to encode ±∞ and NaNs.

The representation r of the floating-point datum, and value v of the floating-point datum represented, are inferred from the constituent fields as follows:

a) If $E = 2^w-1$ and $T \neq 0$, then r is qNaN or sNaN and v is NaN regardless of S and then d_1 shall exclusively distinguish between qNaN and sNaN (see 6.2.1).
b) If $E = 2^w-1$ and $T=0$, then r and $v = (-1)^S \times (+\infty)$.
c) If $1 \leq E \leq 2^w-2$, then r is $(S, (E-bias), (1+2^{1-p} \times T))$;
the value of the corresponding floating-point number is $v = (-1)^S \times 2^{E-bias} \times (1+2^{1-p} \times T)$;
thus normal numbers have an implicit leading significand bit of 1.
d) If $E=0$ and $T \neq 0$, then r is $(S, emin, (0+2^{1-p} \times T))$;
the value of the corresponding floating-point number is $v = (-1)^S \times 2^{emin} \times (0+2^{1-p} \times T)$;
thus subnormal numbers have an implicit leading significand bit of 0.
e) If $E=0$ and $T=0$, then r is $(S, emin, 0)$ and $v = (-1)^S \times (+0)$ (signed zero, see 6.3).

In binary interchange formats, all number and NaN encodings are canonical.

NOTE—Where k is either 64 or a multiple of 32 and ≥ 128, for these encodings all of the following are true (where round( ) rounds to the nearest integer):

$$
\begin{aligned}
k &= 1 + w + t = w + p = 32 \times \text{ceiling}((p + \text{round}(4 \times \log_2(p + \text{round}(4 \times \log_2(p)) - 13)) - 13)/32) \\
w &= k - t - 1 = k - p = \text{round}(4 \times \log_2(k)) - 13 \\
t &= k - w - 1 = p - 1 = k - \text{round}(4 \times \log_2(k)) + 12 \\
p &= k - w = t + 1 = k - \text{round}(4 \times \log_2(k)) + 13 \\
emax &= bias = 2^{(w-1)} - 1 \\
emin &= 1 - emax = 2 - 2^{(w-1)}.
\end{aligned}
$$

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

3.5 Decimal interchange format encodings

3.5.1 Cohorts

Unlike in a binary floating-point format, in a decimal floating-point format a number might have multiple representations. The set of representations a floating-point number maps to is called the floating-point number's *cohort*; the members of a cohort are distinct *representations* of the same floating-point number. For example, if c is a multiple of 10 and q is less than its maximum allowed value, then (s, q, c) and (s, $q+1$, $c/10$) are two representations for the same floating-point number and are members of the same cohort.

While numerically equal, different members of a cohort can be distinguished by the decimal-specific operations (see 5.3.2, 5.5.2, and 5.7.3). The cohorts of different floating-point numbers might have different numbers of members. If a finite non-zero number's representation has n decimal digits from its most significant non-zero digit to its least significant non-zero digit, the representation's cohort will have at most $p-n+1$ members where p is the number of digits of precision in the format.

For example, a one-digit floating-point number might have up to p different representations while a p-digit floating-point number with no trailing zeros has only one representation. (An n-digit floating-point number might have fewer than $p-n+1$ members in its cohort if it is near the extremes of the format's exponent range.) A zero has a much larger cohort: the cohort of +0 contains a representation for each exponent, as does the cohort of −0.

For decimal arithmetic, besides specifying a numerical result, the arithmetic operations also select a member of the result's cohort according to 5.2. Decimal applications can make use of the additional information cohorts convey.

3.5.2 Encodings

Representations of floating-point data in the decimal interchange formats are encoded in k bits in the following three fields, whose detailed layouts and canonical (preferred) encodings are described below.

a) 1-bit sign S.

b) A $w+5$ bit combination field G encoding classification and, if the encoded datum is a finite number, the exponent q and four significand bits (1 or 3 of which are implied). The biased exponent E is a $w+2$ bit quantity $q+bias$, where the value of the first two bits of the biased exponent taken together is either 0, 1, or 2.

c) A t-bit trailing significand field T that contains $J \times 10$ bits and contains the bulk of the significand. When this field is combined with the leading significand bits from the combination field, the format encodes a total of $p = 3 \times J + 1$ decimal digits.

