WG14 N2106
Meeting notes

C Floating Point Study Group Teleconference

2016-12-01
9 AM PDT / 12 PM EDT

Attendees: Rajan, Jim, Fred, Mike, David C., David H.

New agenda items:
Discussion of email from WG14 reflector message 14561

Last meeting action items:
Jim: Check one of the files from the EDG backup for testing the off site backup. - Done.
Jim: DR Set 3: Change positive-signed to non-negative. - Done (discussed later).

New action items:
Jim: Call David Keaton and ask for advice on how and when to present proposals for Parts 3-5.
Jim: Write up a proposed TC for DR501 to make the DECIMAL_DIG macro obsolescent.
All: Make sure we're OK with DDR9/DR11's change.
Jim: Ask the IEEE-754 revision mailing list if the payload for NaN's must be non-negative (0 and up allowed).
Jim: Reflector message 14561: Fix up "macro argument" to something along the lines of 7.25#3.

Next Meeting:
Tuesday January 24th, 2017, 12:00 EDT, 9:00 PST
Same teleconference number.

Discussion:
IEEE 754 revision:
Fred: There was discussion about -0/0 on the IEEE chain. Is this something that we need to bring into the C standard.
David C: fmax/fmin (C) issues (maxNum, minNum in IEEE): Looking for insights into what to do.
David H: Was it a mistake in IEEE 2008? If so, we can remove the item, or add a new item that does what we want (as a recommendation) so both ways don't break anyone.
David H: augmented/twoSum are added and settled down now.
Still issues with multiple exceptions, like signalling NaNs. Needs to be clarified.
Sub-exceptions bring up other issues.
Jim: The fmax/fmin in C adhere to the IEEE-754 operation right now. If it is removed from IEEE, we'd have to change the binding table.
Mike: Does anyone claim conformance to the IEEE-754:2008 standard?
David C: Never saw anyone.
Fred: No compiler I've seen defines the conformance macro.
Jim: HP did for Annex F (but that's the IEEE 754:1985 standard).

C++ liaison:
No update.

What should be proposed for the C standard (C2X):
Current status:
Part 1 and 2 are intended to be included.

Proposals for the other 3 parts:
Next C meeting mailing deadline is March 6th, 2017.
We need to decide what to do by the February meeting.
David H: How do the other C members feel about the other parts?
Part 3: Seemed to be not wanting to be implemented by some members, but not real resistance.
Part 4: Neutral opinion. Library so more likely to be accepted.
Part 5: Negative opinion with regards to exception handling (try/catch).
Jim: Since they will all be optional, others not implementing it should not be too much of a roadblock.
Part 3 is a wide ranging type so would prefer to see implementation experience.
Fred: A possible objection to Part 4 is if there is no implementation experience, do we have all the corner cases handled?
Jim: tanpi may have an argument (ones we added since it seemed to be omissions). The other two functions we added were added to the new IEEE-754 revision, but not this one.
Rajan: We can propose part 4 as is with the caveat of removing tanpi if desired.
Jim: For part 5, seems like a regularization of things implementations already do. This would provide portability. Perhaps separate out try/catch, and possibly the other parts that change flow control.
Fred: We can propose part 5 without try/catch.
Jim: How about break?
Fred: For part 4, should they functions go in the main body or an annex?
Jim: Like the TS, some in both the main body or in Annex F. As stated now, they will stay an optional feature.
Fred: Some compiler implementers may not have enough knowledge to do the alternate exception handling.
The hardware does not support everything. The compiler would have to do the mapping.
David H: Same issue with anything else for a compiler as it always maps from language to hardware.
Fred: One case is having a user requesting denorms flushed to zero, can be much slower than expected.
David H: At some point you have to know what your hardware is capable of.
Jim: The optimization pragma allows flush to zero, doesn't require it. For alternate exception handling, it has to be done, so it could require a lot of work.
Jim: Sub-exceptions is another complexity we need to talk about.
David H: Perhaps don't suggest this in the first round to the C committee?
Supporting sub-exceptions does require a lot of library and compiler support since the hardware normally does not support them all.
Jim: Note that part 5 is optional. Alternate exception handling is an optional part of part 5, and sub-exceptions are a "should" so triply optional.
*Jim: Call David Keaton and ask for advice on how and when to present these.
Present part 3 as is, but not pushed, and make clear the extent of the change, and have a fallback of referencing it if not accepted
Present part 4 with tanpi as a possible removal
Present part 5 without try/catch

DRs:
Set 2:
DR501 (DECIMAL_DIG):
For types wider than long double, like the types in Part 3, allowing DECIMAL_DIG to be larger can handle that.
Jim: Even a committee response saying it can be larger without a textual change in the standard would be good enough.
Since we have type specific ones (including part 3) maybe we don't need this.
Fred: The current wording says largest floating type. So it should be fine.
Jim: You should be able to have wording that supports Part 3 and C11 without part 3.
If you have a type that is wider than long double then DECIMAL_DIG would be larger. But C11 as it is only has the largest floating type as long double so it doesn't conform.
A footnote would be better.
*Jim: We can make it obsolescent since there are type specific macros already.
DDR9/DR11 (%a formatting):
Need to watch this to make sure the late paper gets in and doesn't get missed.
*All: Make sure we're OK with DDR9/DR11's change.

Set 3:
DDR1 (November 1 email from Jim re payload with positive floating point integer)/Last meeting action item:
Precluded the payload from being zero.
Part 1 says 60559 says the payload is an integer value encoding in the significand.
754 says for decimal that the maximum is a certain value which implicitly says positive integers?
Fred: Isn't a payload of 0 for one type of NaN an infinity?
Jim: Maybe get confirmation of this from 754 before we decide on this here?

Other:
WG14 reflector message 14561 (Joseph Myers):
Jim sent an email discussing this issue (2016/12/01).
First change parallels DDR7.
Rajan: Second change: Does macro argument type make sense? Macro arguments don't have types since they are substituted during preprocessing and before we have types.
Need to match what we say in 7.25#3.
Jim: This does give the implementation a choice of functions to use (anything that is wide enough to produce the given result type). Since these are correctly rounded functions, it doesn't matter which one we pick.

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative for Canada
C Compiler Development
Cfp-interest@oakapple.net
http://mailman.oakapple.net/mailman/listinfo/cfp-interest