Doc. No.: WG14/N1227
Date: 2007-03-20
Project: Programming Language C (TR 24732)
Subject: Comments on N1201

The following is a collection of comments on the Decimal TR document N1201.

Misc. edits:

1. 8.1 (pg 16 & 17): Perhaps "or imaginary" should be "nor imaginary" in five places.

2. 8.1 (pg 16 & 17): The constraint for C99 6.5.8p2 can be simplified. Remove ", complex type, or imaginary type". It is already covered by the 1st existing C99 constraint: -- both operands have real type;

3. 9.3 (pg 25) Should there be a footnote attached to 7.12.10 Remainder functions that remquo is missing and why?

4. 9.3 (pg 26) The description is wrong. The interval is [1/10,1) for DFP, and is [1/2,1) for generic FP types.

5. 9.6 (pg 30) strtod*, [#5] "denormalized" seems wrong. Perhaps "subnormal" or "subnormalized" is meant.

6. 9.7 (pg 32) wcstod*, [#5] "denormalized" seems wrong. Perhaps "subnormal" or "subnormalized" is meant.

Comments requiring further committee discussions:

1. I believe, that at the Portland meeting, we agreed that if frexp will be base-10 for DFP arguments, then ldexp should also be base-10 for DFP arguments. I do not see that in the paper.

2. I have a question/issue.

   Given vars:
   ```
   _Decimal32 dfp = ...;
   float      bfp = ...;
   ```

   It is clear to me that
   ```
   if( dfp * bfp ) ...
   ```

   is a constraint violation by DFP TR 8.1

   As I read the DFP TR
   ```
   if( exp1() ? dfp : bfp ) ...
   ```

   is undefined behaviour, not a constraint violation. Seems unusual to me that this operator does not have a constraint violation on mixing DFP with binary FP. Was this done on purpose, or was this something overlooked?

3. Since DEC_INFINITY is of type _Decimal32, quantized64 and quantized128 cannot return DEC_INFINITY. Perhaps, "If both operands are infinity, the result is DEC_INFINITY and the sign is the same as x." should be "If both operands are infinity, the result is x."

4. I do not see how quantize() can overflow. Hence, I do not understand why the spec for quantize mentions overflow.

5. When Decimal FP constants are converted into internal format, are there any constraints on the conversion process? Consider these equivalent values:

```
      1e6DF
     10e5DF
    100e4DF
   1000e3DF
  10000e2DF
 100000e1DF
1000000e0DF
```

   Do they all convert to the same internal format? Or, do they convert into 7 different formats? Implementation defined?

   What about the value zero:

```
0e-95DF
0e0DF
0e+95DF
```

   Same or different internal formats?

6. fp_classify macro issue (see WG14/N???? by Raymond Mak describing the problem)