

CFP Meeting

2026/09/30: 8:00 AM PDT/3:00 PM UTC

[Please submit proposed changes to these minutes to Jerome or to the group. Revision changes appear at the bottom.]

Attendees and introductions

Rajan Bhakta, Jerome Coonen, Damian McGuckin, Ariel Burton, Joshua Cranmer, David Hough, Tue Ly

Agenda

<https://cfp-wiki.esi.com.au/pub/CFP/WebHome/n3940.pdf>

Previous meeting notes

<https://cfp-wiki.esi.com.au/pub/CFP/WebHome/260826-Notes.pdf> - 2026/08/26 Minutes

Study group logistics

Next meeting: 28 October 2026, 8:00 AM PDT/3:00 PM UTC

Note: Zoom link may change for this meeting or in the future. Email update will be provided if it is for this meeting.

C documents

The latest C2Y draft is N3886 2026/05/24: <https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3886.pdf>

C23 has been published ISO/IEC 9899, available for purchase. <https://www.iso.org/standard/82075.html>

IEEE 754 update

Damian: Discussion of the refining language of NaNs, divide by zero, and other extraordinary issues. There is also more work around correct rounding.

Jerome & group: We have looked at the objectionable use of “pole” to describe all instances of division by zero. We hope to improve our wording based on 60559.

WG14 update

Rajan: Finished Aug meeting. Next meeting 2026/11/16-21 in Brazil, mailing deadline a month before.

C++ liaison

Joshua: Next meeting coincides with the C meeting in Brazil. Early work on an “Annex F” for C++. SG22, the compatibility group, will also meet on Friday. Ordinarily, these WG meetings are offset by a week, permitting attendance at both, but in this case the site is available for just one week.

TS-4 and TS-5 revisions

None

News

Jerome: Travel necessitates a quick turn-around of these minutes. Comments to Jerome or CFP will be addressed after 7 Oct.

Carryover action items from last meeting

Fred: Add to open items list the issue in 3928 from Trevor regarding the awkward language in H.4.1 regarding whether or not functions need to be supported, based on what types are supported.

Carry over

Group: Look at the notion of “range of representable values”. Can we tighten up this usage?

Done

Action items from last meeting

Jerome & Rajan: Fix and submit proposal for improved wording in H.11.4.1.

Done n3962

Fred: Submit the nextafter/nexttoward proposal in email 3963.

Carry over

Ariel: Investigate the sign of fromfp(-0.0) across IBM compilers. Group invited to check other systems.

Ariel: Looked at nextafter(). Behavior nextafter(-0, +0).

Joshua: glib and llvm libc have fromfp, most other libraries don't. Not Intel nor others.

Jerome: Gives us freedom to do right thing. Jim Fred

carry over (enhanced by spontaneous discussion).

Fred & group: With input from the group, expand rationale and discussion of nextafter and zero cases

Carry over

Group: Interested members submit input on Fred's email 3964 re. nextafter() behavior around zero.

Carry over

Group: Submit thoughts about how SNaNs ought to behave in the printf/scanf functions.

Carry over, after some inevitable review of the issue and its ties to function calling conventions.

Jerome & Rajan: Send email to Paul (who is probably reading these minutes 8^) re. signgam as a POSIX issue.

Done cfp3998

Discussion of issues (as of 2026/09/28)

- **[cfp-interest 3962] Quantum of fromfp** - *Fred Tydeman - For reference (completed item)*

Carry over

- **[cfp-interest 3966] SNaN not raise invalid** - *Fred Tydeman*

Carry over

- **[cfp-interest 3967] Annex F - Introduction + SpecialCases** - *Damian McGuckin*

Damian: Not planning changes other tahn F.2.2 on “representable values”.

Rajan: There's no need to delay on "range of representable values" issue, which will take time to resolve.

Action

- **[cfp-interest 3969] Re: binary vs decimal sNaN initialization** - *Rajan Bhakta*

Rajan: Fred has completed and I've reviewed but we'll carry over.