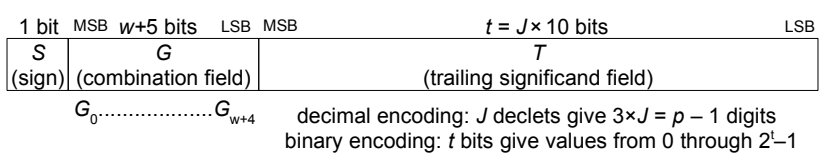

Figure 3.2—Decimal interchange floating-point formats

The values of k, p, t, w, and $bias$ for decimal interchange formats are listed in Table 3.6 (see 3.6).

The representation r of the floating-point datum, and value v of the floating-point datum represented, are inferred from the constituent fields as follows:

a) If G_0 through G_4 are 11111, then v is NaN regardless of S. Furthermore, if G_5 is 1, then r is sNaN; otherwise r is qNaN. The remaining bits of G are ignored, and T is the payload, which can be used to distinguish various NaNs.

   The payload is encoded similarly to finite numbers described below, with G treated as though all bits were zero. The payload corresponds to the significand of finite numbers, interpreted as an

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

integer with a maximum value of $10^{(3\times J)}-1$, and the exponent field is ignored (it is treated as if it were zero). A NaN is in its preferred (canonical) encoding if the bits G_6 through G_{w+4} are zero and the encoding of the payload is canonical.

b) If G_0 through G_4 are 11110 then r and $v = (-1)^S \times (+\infty)$. The values of the remaining bits in G, and T, are ignored. The two canonical encodings of infinity have bits G_5 through $G_{w+4} = 0$, and $T = 0$.

c) For finite numbers, r is (S, $E-bias$, C) and $v = (-1)^S \times 10^{(E-bias)} \times C$, where C is the concatenation of the leading significand digit or bits from the combination field G and the trailing significand field T, and where the biased exponent E is encoded in the combination field. The encoding within these fields depends on whether the implementation uses the decimal or the binary encoding for the significand.

1) If the implementation uses the *decimal* encoding for the significand, then the least significant w bits of the exponent are G_5 through G_{w+4}. The most significant two bits of the biased exponent and the decimal digit string $d_0\ d_1 \ldots d_{p-1}$ of the significand are formed from bits G_0 through G_4 and T as follows:

i) When the most significant five bits of G are 110xx or 1110x, the leading significand digit d_0 is $8+G_4$, a value 8 or 9, and the leading biased exponent bits are $2G_2+G_3$, a value 0, 1, or 2.

ii) When the most significant five bits of G are 0xxxx or 10xxx, the leading significand digit d_0 is $4G_2+2G_3+G_4$, a value in the range 0 through 7, and the leading biased exponent bits are $2G_0+G_1$, a value 0, 1, or 2. Consequently if T is 0 and the most significant five bits of G are 00000, 01000, or 10000, then $v = (-1)^S \times (+0)$.

The $p-1 = 3\times J$ decimal digits $d_1 \ldots d_{p-1}$ are encoded by T, which contains J declets encoded in densely packed decimal.

A canonical significand has only canonical declets, as shown in Tables 3.3 and 3.4. Computational operations accept all 1024 possible declets in operands. Except for quiet-computational operations (see 5.5), computational operations produce only the 1000 canonical declets.

2) Alternatively, if the implementation uses the *binary* encoding for the significand, then:

i) If G_0 and G_1 together are one of 00, 01, or 10, then the biased exponent E is formed from G_0 through G_{w+1} and the significand is formed from bits G_{w+2} through the end of the encoding (including T).

ii) If G_0 and G_1 together are 11 and G_2 and G_3 together are one of 00, 01, or 10, then the biased exponent E is formed from G_2 through G_{w+3} and the significand is formed by prefixing the 4 bits $(8+G_{w+4})$ to T.

The maximum value of the binary-encoded significand is the same as that of the corresponding decimal-encoded significand; that is, $10^{(3\times J+1)}-1$ (or $10^{(3\times J)}-1$ when T is used as the payload of a NaN). If the value exceeds the maximum, the significand c is non-canonical and the value used for c is zero.

Computational operations generally produce only canonical significands, and always accept non-canonical significands in operands.

