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.

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 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. Multiple NaN bit strings 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:

- 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 bit s provides an extra bit of 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

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 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.

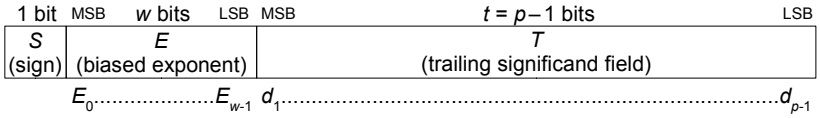

Figure 3.1—Binary interchange floating-point format

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 (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).

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):

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