- **[cfp-interest 3965] Model numbers and double-double** - *Fred Tydeman*

Carry over

Ariel & group: What are the model numbers? See 5.3.5.3.3 Characteristics of floating-point types <float.h> for the concepts behind "model floating-point numbers".

Ly: We are looking more at double-double and Nvidia's floatfloat. The context is llvm libc. For performance, we prefer double-double to float128 for high-precision needs.

- [\[cfp-interest 4006\] Action: range of representable values, revisited](#) *Jerome Coonen*

Rajan & Jerome: Table this proposal and focus on the discussion in the next item.

Closed

- [\[cfp-interest 4008\] Re: Action: range of representable numbers](#) *Jerome Coonen*

This discussion dominated the technical conversation. Given the note-takers active role in it, this summary is a best effort to capture key observations. Most important is the new action item below.

Jerome: Reviews the several uses of range of representable values (RORV) that pertain to floating point. Reiterates the interpretation that RORV refers to just the values of computer arithmetic

Joshua: Another interpretation, and the one taken by C++, says that this suggests that the value 0.1 is not a valid constant because it is not exactly representable. One section just cited is a constraint that requires that a compiler error be issued if the constant is not in the RORV.

Jerome: It's more subtle to talk (very precisely) about just which values may "project" onto a floating point number system. In my view, there is no need to do so. The point is that there is a mechanism to convert the text expressions of numeric values into computer numbers.

Joshua: Here's the C++ wording for constexpr evaluation: <https://eel.is/c++draft/expr.const.core>.

C++ is based on C, and it may have been unclear what was meant by RORV when the C++ concepts were elaborated. Now, infinities and NaNs are not considered part of the RORV.

Jerome: The whole point of 754 forty-five years ago was to avoid just such "errors" by providing infinity and NaN symbols to ensure that there would always be a representable value at the end of a computation.

Joshua & group: C and C++ have different rules, so we need to sort that out. The biggest computational issue may be whether extraordinary expressions like, "1.0/0.0" or "0.0/0.0" or "3.0e5000000000" or even "1.0e500 (overflows in double)" can be permitted in constants. In a 754 environment, there is a possible representable value. Possible questions are what is intended for the compiler and what is the best language to realize that?

Other issues

- <https://cfp-wiki.esi.com.au/pub/CFP/WebHome/c26h.pdf>

Rajan: As a note to readers, these items are referenced in discussion by their WG14 doc numbers or CFP email numbers, not by their index numbers in this list.

Adjournment

9:40 AM PDT

Action items to be carried over

Fred: Add to open items list the issue in 3928 from Trevor regarding the awkward language in H.4.1 regarding whether or not functions need to be supported, based on what types are supported.

Fred: Submit the nextafter/nexttoward proposal in email 3963.

Ariel & group: Investigate the sign of fromfp(-0.0) across IBM compilers and other systems.

Fred & group: With input from the group, expand rationale and discussion of nextafter and zero cases

Group: Interested members submit input on Fred's email 3964 re. nextafter() behavior around zero.

Group: Submit thoughts about how SNaNs ought to behave in the printf/scanf functions.

New action items

Damian & Rajan: Damian to send clean copies of Annex F Intro & Special Cases to Rajan, who will submit to WG14.

Jerome & Joshua & group: Research the use of "range of representable values", what its definition might be, how the term itself might be rephrased, and its ties to C++ and to related C concepts like model numbers.

Carryover discussion items

- **[cfp-interest 3962] Quantum of fromfp** - *Fred Tydeman - For reference (completed item)*
- **[cfp-interest 3966] SNaN not raise invalid** - *Fred Tydeman*
- **[cfp-interest 3969] Re: binary vs decimal sNaN initialization** - *Rajan Bhakta*
- **[cfp-interest 3965] Model numbers and double-double** - *Fred Tydeman*

Carry over

Signoff

Respectfully submitted.

-Jerome Coonen