WG14 N2793
Meeting notes

C Floating Point Study Group Teleconference

2021-07-21
8 AM PDT / 11 AM EDT / 3 PM UTC

Attendees: Rajan, Jim, Ian, Fred, Mike, Damian, David H.,

New agenda items (https://wiki.edg.com/pub/CFP/WebHome/CFP_meeting_agenda_20210721-update2.pdf):
Fred: Request from an implementor to detect tininess before and after rounding.

Carry-over action items:
Fred: WG14 N2714 Add "a" before "NaN" in last bullet.
Jim: [CFP 1997]: Range error definitions of overflow and underflow.

Last meeting action items (done unless specified otherwise):
Jim/Rajan: Look into what we proposed for TS part 4 and the status of it.
Jim: Put the CFP C23 changes summary page for the C++ liaison study group on the agenda for next meeting.
Rajan: Re CFP2024: Ask Paul if there are other values and if it is a real problem.
Fred: Draft a response to the remquo issue brought up in CFP1974.
Fred/Jim: Create a paper re CFP2030 to propose a response.
Rajan: Send N2751 to JeanHeyd.

New action items:
All: Review CFP 2060.
Fred: Submit CFP 2062 to WG14.
Fred: Write up a proposal to remove the *_HAS_SUBNORM macros.
All: Look into the email history to find out why we chose to make float and _Float32 separate and distinct types.
Rajan: See what we can propose for freestanding and IEEE 754 and send it to the CFP mailing list.
Jim: Add Detect tininess before and after rounding to next meeting’s agenda.

Next Meeting(s):
Same time slot. Note: New day.
Tuesday, August 17th, 2021, 4PM UTC
ISO Zoom teleconference
Please notify the group if this time slot does not work.

C++ liaison:
CFP C23 changes summary page for the C++ liaison study group [Cfp-interest 2060]
Rajan: Minor updates since.
Mike: What does conditionally added mean?
Rajan: Depends on whether a feature macro is present or not.
Jim: Put conditionally added at the start of part 2 possibly?
Rajan: But the format specifiers have to be there so it is not conditional. Ex. A lint type tool could not diagnose the decimal-floating point format specifier.
AI: All: Review CFP 2060.

C23 integration

Latest C2X draft: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2{596,573,478}.pdf also as link on CFP wiki

Part 2

Part 3

Part 4ab

Part 5abcd

IEC 60559:2020 support

Carry over action items:

Fred: WG14 N2714 Add "a" before "NaN" in last bullet

Jim: [CFP 1997]: Range error definitions of overflow and underflow.

See N2746 and CFP {2038-69} titled Underflow.

Jim: New definitions for C over/underflow. The underflow case includes exact minimum normalized number. Fred wants to remove that case.

Jim: Previously we had "extraordinary roundoff error" and now the "ordinary accuracy" is just as ill defined. We could add a footnote to try and clarify this saying the functions result doesn't have extreme magnitude or something to that effect.

David: Trying to apply IEEE thinking to non-IEEE implementations. We can't figure it out and we're supposed to be the experts. I think the previous wording is better, but not by much. It doesn't solve anything.

Jim: I used "ordinary accuracy" to avoid mentioning roundoff error.

David/Fred: The 2nd last change in CFP 2080 makes a number that is slightly larger than the minimum normalized number being rounded down an underflow when it wasn't before.

Jim: For the last change, is it OK?

David/Fred: That seems OK.

Fred: The "may" there gives implementations leeway.

Jim: That is a weakness of this specification. Underflow doesn't have to be reported. In a sense all underflows do not have to be reported. A required underflow is given in the specification of the function. This does remove the problem in the previous change from the definition.

Fred: I'm happy with the last change.

David: It may be as good as we could get.

Jim: It is completely up to the implementation on what to do with underflow. So that helps.

Mike: Ordinary accuracy is not defined even in 754.

Jim: The footnote would say the results are not effected by extreme magnitude being ordinary accuracy. We don't have a model to talk about this in real accurate terms.

Mike: If we don't have it, how can a reader or an implementor understand it.

Fred: You have to mention accuracy since a zero result from rounding should be an underflow.

Jim: We can say "without reduced accuracy" at the end?

David: Or even "default accuracy"?

Fred: Or say "not exact zero" meaning an exact zero is not underflow.

Jim: If we can't agree on something, I don't know if it is reasonable for WG14 to do it.

David: "Or when rounding causes the magnitude of the result to increase to the minimum normalized number of the type." could be added.

