Proposal for C2Y
WG14 N3535

Title:	**`frexp`** and double-double (updates N3357)
Author, affiliation:	C FP group
Date:	2025-03-15
Proposal category:	Editorial
Reference:	N3357, N3467

N3357, and variations offered during the recent WG 14 meeting, attempted to address the issue described below by qualifying "floating-point number" with "model" or "C model". WG 14 rejected these proposals, objecting to "model" as too vague and to "C model" as inappropriate in C standard specification. This update to N3357 addresses the issue with a clarifying footnote, leaving the normative specification unchanged.

This issue was pointed out by Hubert Tong in a thread titled "frexp and double-double underflow", beginning with [SC22WG14.25341].

The **`frexp`** specification in 7.12.7.7 #2 says

> The **`frexp`** functions break a floating-point number into a normalized fraction and an integer exponent. …

And 7.12.7.7 #3 says

> If **`value`** is not a floating-point number or if the integral power is outside the range of **`int`**, the results are unspecified. Otherwise, the **`frexp`** functions return the value *x*, such that *x* has a magnitude in the interval [1/2 , 1) or zero, and **`value`** equals *x* × 2`*p` , when the return type of the function is a standard floating type; …

The second sentence in #3 assumes that the input **`value`** can be expressed exactly as the product of a "normalized fraction", with magnitude in the interval [1/base , 1) or zero, times an integer power of the base. This assumption is true for floating-point numbers in the floating-point model described in 5.2.5.3.3. However, it is not true for numbers in a double-double format when scaling the input to the interval [1/2 , 1) causes inexact underflow in the low part of the number. For example, such scaling of the double-double number 1.0 + **`DBL_TRUE_MIN`** yields 0.5, with the low part underflowing to 0.

This part of the **`frexp`** specification explicitly applies to a "floating-point number". Subclause 5.3.5.3.3 defines a *floating-point number* to be a number in the model. The problematic cases for double-double occur only when the difference in the exponents of the high and low parts is so great that the number cannot be represented exactly in the nominal precision of the type and so is outside the C model. The **`frexp`** specification does not apply to double-double numbers outside of the model. Thus, the aforementioned assumption in the **`frexp`** specification is valid, though the issue of how the specification applies to double-double formats remains.

This highlights a pervasive issue: the definition of “floating-point number” (as a model number) in 5.3.5.3.3 is more restrictive than common usage. Readers might interpret “floating-point number” as any number in a floating type, or as any number that can be expressed in scientific notation. This larger issue is on the CFP list to investigate for C2Y.

Suggested change:

In 7.12.7.7 #3, attach a footnote to:

> If **`value`** is not a floating-point number or if the integral power is outside the range of **`int`**, the results are unspecified.*)

where the footnote is:

> *) Unspecified cases include numbers that are not described by the model for floating-point numbers in 5.3.5.3.3.