Template for comments and secretariat observations

Date: 11 Sep 2013	Document: **WG 14 N1754**	Project: 18661

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
GB				ge	The new interfaces defined in this document should generally be considered experimental until there is more implementation and user experience with them. Although this document is generally phrased in terms of amendments to the text of C11, such experience should be considered before any of those amendments are included in a future revision of the C standard, and consideration given at the time of such a revision to being selective in which changes are included, based on such experience.		Agreed – no change requested
GB	Page v line 31	Introduction		ed	The description of operations in IEC 60559:1989 starts a new sentence in parentheses without any punctuation terminating what came before ("system (It ... operations.)").	Say “system; also conversions ...”.	Agreed
GB	Page vi line 4	Introduction		ed	This line uses “--” (two hyphens) as a dash.	Change “--” to an actual dash in both places.	Agreed
GB	Page vi line 21	Introduction		ed	“defines it model” has a typo in it.	Change to “defines its model”.	Agreed
GB	Page 1 line 9	1		ed	The scope description calls out decimal floating point as not covered but without mentioning other uncovered features specifically. Given the increased emphasis in ISO/IEC/IEEE 60559:2011 on reproducible results, in particular, it seems desirable to emphasise the non-inclusion of certain other features.	After “decimal floating-point arithmetic”, add “, reproducible results, order of evaluation of expressions beyond the definitions in C11, or control of optimizations”.	For the features mentioned in the proposed change for this comment, some parts are supported. For example, much of the support for reproducible results is present, and FP pragmas provide some control of optimization. Suggest no change.
GB	Page 2 lines 11-24	5.1		te	Although, formally, the effects of this document's changes to C11 are irrelevant to implementations not defining __STDC_IEC_60559_BFP__ and implementing Annex F, it still seems appropriate for such changes to avoid placing undue burdens on implementations not defining __STDC_IEC_60559_BFP__ and implementing Annex F, if incorporated verbatim in a future revision	Replace the proposed change to C11 by: Append to the third sentence of 4#6: A conforming freestanding implementation that defines __STDC_IEC_60559_BFP__ shall also provide all the library facilities specified in the standard headers <fenv.h> and <math.h> and the numeric conversion functions (7.22.1) of the standard header <stdlib.h>.	Agree in principle, except for last sentence “Furthermore …”. Footnote 3 in C11 explains how strictly conforming programs can use conditional features. Suggest to append after third sentence: “The strictly conforming programs that shall

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
					of the C standard. With that in mind, requiring additional library support (especially the full contents of <math.h>) from a conforming freestanding implementation not claiming to support Annex F seems inappropriate. Instead, the additions to 4#6 should be phrased in a form that only requires any additional library support when __STDC_IEC_60559_BFP__ is defined. Furthermore, this cannot be phrased in terms of strictly conforming programs, because such programs cannot depend on implementation-defined behavior such as whether __STDC_IEC_60559_BFP__ is defined.		be accepted by a conforming freestanding implementation that defines **__STDC_IEC_60559_BFP__** may also use features in the contents of the standard headers **<fenv.h>** and **<math.h>** and the numeric conversion functions (7.22.1) of the standard header **<stdlib.h>**."
GB	Page 2 lines 23-24	5.1		te	This text does not specify whether including <stdlib.h> in a freestanding implementation may define or reserve the other identifiers defined or reserved when <stdlib.h> is included in a hosted implementation.	At the end of the new text, add "All identifiers that are defined or reserved when <stdlib.h> is included in a hosted implementation are reserved when it is included in a freestanding implementation.".	Agreed
GB	Page 2 lines 25-36	5.2		te	Removing __STDC_IEC_559__, if integrated in a future revision of the C standard, would be an incompatible quiet change for applications using this macro (which would then exhibit undefined behavior because it would be a macro in the reserved namespace not specified by the standard). It should be obsoleted like __STDC_IEC_559_COMPLEX__, not removed.	Change "Note that an implementation may continue to define __STDC_IEC_559__, so that current programs that use __STDC_IEC_559__ may remain valid under the changes in this Part of Technical Specification 18661." to "The macro __STDC_IEC_559__ is retained as obsolete, for compatibility with existing applications.". Change "In 6.10.8.3#1, replace … with:" to "In 6.10.8.3#1, before the __STDC_IEC_559__ item, insert the item:". After line 36, insert "In 6.10.8.3#1, append to the __STDC_IEC_559__ item: Use of this macro is an obsolescent feature.".	Agreed
GB	Page 3 line 5	5.2		ed	"__STDC_IEC_559_COMPLEX" is a typo.	Change "__STDC_IEC_559_COMPLEX" to "__STDC_IEC_559_COMPLEX__".	Agreed
GB	Page 3 lines 8-10	5.3		te	__STDC_WANT_IEC_18661_EXT1__ should be handled consistently with other __STDC_WANT_* macros. The simpler practice in this document may	Either insert amendments to the three other documents listed to follow the form of wording listed here, or amend the wording here to follow C11 Annex K.	Current approach in TS follows direction given by WG 14. Suggest no change.