Jim: This kind of definition is in tandem with the overflow. It would be weird if we used a totally different set of terms and definitions from overflow.

Rajan: Before Mike and David leave, please post any new wordings you can think of that would be better for this to the CFP mailing list.

Mike/David: Yes, agreed.

Previous meeting action items:

Jim/Rajan: Look into what we proposed for TS part 4 and the status of it.

See CFP 2056.

Jim: Put the CFP C23 changes summary page for the C++ liaison study group on the agenda for next meeting.

Rajan: Re CFP2024: Ask Paul if there are other values and if it is a real problem.
See CFP 2024.
Paul asked us to drop the issue.
Ian: Should we document this may be an incompatibility?
Rajan: No, since there are other ones like this. Ex. ERANGE, etc. which can have different values between implementations. It allows flexibility.
Ian: OK, makes sense.

Fred: Draft a response to the remquo issue brought up in CFP1974.
See CFP {1974-2062} titled remquo
AI: Fred: Submit CFP 2062 to WG14.

Fred/Jim: Create a paper re CFP2030 to propose a response.
See CFP {2025-2070} titled "Subnormals" or "FLT_HAS_SUBNORM"
Jim: CFP2070: -1, 0 and 1 stay the same functionally. 2 and 4 are new categories.
Fred: I like this.
Damian: Seems orthogonal. So I like it.
Rajan: Afraid this may open up a rabbits hole.
Fred: From ARM and Intel, they have control bits that say flush operands and one for results.
Damian: What other CPU's do we need to worry about?
Rajan: z as well.
Fred: abs and negate never flush even in flushing mode.
Jim: Those generally are not arithmetical operations, they are bit operations. The "typically" is there to cover that as well.
Damian: Does this cover what RISC 5 does as well?
Jim: I don't know what RISC 5 does.
Damian: Neither do I.
David: There is an escape clause so it should cover all cases. Also the default rounding mode shouldn't be the default execution mode. To take into account the various flush bits in various architectures.
Jim: They could be regarded as rounding or not.
David: It is not intended to be dynamic is it?
Fred: Too late for that, but it would have been nice if it was.
David: I favor removing it from the draft.
Fred: This came from a user or implementor.
Jim: I took it as what am I supposed to set it to?
David: I would say execution mode or runtime mode. Whichever is more commonly used in the standard.
Jim: Is it clear?
Rajan: No, the "typically" is too hand wavy to give a good result on both implementation and user side.
David: Yes, agree.
Ian: Anything other than abs/neg/-abs?
Jim: I would be surprised if everything, all libraries were consistent.
Jim: The current definition is not clear in 2 or 3 different ways. It is in flux in terms of technology and implementations. Doesn't seem important to standardize it in this way.
Mike: Makes sense.
David: 0 conforms completely to IEEE. 1 conforms to alternate exception handling. In fact, you could remove 1 and just make it that it has IEEE or not. First preference is removal.
Fred: I guess we can propose removal.
Jim: It does keep the questions for implementations on what they do for issubnormal, nextafter, etc.

AI: Fred: Write up a proposal to remove the *_HAS_SUBNORM macros.

Rajan: Send N2751 to JeanHeyd.
See N2751.

Other issues:

Issues with float and _Float32 being different types
See SC22WG14.19485, 19487, 19488, 19491, 19492
Jim: A concern I heard was passing an array of _Float32 to a function in a library that takes float arrays.
Rajan: With the same representation and alignment in practice, C++ -> C should work.
Jim: If C++ makes them the same type, would there be any issue?
Rajan: In practice, I can't see an issue.
Jim: I think _Complex and other issues with typedef made us consider it separate types.
Fred: I wonder if Joseph Myers would have issues with this.
Jim: He was. In earlier drafts we didn't have it clear and he wanted it clear.
AI: All: Look into the email history to find out why we chose what we did.

Issues with free-standing implementations
See SC22WG14.19444, 19445
Rajan: We really should not have brought in TLS anything or locale anything into freestanding.
Jim: The numeric conversion functions should also be allowed to not have errno.
Rajan: The decimal point and whitespace is an issue for freestanding due to bringing in locales.
Jim: Without the str* functions you couldn't fully conform to the spec for freestanding. We could nail down the whitespace characters and the decimal point character. Or remove whitespace and nail down the decimal point character.
AI: Rajan: See what we can propose for freestanding and IEEE 754.

Regards,

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative for Canada
C Compiler Development
Contact: rbhakta@us.ibm.com, Rajan Bhakta/Houston/IBM