Below are the NB comments for CD2 that CFP has reviewed, and suggested responses. (Some editorial comments seen as clear and uncontroversial are not listed.)

**Agree. Accept proposed change.**

- US-007 ed
- US-008 ed – covered by GB-009
- GB-009 ed
- US-015 ed
- US-016 ed
- US-017 ed
- GB-018 ed
- US-019 ed
- US-020 ed – covered by GB-009
- US-021 ed
- GB-023 ed
- US-024 ed
- US-034 ed
- GB-051 ed
- GB-052 ed
- US-056 ed – duplicate of GB-052
- US-057 ed – duplicate of GB-051
- US-070 ed
- GB-090 ed
- US-092 ed – defer to GB-090
- US-093 ed
- GB-103 ed – duplicate of US-104
- US-104 ed
- US-105 ed
- US-106 ed
- US-113 te
- US-114 te
- US-115 te – exact duplicate of US-113
- US-117 ed
- GB-118 ed
- GB-119 ed
- US-120 te
- US-121 ed
- US-122 ed – duplicate of GB-118
- US-123 ed – defer to GB-124
- GB-124 ed
- US-125 te
- GB-152 te
Disagree. Do not accept proposed change.

US-201 ed
Annex G does not specify the behavior of signaling NaNs. The proposed change is not consistent with support for signaling NaNs per recommended practice in F.2.1.

Generally agree. Modify/complete proposed change.

GB-005 te
The use of "return" in the proposed definition "return the negative of a number" seems off target, because here it is not in the context of an operation. We suggest "make the negative of a number".

The proposed note for the definition uses "sign bit" which refers to a bit representation which C generally does not specify for floating-point numbers. We suggest

**Note 1 to entry:** For a floating-point number (5.2.4.2.2), this changes the sign; for an integer, this is equivalent to subtracting from zero.

CA-022 te
In the proposed note, change "floating point" to "floating-point" (with a hyphen).
Pragmas without `STDC` are not conditional features in the sense used in the Standard. We suggest leaving “Any such pragma that is not recognized by the implementation is ignored” unchanged, and adding:

**Recommended practice**
Implementations are encouraged to diagnose unrecognized pragmas.

To clarify, we suggest re-punctuating the awkward statement in 7.12.1 #5 to: “If a floating result overflows and default rounding is in effect and the integer expression `math_errhandling & MATH_ERRNO` is nonzero, then the integer expression `errno` acquires the value `ERANGE`.

The `wchar.h` summary is also missing `wcstof`, `wcstod` and `wcstol`. Add these to the summary too (not conditional on DFP support).

There’s a typo in the Proposed change. It should be: Change “isinfinite” to “isfinite”.

The proposed change would erroneously invalidate optimizations like code motion and common subexpression elimination which can be safely done between function calls. The issue seems to be about whether “floating-point exceptions need not be precise” implies the result value of the exceptional operation need not be determinant, which is not the intention. To clarify F.9.1 #3, we suggest changing “floating-point exceptions need not be precise” to “the side effects due to floating-point exceptions need not be precise”.