NOTE—Where k is a positive multiple of 32, for these encodings all of the following are true:

$$\begin{aligned}
k &= 1 + 5 + w + t &&= 32\times\text{ceiling}((p+2)/9)\\
w &= k - t - 6 &&= k/16 + 4\\
t &= k - w - 6 &&= 15\times k/16 - 10\\
p &= 3\times t/10 + 1 &&= 9\times k/32 - 2\\
emax &= 3\times 2^{(w-1)}\\
emin &= 1 - emax\\
bias &= emax + p - 2.
\end{aligned}$$

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

Decoding densely packed decimal: Table 3.3 decodes a declet, with 10 bits $\mathbf{b_{(0)}}$ to $\mathbf{b_{(9)}}$, into 3 decimal digits $\mathbf{d_{(1)}}$, $\mathbf{d_{(2)}}$, $\mathbf{d_{(3)}}$. The first column is in binary and an "x" denotes a "don't care" bit. Thus all 1024 possible 10-bit patterns shall be accepted and mapped into 1000 possible 3-digit combinations with some redundancy.

Table 3.3—Decoding 10-bit densely packed decimal to 3 decimal digits

$\mathbf{b_{(6)}, b_{(7)}, b_{(8)}, b_{(3)}, b_{(4)}}$	$\mathbf{d_{(1)}}$	$\mathbf{d_{(2)}}$	$\mathbf{d_{(3)}}$
0 x x x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$4b_{(7)} + 2b_{(8)} + b_{(9)}$
1 0 0 x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$8 + b_{(9)}$
1 0 1 x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$8 + b_{(5)}$	$4b_{(3)} + 2b_{(4)} + b_{(9)}$
1 1 0 x x	$8 + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$4b_{(0)} + 2b_{(1)} + b_{(9)}$
1 1 1 0 0	$8 + b_{(2)}$	$8 + b_{(5)}$	$4b_{(0)} + 2b_{(1)} + b_{(9)}$
1 1 1 0 1	$8 + b_{(2)}$	$4b_{(0)} + 2b_{(1)} + b_{(5)}$	$8 + b_{(9)}$
1 1 1 1 0	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$8 + b_{(5)}$	$8 + b_{(9)}$
1 1 1 1 1	$8 + b_{(2)}$	$8 + b_{(5)}$	$8 + b_{(9)}$

Encoding densely packed decimal: Table 3.4 encodes 3 decimal digits $\mathbf{d_{(1)}}$, $\mathbf{d_{(2)}}$, and $\mathbf{d_{(3)}}$, each having 4 bits which can be expressed by a second subscript $\mathbf{d_{(1,0:3)}}$, $\mathbf{d_{(2,0:3)}}$, and $\mathbf{d_{(3,0:3)}}$, where bit 0 is the most significant and bit 3 the least significant, into a declet, with 10 bits $\mathbf{b_{(0)}}$ to $\mathbf{b_{(9)}}$. Most computational operations generate only the 1000 canonical 10-bit patterns defined by Table 3.4.

Table 3.4—Encoding 3 decimal digits to 10-bit densely packed decimal

$\mathbf{d_{(1,0)}, d_{(2,0)}, d_{(3,0)}}$	$\mathbf{b_{(0)}, b_{(1)}, b_{(2)}}$	$\mathbf{b_{(3)}, b_{(4)}, b_{(5)}}$	$\mathbf{b_{(6)}}$	$\mathbf{b_{(7)}, b_{(8)}, b_{(9)}}$
0 0 0	$d_{(1,1:3)}$	$d_{(2,1:3)}$	0	$d_{(3,1:3)}$
0 0 1	$d_{(1,1:3)}$	$d_{(2,1:3)}$	1	$0, 0, d_{(3,3)}$
0 1 0	$d_{(1,1:3)}$	$d_{(3,1:2)}, d_{(2,3)}$	1	$0, 1, d_{(3,3)}$
0 1 1	$d_{(1,1:3)}$	$1, 0, d_{(2,3)}$	1	$1, 1, d_{(3,3)}$
1 0 0	$d_{(3,1:2)}, d_{(1,3)}$	$d_{(2,1:3)}$	1	$1, 0, d_{(3,3)}$
1 0 1	$d_{(2,1:2)}, d_{(1,3)}$	$0, 1, d_{(2,3)}$	1	$1, 1, d_{(3,3)}$
1 1 0	$d_{(3,1:2)}, d_{(1,3)}$	$0, 0, d_{(2,3)}$	1	$1, 1, d_{(3,3)}$
1 1 1	$0, 0, d_{(1,3)}$	$1, 1, d_{(2,3)}$	1	$1, 1, d_{(3,3)}$

The 24 non-canonical patterns of the form 01x11x111x, 10x11x111x, or 11x11x111x (where an "x" denotes a "don't care" bit) are not generated in the result of a computational operation. However, as listed in Table 3.3, these 24 bit patterns do map to values in the range 0 through 999. The bit pattern in a NaN trailing significand field can affect how the NaN is propagated (see 6.2).

