Doc. No.: WG14/N1212
Date: 2007-03-26
Project: TR 24732
Authors: Rich Peterson, Jim Thomas
Reply to: Rich Peterson <Rich.Peterson@hp.com>

Subject: Miscellaneous edits to N1201 decimal TR

This paper proposes 15 miscellaneous small-scale edits to WG14/N1201 (TR 24732 draft of 2006/11/10). It does not propose new features, although some changes are more than editorial (e.g. explicit application of 754R semantics to library functions).

The edits are generally independent of each other, and each is numbered and given a title to facilitate discussion.

1. More accurate representation of current practice.

Page 1, section 1.1, paragraph 5, last sentence, change
"The arithmetic used, nowadays,"
to
"The arithmetic used in business applications, nowadays,"

2. Remove speculative claims about hardware vs software performance and market position.

Page 1/2, section 1.1: Omit paragraph 6 (or at least the second sentence).
Omit sentence 2 in paragraph 8.

3. IEEE standard should be final this year.

Page 2, section 1.2 paragraph 1 identifies the IEEE revision document as IEEE-754R. That should be a placeholder to be updated (here and throughout the document) to the specific standard which should be approved before this TR is (i.e. IEEE-754-2007).

4. Feature test macro __STDC_DEC_FP__ should allow for revisions.

Page 5 section 3: the definition of the feature-test macro with a value of 1 does not support implementations that track revisions to this TR or changes that might be made to features in a future revision of the standard that might include decimal floating point support.

Change "The integer constant 1" to "The integer constant 20061110L"

5. More self-descriptive names for minimum subnormal macros.

Bottom of page 8 in section 5, the macros for "minimum positive subnormal decimal floating number" have names DECnn_SUBNORMAL. More self-descriptive, and more consistent with the preceding set of minimum positive normalized decimal floating number macros, would be DECnn_MIN_SUBNORMAL. Alternatively, these macros might be removed from the TR, as they are not directly derived from the decimal floating-point requirements of 754R, and are not mentioned in the rationale document. These macros, along with the generic floating-point versions of them, provide useful information, but it would probably be more appropriate to consider them in some possible future effort to enhance both generic and decimal floating-point support.

6. **Avoid changes not specifically related to decimal floating-point.**

   Bottom of page 8, top of page 9 in section 5, there are macros for the minimum subnormal generic floating numbers. These are not related to decimal floating point, the subject of the TR. And even the decimal versions of these macros are not mentioned in the rationale document. It is suggested that these be removed.

   If they remain in this TR, more self-descriptive names would be [FLT|DBL|LDBL]_MIN_SUBNORMAL.

   By contrast, the [FLT|DBL|LDBL]_MAXDIG10 macros, while not of decimal floating type, are related to the issue of decimal representation, are listed in the rationale document, and are not being proposed for removal.

7. **Refer to 754R rounding rules for conversions, as done in Annex F, rather than duplicating a complex re-wording of the rules.**

   Page 10 section 6.1 paragraph 2a, and page 11 section 6.2 paragraph 4 each mention that a result is correctly rounded, and then go on to describe rounding rules in more detail. The existing C99 Annex F specification handles this situation for binary floating-point simply by referring directly to the 754 specification. It would be simpler and more reliably express the intent to do something similar here:

   In both paragraphs, there is a sentence that ends with the words "correctly rounded.". And following that sentence there is more text and then a list of rounding behaviors. That following text and the rounding behavior list should be removed, and the end of the preceding sentence changed to read "correctly rounded with exceptions raised as specified in 754R."

8. **The document does not mention the important 754R concept for decimal floating-point that each operation produces a result with a preferred exponent.**

   Suggest that at the end of section 4, page 6, another suggested change to C99 be added as follows:

   > Add a new paragraph to 6.5 Expressions, between paragraphs 8 and 9:
   >
   > [8a]Expressions involving decimal floating-point operands are evaluated according to the semantics of 754R, including production of results with the preferred exponent as specified in 754R.

9. "All extra precision and/or range are removed"???

Top of page 11, section 6.2 paragraph 3 last sentence reads: "All extra precision and/or range (for the converted type) are removed." What does this mean? Can it just be removed? It might intend to say that the value is unchanged and the exponent of the result is the same as the exponent of the source, but that would be covered by the change suggested to C99 6.5 in edit **8**, above.

