WG14 N3083
Meeting notes

C Floating Point Study Group Teleconference

2023-01-04 and 2023-01-05
8 AM PST / 10 PM EST / 4 PM UTC

Attendees: Rajan, Jim, Fred, Mike, David H, Damian, Hans Boehm, Ian

New agenda items (https://wiki.edg.com/pub/CFP/WebHome/CFP_meeting_agenda_20230104-update.pdf):
None.

Previous meeting minutes:
CFP2563 (Posted on CFP wiki)

Next Meeting(s):
January 18th, 2023, 4PM UTC – **If needed only.** If the meeting is not needed, the next meeting date will be determined based on the WG14 meeting results.
ISO Zoom teleconference
Please notify the group if this time slot does not work.
Note: The WG14 meeting is January 23-27, 2023

Carry over action items:
Done unless specified otherwise.
Details below in "Carry-over action items results" section.

Last meeting action items:
Done unless specified otherwise.
Details below in “Action items results” section.

New action items:
All: Look at NB comments needing conclusions (US26-075, GB157, GB279, GB287) – Done by second day of meeting
Jim: Look to provide words for constexpr float issues saying something like "for determination of constraint violations, the translation time state is assumed" for GB279 and US26-75.

C++ liaison:
Rounding direction specific functions. Discussion below. – Did not get to this item.

C23 integration:
CD ballot done (https://wiki.edg.com/pub/CFP/WebHome/ISO-IEC_JTC_1-SC_22_N5777_Text_for_CD_9899.pdf).
See CFP2558

Carry-over action items results:

David H: Get an example for the scaled reduction functions (perhaps by asking Jason or Jim or looking into the IEEE references). - Not done.

See https://754r.ucbtest.org/background/traps-and-wraps.txt

David H: Get an example for the augmented arithmetic functions (perhaps by asking Jason or Jim or looking into the IEEE references). - Not done.

Action items results (from previous meeting):

Rajan/Jim/Fred: Provide the CD comments to the CFP list for review.

See CFP2569.

All: Review CD comments related to CFP (topic continued over two days).

See CFP2572, CFP2574-2577

https://wiki.edg.com/pub/CFP/WebHome/NB_comments_on_CD-JWT-20230102.pdf

GB7: Third change (6.7.1#16): binary32 is not applicable here so it should be "float" for the first and "IEC 60559 binary32" for the second. - All agreed.

US6: Mike has checked them all. Agree to the change. N3xxx has the correct font already. - Agreed with the suggested NB comment changes.

GB63: Mike has checked them all. Agree to the change. - Agreed.

US75 (US26-075): Real to complex (from N3071):

Mike: What do existing implementations do with this?

Jim: There are none. This is new for C.

Hans: C++ is trying to create new areas constexpr is allowed in. Complex is not C++ so this is not something we have handled.

Jim: The real to complex should be allowed. - Agreed.

(Complex to real) Jim: This is a change in value.

Hans: This is C specific. In C++ it needs to be compile time evaluable for some arguments. Weaker than what is in C.

Jim: Constraint violation suggested due to value preservation failing. - Agreed.

Question 4 (Quiet NaNs) Fred: Is NaN even representable in the type?

Jim: It is since the NaN is there.

Fred: Earlier we say range is -Inf to Inf without mentioning NaNs.

Jim: That is for range of representable values. Different use.

Question 5 (SNaN) Jim: Constraint violation in all cases.

Fred: To the same type, it should be OK right?

Jim: Correct.

Fred: Your answer doesn't talk about same type.

Jim: Correct, the question didn't ask that. I can note that. - Agreed.

Question 6 (standard to decimal): Jim: Should not be a constraint violation. - Mostly agreed.

Question 7 (decimal to standard): Jim: Violation. Losing quantum. - Agreed.

Mike: Finite should be violation, but others should not.

Fred: Easier to specify and implement as a whole instead of exception cases.

Mike: OK. - Mostly agreed.

Question 8 (quantum exponents) Jim: Change in value so not allowed. - Agreed.

Main issue: Jim: Only needed if FENV_ACCESS is on.

Fred: If the value depends on state, it is prohibited.

Jim: Do you mean "potentially depends on state"?

Fred: Yes. I wouldn't make it depend on FENV_ACCESS. Just blanket prohibition.