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020

3.6 Interchange format parameters

Interchange formats support the exchange of floating-point data between implementations. In each radix, the precision and range of an interchange format is defined by its size; interchange of a floating-point datum of a given size is therefore always exact with no possibility of overflow or underflow.

This standard defines binary interchange formats of widths 16, 32, 64, and 128 bits, and in general for any multiple of 32 bits of at least 128 bits. Decimal interchange formats are defined for any multiple of 32 bits of at least 32 bits.

The parameters *p* and *emax* for every interchange format width are shown in Table 3.5 for binary interchange formats and in Table 3.6 for decimal interchange formats. The encodings for the interchange formats are as described in 3.4 and 3.5.2; the encoding parameters for each interchange format width are also shown in Tables 3.5 and 3.6.

Table 3.5—Binary interchange format parameters

Parameter	binary16	binary32	binary64	binary128	binary{*k*} ($k \geq 128$)
k, storage width in bits	16	32	64	128	multiple of 32
p, precision in bits	11	24	53	113	$k - \text{round}(4 \times \log_2(k)) + 13$
emax, maximum exponent *e*	15	127	1023	16383	$2^{(k-p-1)} - 1$
Encoding parameters					
bias, $E - e$	15	127	1023	16383	*emax*
sign bit	1	1	1	1	1
w, exponent field width in bits	5	8	11	15	$\text{round}(4 \times \log_2(k)) - 13$
t, trailing significand field width in bits	10	23	52	112	$k - w - 1$
k, storage width in bits	16	32	64	128	$1 + w + t$

In Table 3.5, round( ) rounds to the nearest integer.

For example, binary256 would have $p = 237$ and $emax = 262143$.

Table 3.6—Decimal interchange format parameters

Parameter	decimal32	decimal64	decimal128	decimal{*k*} ($k \geq 32$)
k, storage width in bits	32	64	128	multiple of 32
p, precision in digits	7	16	34	$9 \times k/32 - 2$
emax	96	384	6144	$3 \times 2^{(k/16+3)}$
Encoding parameters				
bias, $E - q$	101	398	6176	$emax + p - 2$
sign bit	1	1	1	1
w+5, combination field width in bits	11	13	17	$k/16 + 9$
t, trailing significand field width in bits	20	50	110	$15 \times k/16 - 10$
k, storage width in bits	32	64	128	$1 + 5 + w + t$

For example, decimal256 would have $p = 70$ and $emax = 1572864$.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

NOTE—This standard defines the representation of individual data as conceptual Level 4 entities. Applications exchanging data between different implementations must communicate certain parameters that describe the formats and layout of the data. Besides issues such as byte order which affect all data interchange, certain implementation options allowed by this standard must also be considered:

— for binary formats, how signaling NaNs are distinguished from quiet NaNs

— for decimal formats, whether binary or decimal encoding is used.

This standard does not define how these parameters are to be communicated.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

3.7 Extended and extendable precisions

Extended and extendable precision formats are recommended for extending the precisions used for arithmetic beyond the basic formats. Specifically:

— An **extended precision format** is an arithmetic format that extends a supported basic format with both wider precision and wider range.

— An **extendable precision format** is an arithmetic format with a precision and range that are defined under user control.

These formats are characterized by the parameters *b, p,* and *emax,* which may match those of an interchange format and shall:

— provide all the representations of floating-point data defined in terms of those parameters in 3.2 and 3.3

— provide all the operations of this standard, as defined in Clause 5, for that format.

Encodings in these formats should be fixed width and may match those of an interchange format. Each representation of a floating-point number has a unique canonical encoding and may have non-canonical encodings. Each NaN in these formats has a payload, which might encode diagnostic information. Each NaN payload has a canonical encoding and may have non-canonical encodings. All other aspects of encodings for these formats are implementation defined.