10. Conversion between decimal and complex is fully covered by C99.

Page 11, section 6.3 paragraph 1 describes conversion from decimal to complex (but not vice-versa), and then it is stated that "This is covered by C99 6.3.1.7". In fact because decimal types are real types it appears that C99 6.3.1.7 covers conversions in both directions between decimal float and complex, and there is no need for the first paragraph at all; having it describe conversion in only one direction adds more confusion than clarity. The paragraph should be removed, leaving only the note that "This is covered by C99 6.3.1.7".

11. Comma fault.
Middle of page 12, section 6.4, first sentence of the newly-inserted text: Insert a comma after "complex type" in "generic floating type, complex type or imaginary type".

12. More complete correspondence of C library functions with 754R decimal support.

Page 17 section 8.2 lists a tiny handful of 754R operations that support decimal operands and notes that these operations are implemented as library functions in C. But there are a much larger number of 754R operations that are implemented as C library functions, and it would be useful to have a complete list, as it is also important to note that they provide the 754R semantics, including producing results with the preferred exponent. Suggest the paragraph be replaced by the following:

The headers and library supply a number of functions and macros that implement support for decimal floating point data with the semantics specified in 754R, including producing results with the preferred exponent where appropriate. That support is provided by the following:

From **<math.h>**, the decimal floating-point type versions of:
sqrt, fma, fabs, fmax, fmin, ceil, floor, trunc, round, rint, lround, llround, ilogb, logb, scalbn, scalbln, copysign, nextafter, remainder, isnan, isinf, isfinite, isnormal, signbit, fpclassify, isunordered, isgreater, isgreaterequal, isless, islessequal, quantize, samequantum

From **<fenv.h>**, facilities dealing with decimal context:
feraiseexcept, feclearexcept, fetestexcept, fesetexceptflag, fegetexceptflag, fe_dec_getround, fe_dec_setround

From **<stdio.h>**, decimal floating-point modified format specifiers for:
The printf/scanf family of functions.

From **<stdlib.h>** and <**wchar.h**>, the decimal floating-point type versions of:
strtod, wcstod

From **<wchar.h>**, decimal floating-point modified format specifiers for:
The wide printf/scanf family of functions

13. Accuracy of decimal functions specified by 754R.

Page 19, section 9.3, first paragraph, third sentence reads:
"With the exception of sqrt, max, and min, the accuracy of the decimal floating-point results is implementation-defined."

This should be changed to read:
"With the exception of the decimal floating-point functions listed in 8.2, which have accuracy as specified by 754R, the accuracy of decimal floating-point results is implementation-defined."

If this change were not made, note that max and min in the original version should be fmax and fmin.

14. Correct rounding from decimal strings to decimal types is always easy enough.

Page 30 section 9.6 "Recommended practice" and page 32 section 9.7 "Recommended practice" are identical, applying to conversions to decimal floating-point from multibyte string versus wide string representations. But unlike conversions to binary-radix floating-point representations, it is always easy to produce the correctly-rounded result no matter how many digits are in the input. Suggest that both "Recommended practice" sections be removed, and a new ordinary paragraph 8 be added following paragraph 7:

[8]The result is correctly rounded as specified by 754R.

Note, however, that this edit would be superseded by a separate proposal (in WG14/N1215) for a more complete specification for strtodxx behavior that meets the 754R requirement that the internal representation (including the exponent) be preserved when a decimal floating-point number is printed and then read back in.

15. Examples in 9.8 are misleading/incorrect.

Page 33, section 9.8 last paragraph EXAMPLES. The 2.f in the first example is incorrect in the sense that integer arguments select double functions, so 2. would be more correct (e.g. what if the integer were too large for type float?) And in the second example, the DF suffix is being applied to integer constants. It might also be clearer to present the expansions using casts rather than suffixes, since that more closely matches what actually happens. Finally, since there is also a separate proposal (in WG14/N1213) to remove the _Decimal32 versions of **<math.h>** functions, and to have a mix of integer and _Decimal32 arguments select the _Decimal64 function rather than _Decimal32, it seems simplest to change the second example to avoid expansion to a _Decimal32 function, making this change independent of the decisions reached on the other proposal.

Suggest changing the two examples to read:

```
  pow(2,3.0)  // expands to the double version of pow:
              //   pow((double)2,(double)3.0)

  pow(2,3.DD) // expands to the _Decimal64 version of pow:
              //   powd64((_Decimal64)2,(_Decimal64)3.DD)
```