Document: WG14/N1214
Date: 2007/03/26
Project: TR 24732
Authors: Jim Thomas, Rich Peterson
Reply to: Rich Peterson <Rich.Peterson@hp.com>

Subject: floating-point model for decimal

This paper notes two problems with the floating-point model assumed in WG14/N1201 (TR 24732 draft of 2006/11/10) and proposes a solution to both.

Problem 1: The TR assumes a floating-point model where normal numbers have one significant digit to the left of the decimal point (a model presented in 754R). This model is reflected in the values of minimum and maximum exponents it shows for each format, which in turn become the values of the <float.h> macros DECnn_MIN_EXP and DECnn_MAX_EXP. However, the C standard has always used a floating-point model where numbers have all significant digits to the right of the radix point, and that model is reflected in the values of macros for minimum and maximum exponents for the generic floating-point types. Thus, the model assumed by the TR is not consistent with the model presented in the C standard. The subtle difference in interpretation of exponent values for decimal versus generic floating-point types would prove confusing to users and complicate moving codes between binary and decimal.

Problem 2: Neither the model assumed by the TR nor the C99 model is elaborated enough to talk conveniently about the integer coefficient and quantum exponent of finite decimal numbers, which are key to decimal semantics. These concepts are important for specifying conversion between decimal floating types and decimal character sequences that meets 754R requirements (see N1215).

The basic idea of this proposal is to adhere to the traditional C model in 5.2.4.2.2 for decimal (as well as generic) floating point and to present an equivalent model that introduces the natural representation and extra semantics for decimal.

Suggested TR changes for section 5:

On page 7, replace 5.2.4.2.2a [1] with:

Macros in <float.h> provide characteristics of floating types in terms of the model presented in 5.2.4.2.2. The prefixes DEC32_, DEC64_, and DEC128_ denote the types _Decimal32, _Decimal64, and _Decimal128 respectively.

For decimal floating-point, it is often convenient to consider an alternate equivalent model where the significand is represented with integer rather than fraction digits: a floating-point number (x) is defined by the model

x = s * b^(e-p) * summation from 1 to p of f[k]*b^(p-k)

where s, b, e, p, and fk are as in 5.2.4.2.2.

The term "quantum exponent" refers to q = e - p and "coefficient" to c = f[1]f[2]...f[p], an integer between 0 and b^p - 1 inclusive. Thus, x = s * c * b^q is represented by the triple of integers (s, c, q).

For binary floating-point following IEC 60559 (and 754R), representations in the C model that have the same numerical value are indistinguishable in the arithmetic. However, for decimal floating-point, representations that have the same numerical value but different quantum exponents, e.g., (1, 10, -1) representing 1.0 and (1, 100, -2) representing 1.00, are

distinguishable. To facilitate exact fixed-point calculation, standard decimal floating-point operations and functions have a "preferred quantum exponent", determined by the quantum exponents of the operands, and they produce a result with that preferred quantum exponent, or as close to it as possible within the limitations of the type. For example, the preferred quantum exponent for addition is the minimum of the quantum exponents of the operands. Hence (1, 123, -2) + (1, 4000, -3) = (1, 5230, -3) or 1.23 + 4.000 = 5.230.

On page 6, in the table of format characteristics, change the Emax/Emin entries to 97/-94, 385/-382, and 6145/-6142 (to adhere to the model in 5.2.4.2.2).

On pages 7 and 8, change the DECxx_MIN_EXP and DECxx_MAX_EXP macro values to match the new table entries above.