Language standards should define mechanisms supporting extendable precision for each supported radix. Language standards supporting extendable precision shall permit users to specify p and *emax*. Language standards shall also allow the specification of an extendable precision by specifying p alone; in this case *emax* shall be defined by the language standard to be at least $1000 \times p$ when p is ≥ 237 bits in a binary format or p is ≥ 51 digits in a decimal format.

Language standards or implementations should support an extended precision format that extends the widest basic format that is supported in that radix. Table 3.7 specifies the minimum precision and exponent range of the extended precision format for each basic format.

Table 3.7—Extended format parameters for floating-point numbers

	Extended formats associated with:				
Parameter	**binary32**	**binary64**	**binary128**	**decimal64**	**decimal 128**
p digits $\geq$	32	64	128	22	40
emax $\geq$	1023	16383	65535	6144	24576

NOTE 1—For extended formats, the minimum exponent range is that of the next wider basic format, if there is one, while the minimum precision is intermediate between a given basic format and the next wider basic format.

NOTE 2—For interchange of binary floating-point data, the width k in bits of the smallest format that will allow the encoding of a significand of at least p bits is given by:

$k = 32 \times \text{ceiling}((p + \text{round}(4 \times \log_2(p + \text{round}(4 \times \log_2(p)) - 13)) - 13)/32)$, where round( ) rounds to the nearest integer and $p \geq 113$; for smaller values of p, see Table 3.5.

For interchange of decimal floating-point data, the width k in bits of the smallest format that will allow the encoding of a significand of at least p digits is given by:

$k = 32 \times \text{ceiling}((p + 2)/9)$, where $p \geq 1$.

In both cases the chosen format might have a larger precision (see 3.4 and 3.5.2).

NOTE 3—For binary formats, the precision p should be at least 3, as some numerical properties do not hold for lower precisions.

Similarly, *emax* should be at least 2 to support the operations listed in 9.2.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.

Full document details: ISO/IEC/IEEE 60559-2020

4. Attributes and rounding

4.1 Attribute specification

An attribute is logically associated with a program block to modify its numerical and exception semantics. A user can specify a constant value for an attribute parameter.

Some attributes have the effect of an implicit parameter to most individual operations of this standard; language standards shall specify

— rounding-direction attributes (see 4.3)

and should specify

— alternate exception handling attributes (see Clause 8).

Other attributes change the mapping of language expressions into operations of this standard; language standards that permit more than one such mapping should provide support for:

— preferredWidth attributes (see 10.3)
— value-changing optimization attributes (see 10.4)
— reproducibility attributes (see Clause 11).

For attribute specification, the implementation shall provide language-defined means, such as compiler directives, to specify a constant value for the attribute parameter for the operations in a block; the scope of the attribute value is the block with which it is associated. Language standards shall provide for constant specification of the default and each specific value of the attribute.

4.2 Dynamic modes for attributes

Attributes in this standard shall be supported with the constant specification of 4.1. Particularly to support debugging, language standards should also support dynamic-mode specification of attributes.

With dynamic-mode specification, a user can specify that the attribute parameter assumes the value of a dynamic-mode variable whose value might not be known until program execution. This standard does not specify the underlying implementation mechanisms for constant attributes or dynamic modes.

For dynamic-mode specification, the implementation shall provide language-defined means to specify that the attribute parameter assumes the value of a dynamic-mode variable for the operations within the scope of the dynamic-mode specification in a block. The implementation initializes a dynamic-mode variable to the default value for the dynamic mode. Within its language-defined (dynamic) scope, changes to the value of a dynamic-mode variable are under the control of the user via the operations in 9.3.1 and 9.3.2.

The following aspects of dynamic-mode variables are language-defined; language standards may explicitly defer the definitions to implementations:

— The precedence of static attribute specifications and dynamic-mode assignments.
— The effect of changing the value of the dynamic-mode variable in an asynchronous event, such as in another thread or signal handler.
— Whether the value of the dynamic-mode variable can be determined by non-programmatic means, such as a debugger.

NOTE—A constant value for an attribute can be specified and meet the requirements of 4.1 by a dynamic mode specification with appropriate scope of that constant value.

Published by IEC under license from IEEE. © 2019 IEEE. All rights reserved.
Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 25,2026 at 19:49:16 UTC from IEEE Xplore. Restrictions apply.
Full document details: ISO/IEC/IEEE 60559-2020