Proposal for C26
WG14 Nnnnn

Title:	**DRAFT** Annex F.10.1 and G.4.1 updates
Author:	Damian McGuckin
Date:	2024-08-08
Proposal Category:	Editorial
Reference:	N3301

The first page of changes, most of which are in the body of the document, should probably be a **separate** document going to WG14. It is included at this stage simply because it is very closely related to Annex G.

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 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, A paragraph may be **REWORD**ed and/or **ENHANCED**ed for reasons of grammar, clarity, or consistency, or some mix thereof.

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 about ensuring identical rounding behavior of odd or even functions in Annex G. **ODD**! **Do we need to cross reference F.10.1#14**?

With the new paragraph **G.4.1#8+**, **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.

The term *real values* appears only once within the standard and is never defined. As a way arund that, the word *mathematical* is used to qualify *real values* in the hope that the concept of mathematical real value needs no definition.

The term *complex number* is never defined even though it appears in the contents and index and once (wrongly) in the body of the standard where it is referring to an object of complex type.

The term *complex value* is used in **7.3.9.3#3** and **G.2#1**. But it too is never defined. Note that **7.3.9.3#3** wants to return a complex object whose value is the mathematical equivalent of $x + i\,y$. But it never mentions what i is, nor has i even been mentioned anywhere in the body of the standard to that point.

As amended by CFP 3155, **6.2.5#17** uses both *complex number* and *complex values*.

> Each complex type has the same representation and alignment requirements as an array type containing exactly two elements of the corresponding real type; the first element is equal to the real part, and the second element to the imaginary part, of the complex number. In this document complex values are sometimes written in the form $x + i\,y$, where x and y are the values of the real and imaginary parts. The form $x - i\,y$ represents $x + i\,(-y)$.

The definition of i is not introduced until 150 pages later, and even then, in a footnote. And apart from one instance in **7.3.9.3#3**, it is not until Annex G that a complex number appears in that form again.

The following two changes, along with the new clause for **7.3.1#5** is a suggested fix for all of these issues.

6.2.5#17 Types - REWORD - is this tight enough

Each complex type has the same representation and alignment requirements as an array type containing exactly two elements of the corresponding real type. The real part of an object of complex type would be the first element of that array type, while the imaginary part of an object of complex type would be the second element of that array type.

7.3.1# Introduction - NEW - notational explanation

An object of complex type is often subsequently referenced in this document as z. Such a z may sometimes be written in terms of its real and imaginary parts as $x + i\,y$ where i is the imaginary unit[227)] and x is the real part of z, or mathematically $x = \mathbf{Re}(z)$, and y is the imaginary part of z, or mathematically $y = \mathbf{Im}(z)$. And conversely, the expression $x - i\,y$ represents $x + i\,(-y)$, (by definition the mathematical conjugate of $x + i\,y$).

G.2 Conventions - REWORD - the standard does not define the concept of a complex value

An object of complex type is regarded as an infinity if either of its real and imaginary parts is also an infinity, even if its other part is a quiet NaN. An object of complex type is regarded as finite if both its real and imaginary parts are also finite, i.e. neither infinite nor a NaN. An object of complex type is regarded as a zero if both its real and imaginary parts are also a zero.

Other Miscellaneous Fixes

In **6.3.2.7#2**, change *complex value* to *value of complex type* to agree with the earlier part of that sentence.

In **7.2.27#10**, change *complex floating type* to *complex type* because the former concept is meaningless.

F.10.1#1 - REPLACE with tighter English - last sentence came from #15

This clause contains specifications for `<math.h>` and `<tgmath.h>` functions that are particularly suited to ISO/IEC 60559 implementations. 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. Unless otherwise noted, where the ± symbol occurs within both a function argument *f* and its result, it implies two cases: one with each ± symbol replaced with a plus (+) symbol in both argument and result, and the other with each ± symbol replaced with a minus (–) symbol in both argument and result.