Mike: That would prohibit most arithmetic.

Jim: Inexact additions would be prohibited.

Fred: Agreed. Exact is fine. If it depends on state, the value can wobble.

Mike: Correct.

Hans: C++ is going in the other direction like allowing trig functions for constexpr.

Jim: If the initializer is dependent on state when evaluated at execution time, it is prohibited.

Mike: Another way of putting it is if the value can change for any reason, it is prohibited.

Jim: Would the analysis allow changing the value later on?

Jim: For Fred's, it always depends on state since the default state is a state.

*No resolution beyond there needs to be a change since it is a problem.

GB127: Jim: In another place we say the preferred quantum exponent is implementation defined. - Mostly agree.

GB147: Jim: The floating point annexes weren't structured with merging into other sections, but it seems OK since nothing is said on how the merging is done. - Agreed.

GB149: No issues. - Agreed.

GB151: Jim: == should have been "is equivalent to" due to NaNs.

Mike: These are footnotes so non-normative and sloppy language is fine here. We could just leave the "equivalent" text.

Fred: The problem is if I is complex instead of imaginary, Infinity * it gives a Nan instead of infinity.

Jim: Using CMPLX defeats the point of the footnote. By saying "if imaginary types are supported" that works.

Mike: Just say that formulation can be used. Or we can just delete the footnote.

- Mostly agreed to Jim's change.

GB152: No issues. - Agreed.

*GB153: Jim: Don't see an issue. - Disagree with the NB comment.

US155: Editorial, not technical - Agreed.

GB156: Jim: Macros are not pragma expanded. We did have inconsistencies elsewhere as well so better to accept the change. - Agreed.

*GB157: Possibly not needed. Need to review further.

GB163: Fred: It is possible an implementation may use an expression that is a valid constant expression but not an arithmetic constant expression.

- Mostly agreed.

GB164: Jim: The previous semantics should NOT be restored. It is more important to report the overflow if you get a finite result. - Mostly agreed.

US42-169: Add the returns section. Similar to 7.24.1.3 - Agreed (add a return section).

*US43-170: Jim: Both implied, so not normative. – Disagree.

GB173: No issues. - Agreed.

US56-187: No issues. - Agreed.

US57-189: No issues. - Agreed.

GB229: Jim: If we intended default argument promotions, we wouldn't have added the float versions of the functions. - Agreed.

GB230: No issues. - Agreed.

GB267: Mike: Program font vs pretty font. The comment is correct. - Agreed.

GB268: Jim: If the value is Inf or NaN, the conversion to int would raise an exception. The spec makes it unspecified. - Agreed.

Jim: Note that F.10.3.7 uses exp while 7.12.6.7 uses p.

GB269: Generally we do say the infinity cases. - Agreed.

US68-270/GB271: Jim: 271's wording is better. - Agreed.

US71-275: Jim: The line doesn't add anything. Suggest deleting. - Agreed (delete).

GB276: No issues. - Agreed.

US67-278: No issues. - Mostly agreed.

GB279: Fred: constexpr should not apply to auto.

Jim: There is no need for the feature then.

*Same as US26-75 - Need to discuss further.

GB286: Jim: For strfromd you don't need the wide version due to composing 2 other functions. This does not apply to strto though, and we need it. They should be added, but unlikely to be added now.

Rajan: We could say this was missed and should be present. But if we can't do it, it can be in a future standard.

Jim: Put it in "Future directions". - Agreed (add wide string analogs).

GB287: Jim: For decimal, hex input is prohibited in the base standard. Annex H allows hexadecimal input for those same functions. Adding the functions seems like a big change at this stage.

Jim: H.12.2 can introduce strtohex functions. But that seems messy and not right.

Rajan: We can say it is implementation defined in strtodN whether hex inputs are supported or not. That way Annex H becomes an extension, and allows existing implementations to keep diagnostics if they don't support Annex H.

Fred: Another option is to allow 7 to allow hex and point to Annex H.

Jim: We can have a footnote to say it is intended that implementations follow Annex H.

*Direction is to do implementation defined in section 7 and pointing to Annex H if implemented.

GB288: Jim: We do need words. "The preferred quantum exponent for the result is zero if it is exactly represented in the type. It is the least possible exponent if it is not exactly represented in the type." - Agreed (needs new words).

