WG14 N2831
Meeting notes

C Floating Point Study Group Teleconference

2021-09-29
8 AM PDT / 11 AM EDT / 3 PM UTC

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

New agenda items (https://wiki.edg.com/pub/CFP/WebHome/CFP_meeting_agenda_20210929-update2.pdf):
None

Carry-over action items:
None

Last meeting action items (done unless specified otherwise, details below)**:**
Jim: Update N2746 with CFP 2090.
Fred: Send CFP 2094 to WG14.
Rajan: Ensure the C/C++ study group presentation sees P1467r4.
Rajan: Draft words for making freestanding support for CFP for both options in CFP2085.
Jim: Send CFP2089 as an update to N2672 barring any issues from this group.
All: Look over CFP2096 and give feedback within 2 weeks.

New action items:
Jim: Propose new wording for N2716's change that WG14 did not accept as is.
Rajan: Get a document number for CFP2140 and send an updated document to CFP before WG14.
Jim: Put the formulas into CFP2130's text and show CFP before submitting it to WG14.
Jim: Reword the footnote for CFP2153 and show it to CFP before submitting it to WG14.
Jim: Send out a change to CFP changing the wording of the FLT_MAX_EXP family of macros to say they are for normalized numbers.
Anyone: Look into the use of the term "floating-point number".
Rajan: Get back to Aaron about the FLT_EVAL_METHOD constant value question and point him to part 5 of the TS which does have changing values.
All: Look at CFP2136 and respond within a week.
Jim: Submit a paper to remove the promotion rules as per discussion in CFP2141.
Jim: Propose the change in CFP2154 to WG14.

Next Meeting(s):
CFP, C++ liaison, and C++ Numerics study groups:
October 6th, 2021.
See CFP2151 for details.

CFP:
Same time slot.
Wednesday, October 13th, 2021, 3PM UTC
ISO Zoom teleconference
Please notify the group if this time slot does not work.

WG14 meeting:

See CFP 2128, 2129, 2131.

Jim to propose a response to WG14's request for different wording for the numerically equal paper along the lines of CFP2145.

C++ Liaison:

CFP C23 changes summary page for the C++ liaison study group (See CFP 2060, 2142, 2147, 2151, 2157)

David O: Beyond CFP 2142, there is also implicit conversion that need to be looked at. C does conversions between any floating point types implicitly like other arithmetic types. C++ does not not. Lossy conversions have to be explicit.

Jim: We did change that, but it was not seen by WG14. Oversight on our part. It's later in the agenda. There is a reason why we do what we do for C. Is there a reason for C++?

David O: It matches what we do for the integer types.

Jim: The float type may not the same as _Float32 in C++, whereas in C, we prefer IEEE over non-IEEE.

David O: The biggest issue is the naming. In C they are optional keywords. In C++ they are typedefs in the std namespace.

Note: We do have the October 6th meeting with C++. See CFP2151 for detail.

C23 integration:

Latest C2X drafts: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2596.pdf http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2573.pdfhttp://www.open-std.org/jtc1/sc22/wg14/www/docs/n2478.pdf

IEC 60559:2020 support

Action item resolutions:

Jim: Overflow/Underflow definitions: Update N2746 with CFP 2090 (See CFP 2116, 2116, 2118, 2122, 2133)

Done.

Fred: Send CFP2094 to WG14 (subnormal macros)

See N2797.

Rajan: Ensure the C/C++ study group presentation sees P1467r4

Done.

Rajan: Draft words for making freestanding support for CFP for both options in CFP2085 (CFP 2140, others)

Fred: The original errno requirement was from C89 for the strtod functions. We missed updating it in C99 to remove errno.

Jim: I see how freestanding does not want global state, but the second alternative still has it.

Rajan: Yes, the first alternative doesn't require either, while the second requires at least TLS or global state.

Fred: strtol has errno too. This should apply to it.

David O: I don't think this would be good for the next CFP/C++ meeting and Ben Craig is the one interested in this.

AI: Rajan: Get a document number for CFP2140 and send an updated document to CFP before WG14.

Fred: The strtol is a numeric conversion function that refers to errno. Perhaps we should say floating point numeric conversion functions to avoid adding strtol in the list.

Rajan: I can name the functions instead of saying floating point numeric conversion functions.

Jim: I prefer naming the functions. Or both.

