Proposal for C26
WG14 Nnnnn

Title:	**DRAFT** Annex G and F updates - Introduction
Author:	Damian McGuckin
Date:	2024-08-08
Proposal Category:	Editorial
Reference:	N3301

The changes listed here fall into two categories. Some make details of the various paragraphs more consistent. Others enhance readability or clarity, and in a very few cases, fill in gaps. Both Annex F and Annex G should now be consistent both internally and with each other especially in the use of commas, terms, sentence structure and ways of expressing the same sort of sentence in two different places,

Comments within this document including **ASIDE:** comments, and various editorial annotations used to highlight changes, will disappear from this document before submission.

The mention of whether individual functions are odd or even is used for different reasons in Annex F and Annex G, and hence it is not crucial that the explanation is consistent. In the former, it is used in the context of ensuring that rounded results are identical. In the latter, it is used as part of the specifications of the special cases. That said, nothing is mentioned ebout ensuring identical rounding behavior of odd or even functions in Annex G. **It this being too picky**?

Consistency Fixes in F.10.1 General and G.4.1 General

F.10.1 Paragraph 1 - REPLACE

This clause contains specifications for `<math.h>` and `<tgmath.h>` functions that are particularly suited to ISO/IEC 60559 implementations.

G.4.1 Paragraph 1 - REPLACE

This clause contains specifications for `<complex.h>` functions that are particularly suited to ISO/IEC 60559 implementations.

F.10.1 Paragraph 2 - REPLACE

The specifications in the following subclauses augment the definitions in `<math.h>` and `<tgmath.h>`. Where specifications appear for a function *f* which is a member of a family of functions, those specifications apply to all member functions of that family. even though only the (principal) member function is shown.

G.4.1 Paragraph 1+ - ADD - Single Paragraph

The specifications in the following subclauses augment the definitions in `<complex.h>`. Where specifications appear for a function *f* which is a member of a family of functions, those specifications apply to all member functions of that family. even though only the (principal) member function is shown.

F.10.1 Paragraph 3 - REPLACE

Unless otherwise specified, where the $\pm$ symbol occurs within both a function argument *f* and its result, it implies two cases: one with each $\pm$ symbol replaced with a plus (+) symbol in both argument and result, and the other with each $\pm$ symbol replaced with a minus (–) symbol in both argument and result.

Each of the functions *f* seen in this annex generally takes a single floating point number *x* as an argument and returns a single floating point number result.

G.4.1 Paragraph 1++ - ADD Multiple Paragraphs

Unless otherwise specified, where the $\pm$ symbol occurs within both a function argument *f* and its result, it implies two cases: one with each $\pm$ symbol replaced with a plus (+) symbol in both argument and result, and the other with each $\pm$ symbol replaced with a minus (–) symbol in both argument and result. Where the $\pm$ symbol occurs only within an argument and not in the function result, the result is independent of that sign. Where the $\pm$ symbol occurs only within a function result and not in an argument, the sign of that part of the result is unspecified.

A complex number *z* as used in the following subclauses can be written (6.2.5#17) as $z = x + i\,y$, i.e. as the sum of its real ($x = \mathbf{Re}(z)$) and imaginary ($y = \mathbf{Im}(z)$) components. Similarly, the complex conjugate of *z*, known also as $\overline{z}$ or conj(*z*), can be written as $\overline{z} = x - i\,y$ (7.3.9.4).

Each of the functions *f* seen in this annex generally takes a single complex number *z* as an argument. When*f* returns a complex number result, it is generally written as the sum of its real and imaginary components.

F.10.1 Paragraph 14 - REWORD for clarity

Functions with an argument of NaN return a result of NaN and raise no floating-point exception, except where explicitly stated otherwise. Where a NaN appears in a functions's argument(s) or result, even though its representation might have a sign bit, that bit has no interpretation.

G.4.1. Paragraph 4+ - ADD to match preceding paragraph

Functions with an argument of $\mathrm{NaN} + i\,\mathrm{NaN}$ return a result of $\mathrm{NaN} + i\,\mathrm{NaN}$ and raise no floating-point exception, except where explicitly stated otherwise. Where a NaN appears in the real or imaginary component of a function's argument(s) or result, even though its representation might have a sign bit, that bit has no interpretation.