G.4.1#1 - REPLACE with tighter English - enhance the explanation

This clause contains specifications for `<complex.h>` functions that are particularly suited to ISO/IEC 60559 implementations. 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. Unless otherwise noted, where the ± symbol occurs within both a function argument *f* and its result, it implies two cases: one with each ± symbol replaced with a plus (+) symbol in both argument and result, and the other with each ± symbol replaced with a minus (–) symbol in both argument and result. Where the ± symbol occurs only within an argument and not in the function result, the result is independent of that sign. Where the ± symbol occurs only within a function result and not in an argument, the sign of that part of the result is unspecified.

F.10.1#3 - REPLACED - the original has been moved to #15

Generally, each function *f* in this annex takes at least one argument of real floating type and returns a result of that same type.

G.4.1#3 - ENHANCED - last sentence moved to #4

Generally, each function *f* in this annex takes an argument of complex type and returns a result of that same type. Since both the real and imaginary part of both the argument and result are both mathematically real values, functions returning a result of complex type from an argument of complex type may be regarded (in a loose mathematical sense) as computing real values from real values.

F.10.1#4 - REWORDED - 1st and 2nd sentence have been moved to #15+

Except as noted, real infinities, NaNs, signed zeros, subnormals, and, provided the state of the **`FENV_ACCESS`** pragma is "on", the floating-point status and exception flags, are treated by these functions in a manner consistent with ISO/IEC 60559 operations. In this annex, a NaN, unless explicitly qualified, refers to a quiet NaN and the behavior of signaling NaNs is implementation-defined (F.2.2#2).

G.4.1#4 - ENHANCED - prefix with the last sentence of #3 to match F.10.1#4

Except as noted, real infinities[446)], NaNs, signed zeros, subnormals, and the floating-point status and exception flags are treated by these functions in a manner consistent with that by real functions seen in F.10. In this annex, a NaN refers to a quiet NaN and written *real* is the *x* part of *z*, or mathematically = **Re**(*z*) and *imaginary* is the *y* part of *z*, or mathematically = **Im**(*z*), may appear subsequently and the behavior of signaling NaNs is implementation-defined.

F.10.1#14 - REWORDED and ENHANCED

Unless explicitly stated otherwise, functions with an argument of NaN return a result of NaN and raise no floating-point exception. Should the representation of a NaN appearing in a functions's argument(s) or result have a sign bit, that bit has no interpretation.

G.4.1#8+ - MERGER of last item of G.4.2.1+G.6.3.1-6+G.6.4.1-2+G.6.5.2

Unless explicitly stated otherwise, functions with an argument of NaN + *i* NaN return a result of NaN + *i* NaN and raise no floating-point exception. Should the representation of a NaN appearing in the real or imaginary part of a functions's argument(s) or result have a sign bit, that bit has no interpretation.

F.10.1#15 - MOVED - was Paragraph 3 with fixed fonts + improved English

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#15+ - MIXTURE - first sentence came from #4 and rest from #G4.1#8

In Table F.3 are the functions of `<math.h>` whose special cases are covered directly by ISO/IEC 60559. Their behavior, 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 are 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#6 - table text now consistent with paragraph following

Each of the functions **`cabs`** and **`carg`** (7.3) which takes an argument $z = x + i\,y$ of complex type is specified by a formula in terms of a real function (whose special cases were specified earlier in Annex F):

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

Each of the functions **`casin`**, **`catan`**, **`ccos`**, **`csin`**, and **`ctan`** (7.3) which takes an argument z of complex type is specified by a formula in terms of some other complex function (whose special cases will be specified later in this annex):

G.4.1#8 - REWORDED and ENHANCED - 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 $\overline{z} = x - i\,y$ in the lower complex half-plane where $\overline{z}$ is called the complex conjugate of z (7.3.9.4). For each such f, the conjug ate of the evaluation of f for some argument z is identical to the evaluation of that same f for an ar gument of $\overline{z}$, the conjugate of that same z, or more compactly $\mathrm{conj}(f(z)) = f(\mathrm{conj}(z))$. This means that specifications for the upper comple x 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 $f(-z) = -f(z)$ respecti vely, 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 are seen, they apply to all member functions within that family.

G.4.1#9 - DELETE - no longer necessary