**WG14 N2062**
**Meeting notes**

## C Floating Point Study Group Teleconference

2016-06-28
9 AM PDT / 12 PM EDT

**Attendees**: Rajan, Jim, Fred, Ian, Mike, David

**New agenda items**:
Conversion of DFP to character strings (Fred's 2016/05/06 email) continued.

**Last meeting action items**:
Mike: Monitor the Part 5 DTS ballot to determine whether or not to put it in the IEEE-754 revision bibliography. - Done.
Passed ballot. Mike will remove the parenthetical that Part 5 is subject to change.
Jim: Check with EDG for testing the off-site backup. - Done.

**New action items**:
Jim: Check one of the files from the EDG backup for testing the off site backup.
Jim: Correction of David H's name in the minutes from last meeting.
Work Item: Discuss writing up this issue (email reflector message 14283) as a DR against C11.
Jim: Create a DR for Part 1 for email reflector message 14280.
Work Item: DR for Part 3 with words needed for email reflector message 14285.
Work Item: DR for Part 3 with words needed for email reflector message 14282 first part.
Work Item: DR for Part 3 with words needed for email reflector message 14282 second part (tgmath).
Work Item: Consider changing the specification to reflect the C11 and IEEE mechanism of conversion to strings to see what it would look like (re Fred's DFP to character string email) in Part 2.

**Next Meeting**:
July 26th, 2016, 12:00 EST, 9:00 PDT
Same teleconference number.

**Discussion**:
IEEE 754 revision:
If nothing is done with twoSum/twoAdd/tailAdd (like IBM's double-double), it'll likely be done in the next couple meetings.
One of the rationale for twoSum is the double format, there is implication for language standards (as a type?).
Expected to have quad instead of double-double for language standards.
In person meeting after Arith23.

Arith23:
Jim and David presenting a summary overview of the TS.

C++ liaison:
Nothing new. Will start soon.

Part 5:
Passed DTS ballot.
Expected to get the publication draft to David Keaton by end of week.
Should be quick turnaround for publish due to only having the date value in macros (us) and cover page to change (ISO).

Email comments from Joseph Myers (sent on 2016/06/21 to WG14 reflector, numbers 14287-14289 and surrounding):
14278: Jim: Re Ian's response. Would a sNaN signal?
Ian: Moving to a register doesn't signal, but may for other things.
Jim: Shouldn't change the type. For the old IEEE standard, the negate C uses was just changing the sign, not anything else.
Fred: Pseudo-* (normals, etc.) may have issues. Can only get them with bit twiddling.
Not part of the standard.
We should mention assignment and cast and unary operators and the reason for this in the footnote.
Jim: This is widening an assignment or cast.
Some implementers have not removed the extra range and precision even with a cast. Making it explicit is required.
Need to propose a DR against the C standard to get this fixed.
*ToDo: All: Discuss writing up this issue (email reflector message 14283) as a DR against C11.
14280: For function returns, the value of the return expression is taken to be the value of the function. No assignment, conversion, etc.
In Annex F it was said there is a conversion to the type of the return type.
Jim: Our intention was that this should apply to function returns. Though we hadn't realized there is no assignment or conversion.
No objections to the proposed idea for a fix. Will need to create a DR for Part 1.
*ToDo: Jim: Create a DR for Part 1 for email reflector message 14280.
14285: Support for making the macro value dependent on the WANT macro.
*ToDo: DR for Part 3 with words needed for email reflector message 14285.
14282: If short float gets added we may need to revisit the comment in 14293.
This is a defect right now. The proposed fix seems good with an addition of a footnote to handle the float case brought up in 14293.
*ToDo: DR for Part 3 with words needed for email reflector message 14282 first part.
For the tgmath part:
Change part 3 only.
*ToDo: DR for Part 3 with words needed for email reflector message 14282 second part (tgmath).

Conversion of DFP to character strings (Fred's 2016/05/06 email) continued.
The C format specifiers give what is needed.
Fred: I know of an implementation that does not round to an infinity (and not raise overflow) from %A.
Our TS specification is wrong. The conversion should happen in the character string domain and not the floating point domain. This is because overflow only happens when converting to a format. With character strings you cannot have an overflow.
Jim: This is not a IEEE-754 conversion. This is a C conversion that is an extension of 754.
David: Don't want to make every language consider string to IEEE numerical formats.
C explicitly says if the field width is not large enough for all the digits in the exponent, the implementation is required to expand the exponent.
Jim: This is not the case here (re exponent).

Jim: The rounding and the result don't fit in the internal format. So we round in the format, then give a representation of that as a character sequence. IEEE says round as a character sequence.
Rajan: An analogous case without any TS's.
Jim: It does it the IEEE way. So different from what we have specified.
David: If you are writing out and reading it back in again. What do you want to read back? Not sure...
*ToDo: Group: Consider changing the specification to reflect the C11 and IEEE mechanism of conversion to strings to see what it would look like (re Fred's DFP to character string email) in Part 2.
Fred: There was that other case regarding leading zeros I mentioned in passing.
Jim: David answered that in that considering them as triples, there is no leading zeros.
Fred: Ex. 0001234, asking for 2 significant digits. Is it 00 or something else?
Jim: Do you mean encoding?
Fred: Due to DFP encodings, the leftmost digit is not always the most significant digit.
Jim: Does not include leading zeros.
Fred: Question was regarding the 754 specification.
David: Significant digits was always specified as first non-zero digit.
Jim: The number of zeros changes the quantum exponent. 0.0 vs 0.00 for example.
Ian: Trailing zero's are significant, not leading. Different from binary where you don't know if trailing digits are significant or not.
Fred will look at 754 and bring this up again if there is a problem.

What should be proposed for the C standard (http://wiki.edg.com/pub/CFP/WebHome/TS_18661_for_C_standard.pdf):
Parts 1/2:
Static/constant rounding mode:
Jim: More like syntactic sugar from a programmer and implementation point of view.
Rajan: Can be very bad performance if implementing this with dynamic rounding.
Part 3:
Allows other floating types. _Float16 for example.
Rajan: Significant overhead for each new type (cos, sin, etc. functions need to be added, encoding/decoding functions, redundant set of functions if the types are the same as standard types).
Jim: Special casing for standard types needed for specification.
Allows importing data for types that are not fully supported (no arithmetic for example).
Rajan: For IEEE-754 format implementers, requires _Float{32, 64, 32x} and whatever long double corresponds to. Can remove that requirement to possibly ease concerns as to work required? Not advocating it, just proposing it.
Part 4:
Should not have major conflict with CPLEX's reduction objects since they deal with collapsing views. They can use our functions if needed for float/double sum/product reductions.
Rajan: This part should explicitly be optional. Can discuss the other parts (likely optional too), but this one has to be.
Part 5:
Pick this up next time.

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative for Canada
C Compiler Development