GB292: No issues. - Agreed.

*US74-303: Jim: Not needed since interchange formats and the least value greater than 1 is normalized. Note that normalized doesn't appear for *_EPSILON for decimal. – Disagree with comment.

GB322: Jim: Also missing crootn and crsqrt (even the double versions).

Fred: They are present. - Agreed

Continuing NB comment discussion (2023/01/05): Attendees: Rajan, Jim, Damian, David H, Fred

Missed issues (from yesterday):

GB14: Jim: If the constants are wider, you can't use them with constexpr since there is a change in value.

Fred: You can add a cast since they are allowed for arithmetic constant expressions.

Damian: Can't you do "constexpr float a = (float)0.1;"?

Jim: Yes, though rounding direction can change it. There is another comment for that. - Agreed.

US5-18: No issues. - Agreed.

Revisit issues needing a look:

US26-75: Jim: The problem with Fred's approach would vary between implementations. Ex. One could only do the default rounding mode so the constexpr evaluation would be valid on that implementation but not in another.

Rajan: Not an issue for strictly conforming programs.

Jim: Lets look at GB279 first and come back to this. It is related.

GB279: Jim: Why can't it have the same semantics as static initialization?

Damian: The benefit of constexpr allows constexpr variables to be used on the right hand side of other constexpr initializations.

Jim: So looking at "like static" or "independence of mode". This means "constexpr double x = (double)(1.0/3.0);" would be a constraint violation for independence of mode but not for static.

Rajan: We could propose both and let WG14 decide.

Fred: Probably the best choice.

Damian: constexpr can use auto/register/static as well. See 6.7.1#14 for a similar example.

Jim: Doesn't change the rounding mode. Can say "for determination of constraint violations, the translation time mode is assumed."

Rajan: Mode should be state or something else since it is not just rounding mode.

Jim: Right, wide evaluation would also effect this.

^ToDo: Jim: Look to provide words for constexpr float issues saying something like "for determination of constraint violations, the translation time state is assumed" for GB279 and US26-75.

Jim: What does this do to C++ compatibility? Side issue.

GB127: No issues. - Agreed.

Fred: 6.7.10#11 is the related clause in the standard.

GB151: Jim: Issues with -0 and +0, since they are equal but not equivalent. Instead we can say "For a complex variable z, z and CMPLX(…) are equivalent expressions, and z and creal(z) … are equivalent expressions if imaginary types are supported." - Mostly agreed.

GB157: No issues. - Agreed.

GB163: No issues. - Agreed.

GB286: Jim: Add in 7.33.20 post paragraph 1 saying wcstofN are reserved for wide character analogs of the strtofN functions. - Agreed.

GB287: Jim: Suggested change is too large at this stage. Instead change 7.24.1.6#4 bullet 1 from "it is not a hexadecimal floating number" to "Whether the subject sequence may be hexadecimal floating number is implementation defined" with Recommended Practice being "Rounding for hexadecimal input should follow the method in H.12.2". Also update the "0x1.8p+4" example to say "If hexadecimal input is accepted (+1, 24, 0). If hexadecimal input is not accepted, ".

Fred: Should we add to future directions that strtod will accept hexadecimal input in the future.

Rajan: I disagree. We don't know if WG14 will accept that.

Fred: Should we have a feature test macro for this then?

Jim: Want to keep it as simple as possible right now. - Agreed.

GB288: Jim: Updated the text in the 20230105 document. - Agreed.

Fred: Can you get an exact value for a value smaller than one so the preferred exponent wouldn't be zero?

Jim: 0.8. That's exact.

Fred: My concern is you are requiring the preferred quantum exponent to be zero for exact. It should be as close to zero as possible.

Jim: That is the meaning of preferred quantum exponent. It is not the quantum exponent.

Jim: Clean up participants list on the wiki.
See CFP2567.
Did not get to this item.

Other issues:

Review TS part 4 revision
See [Cfp-interest 2454] Re: post-C23 update for TS 18661-4
Did not get to this item.

TS part 5 revision
See [Cfp-interest 2560] TS 18661-5 revision
Did not get to this item.

Rounding direction specific functions
[Cfp-interest 2561] Fwd: Floating point environment
Did not get to this item.

Others:

None

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative (Canada, USA), PL22.11 Chair
C/C++ Compiler Development