DELETE the last item of G.4.2.1, G.6.3.1-6, G.6.4.1-2, and G.6.5.2 as this new paragraph in the preliminary section of "G.4.1 - General" now makes them totally redundant.

Miscellaneous Fixes in F.10.1 General and G.4.1 General

F.10.1 Paragraph 14+ - was Paragraph 3 - MOVED + English Fixes + Font Fix

For each function f of a single argument in `<math.h>` whose mathematical analog is an even function, then $f(-x) = f(x)$ for all the ISO/IEC 60559 rounding modes for all x in the domain of f. Likewise, for each function f of a single argument in `<math.h>` whose mathematical analog is an odd function, then $f(-x) = -f(x)$ for the roundTiesToEven, roundTiesToAway, and roundTowardZero ISO/IEC 60559 rounding modes for all x in the domain of f. The **`atan2`** and **`atan2pi`** functions are odd in their first ar gument.

ASIDE: Annex F already uses the concept of an *analog* but introduced the similar term *counterpart* for a single use in this one place. So, *mathematical counterpart* was replaced with *mathematical analog*.

F.10.1 Paragraph 15 - was Paragraph 2 - MOVED

The macro **`HUGE_VAL`** and its **`float`** and **`long double`** analogs, **`HUGE_VALF`** and **`HUGE_VALL`**, expand to expressions whose values are positive infinities.

F.10.1 Paragraph 15+ - ADD - Special Cases need an introduction

The behavior of the special cases of the functions of Table F.3, including their treatment of any "invalid" and "divide-by-zero" floating-point exception, is specified in the following subclauses. As noted earlier, where the specifications for the principal function of a family of functions is seen, they apply to all member functions within that family.

ASIDE: The raising of floating-point exceptions associated with range errors is missing. Is that intentional?

G.4.1 Paragraph 6 - table text now consistent with paragraph following

Each of the functions **`cabs`** and **`carg`** (7.3) is specified by a formula in terms of a real function (whose special cases were specified earlier in Annex F):

G.4.1 Paragraph 7 - table text now consistent with paragraph preceding

Each of the functions **`casin`**, **`catan`**, **`ccos`**, **`csin`**, and **`ctan`** (7.3) is specified by a formula in terms of some other complex function (whose special cases are specified later in this annex):

G.4.1 Paragraph 8 - Reword for clarity - Uses Mathematical domain

The behavior of the special cases of the functions **`cacos`**, **`cacosh`**, **`casinh`**, **`catanh`**, **`ccosh`**, **`csinh`**, **`ctanh`**, **`cexp`**, **`clog`**, and **`csqrt`** (7.3), including their treatment of any "invalid" and "divide-by-zero" floating-point exception, is specified in the following subclauses. Each of these functions is an instance of a complex function f which can tak e an argument $z = x + i\,y$ in either the upper complex half-plane, i.e. where $|x| < \infty$ with $0 < y < \infty$ or y is a +0, or an ar gument $z = \overline{z} = x - i\,y$ in the lower complex half-plane. For each such f, the conjug ate of the evaluation of f for some ar gument z is identical to the evaluation of that same f for an argument of $\overline{z}$, the conjugate of that same z, or more compactly $\mathrm{conj}(f(z)) = f(\mathrm{conj}(z))$. This means that any specifications for the upper complex half-plane (i.e. where **Im***(z)* is always positively signed) imply specifications for the lower complex half-plane (i.e. where **Im***(z)* is always negatively signed). Additionally, if such a function f is either even or odd, i.e. it satisfies either $f(-z) = f(z)$ or $f(-z) = -f(z)$ respectively, then specifications for the first quadrant imply specifications for the other three quadrants. As noted earlier, where the specifications for the principal function of a family of functions is seen, they apply to all member functions within that family.

ASIDE: Ensure that the explanation about odd and even functions reads much the same as that in Annex F Paragraph 14 above. In Annex G, these words are used only in a special case context while in Annex F, they hold beyond just special cases and also refer to rounding. **ODD**! **Do we need to cross reference F.10.1#14**?

G.4.1 Paragraph 9 - DELETE - Surplus to Requirements