Jim: Send CFP2089 as an update to N2672 barring any issues from this group (See CFP 2112-2114, 2117, 2119, 2132, ...)
Looks good.

All: Look over CFP2096 and give feedback within 2 weeks (See CFP 2105, ...)
Jim: Better to make the change in 7.12 instead of the floating point model in 5.2.4.2.2.
Mike: I like CFP2134, but I get your point about it not being usual for C standard readers. For decimal we don't normalize at all so it doesn't make sense. For binary it does.
Jim: I argue it does for both binary and decimal.
Mike: The first paragraph works for both. You can clarify it for binary users by saying how normalizing refers to normal. There is no normalizing concept for decimal. "Minimum normalized floating-point number" means nothing in decimal.
Jim: It is the one with the minimum exponent in the cohort. A leading non-zero digit in the model in 5.2.4.2.2. That translates into a full co-efficient.
Mike: If that is true, that is unfortunate for decimal. It is better if you have a clear formula and definition in one place. I would argue having both paragraphs there is better. i.e "in other words" to join the two definitions.
Rajan: Not normally good to have two definitions in case of disagreements between the two.
Mike: Perhaps put one as a footnote.
Jim: Perhaps put the formula in there (in parenthesis). Or vice versa.
Fred: The DFP definition uses the word normalized.
Mike: That slipped by me.
Ian: I prefer the formula first.
AI: Jim: Put the formulas into CFP2130's text and show CFP before submitting it to WG14.

Other issues:
Underflow with exact subnormal (See CFP 2106, ..., 2172)
Fred: For F.9.2 for DFP the transformations wouldn't work as being equivalent.
Jim: The beginning of the section qualifies this so we should be OK.
Ian: CFP2158 is what I implemented in the IBM compilers.
Looks good.
Jim: CFP2153.
Fred: I don't see how this would cause a difference for 1*x.
Jim: 1*x would get the precision of x shortened.
Jim: The notes are supposed to be guidance and not complete coverage.
Rajan: Agree. We can't cover every case.
Fred: Me too.
Jim: Anyone thing we should delete the dynamic rounding precision part?
Rajan: Can add in a "for example" at the beginning or later to make it clearer. It is fine how it is now for my reading.
Jim: Will look into doing a change and send it out for a quick email poll.

Supernormal numbers (See CFP N2138, ...)
Jim: The underlying problem is we're not consistent what we mean by floating point number. Sometimes it is a model number, other times it is a number in a floating point type.
Rajan: Vicents change does make more English obvious what MAX refers to. I am actually OK with and prefer the change to add "normalized" to the macro descriptions.
AI: Jim: Send out a change to CFP changing the wording of the FLT_MAX_EXP family of macros to say they are for normalized numbers.
David H: I think trying to specify all the types of numbers and types is risky business.
Jim: Specifying the arithmetic of model numbers was what C wanted to do but then tried to generalize it to allow double-double and other arithmetic. Unfortunately we're in between those two approaches at this point.
Jim: It would be interesting to go through the 21 instances of floating-point number and see which meaning they have.
AI: Anyone: Look into the use of the term "floating-point number".

Floating-point numbers (See CFP 2110)

FP_NAN (See CFP 2120, ...)
Rajan: IBM's compiler defines these for HEX float formats for C99 and up.
Fred: OK with doing nothing here.

FLT_EVAL_METHOD (See CFP 2126, 2127)
Jim: Perhaps pass a reference to part 5 which shows changing values for FLT_EVAL_METHOD based on block scope pragmas.
AI: Rajan: Get back to Aaron about the FLT_EVAL_METHOD constant value question and point him to part 5 of the TS which does have changing values.

Tgmath.h narrowing macros with integer arguments (See CFP 2136)
AI: All: Look at CFP2136 and respond within a week.

Intended removal of promotions in Part 3 (See CFP 2139, 2141)
David O: This is not the issue I raised. My issue was _Float64->_Float32 can be done implicitly in C. In C++ it can't.
Jim: Anything C++ wants here?
David O: Nothing. There is a difference, but that is something that we need to live with. It is an education issue, not a technical issue. We are OK with only float->double promotions.
AI: Jim: Submit a paper to remove the promotion rules as per discussion in CFP2141.

feraiseexcept update (See CFP 2154)
David H, Ian: Looks good.
AI: Jim: Propose the change in CFP2154 to WG14.