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
					be better than the more complicated practice in C11 Annex K and TR 24731-2 and ISO 24747, but either this document should follow those documents, or those documents should be amended to follow the simpler practice.		
GB	Page 4 lines 19-20	7.1		te	No method is provided for an application to determine which choice has been made for the long double type. (This is also an issue with C11 as it stands and so could also be addressed through a TC.)	At the end of subclause 7.1 in this document, insert the following: In 5.2.4.2.2, insert a new paragraph after paragraph 10: Whether a type matches an IEC 60559 type is characterized by the implementation-defined values of FLT_IS_IEC_60559, DBL_IS_IEC_60559, and LDBL_IS_IEC_60559: 0 type does not match an IEC 60559 format 1 type's values and operations are those of an IEC 60559 basic, interchange or extended type	New feature request? Suggested change wouldn't fully identify format of long double. Suggest no change.
GB	Page 4 lines 14-32	7.1		te	The C11 definition of FLT_ROUNDS is inadequate in that it refers to floating-point addition but does not say addition of what type. If long double is not an IEC 60559 type, as still permitted by this specification, it may not fully support all rounding modes even though they are supported by other types. (This is also an issue with C11 as it stands and so could also be addressed through a TC.)	At start of changes to C11 on this page, insert: In 5.2.4.2.2#8, insert "for type float" after "floating-point addition". Before the changes to F.2 recommended practice, insert: At the end of F.2#1, insert "The value of FLT_ROUNDS applies to all IEC 60559 types supported by the implementation, but may not apply to non-IEC 60559 types.".	Implementations with inconsistent rounding among types can define FLT_ROUNDS to be -1. Better methods exist for determining rounding directions. Suggest no change.
GB	Page 5 line 23	7.2		te	"6.2.6.1 not withstanding" is erroneous as nothing here conflicts with 6.2.6.1.	Remove ", 6.2.6.1 not withstanding".	6.2.6.1#8 says "Where an operator is applied to a value that has more than one object representation, which object representation is used shall not affect the value of the result" which is not true for iscanonical. It also says "Where a value is stored in an object using a type that has more than one object representation for that value, it is unspecified which representation is used, …", but most operations are specified (in

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
							IEC 60559) to return a canonical representation. Suggest no change.
GB	Page 6 lines 5-7	8		te	The description of operation binding fails to make sufficiently clear that floating-point exceptions are not exceptional conditions within the meaning of 6.5#5. (This is also an issue with C11 as it stands and so could also be addressed through a TC.)	At the start of the changes to C11 in this subclause, insert the following: Append to 6.5#5: For implementations defining __STDC_IEC_60559_BFP__, this does not apply to exceptional conditions where the behavior (such as raising a floating-point exception and returning a NaN) is defined by Annex F, directly or by reference to IEC 60559.	Terms “exceptional condition” and “floating-point exception” are used consistently to refer to different things (reviewed and updated for C11). Suggest no change.
GB	Page 12 lines 21-25	10.2		te	The specification of the new strfrom* functions seems unclear about the format string contents. Taken literally, the string contains optionally ".", ".*" or ".<decimal-integer>", followed by one of the given letters, but not a leading "%". However the equivalence to snprintf rather suggests a leading "%" should be included. And the absence of any way to provide an "int" field for ".*" precision suggests that case should not be allowed here.	Insert “an initial % character,” after “contains only”. Insert “, not .*,” after “optional precision”.	Agreed
GB	Page 12 lines 28-29	10.2		te	The error condition given is "if an encoding error occurred", but that condition doesn't seem applicable to these numeric conversions.	Remove the word “encoding”.	Agree with comment. Suggest to remove "if an encoding error occurred" and change “Thus, the null-terminated output has been completely written if and only if the returned value is nonnegative and less than **n” to** “Thus, the null-terminated output has been completely written if and only if the returned value is less than n”
GB	Page 12 lines 13-30	10.2		te	A deficiency in the printf family functions is that they have undefined behavior if the number of characters that are written, or would be written, exceeds INT_MAX, and no good way for applications to detect that condition in advance and avoid the	Change “int” to “size_t” for all three functions. In the comparison to snprintf, insert “the return type is size_t and” between “except” and “the format string”. Change “a negative value” to “SIZE_MAX”. Change “error occurred” to “error occurred or the number of	Defer to WG14

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
					undefined behavior. (POSIX adds an error condition with errno set to EOVERFLOW for this case.) The strfrom* functions are not printf-family functions and so need not follow the deficiencies of such functions; size_t is now the standard type for C objects storing the size of something in memory, so should be used for the return type here.	characters required would have exceeded SIZE_MAX". Remove "nonnegative and".	
GB	Page 12 line 31	10		te	C11 7.22.1.3#3 specifies input strings for strtod, strtof and strtold representing infinities as "INF or INFINITY, ignoring case"; 7.29.4.1.1#3 says "INF or INFINITY, or any other wide string equivalent except for case". The IEC 60559:2011 requirement, however, refers to '"inf" and "infinity" (regardless of case)'. But in a Turkish locale, these descriptions are not equivalent; the lowercase version of 'I' is a dotless 'i', and the uppercase version of 'i' is a dotted 'I'. The requirements should include that all strings matching the description in IEC 60559:2011 (as interpreted according to the current locale) are properly interpreted, in addition to any other locale-specific forms the implementation may accept.	Page 12, after line 30, insert a new subclause 10.3 Conversions of character sequences representing infinities and NaNs: The following changes to C11 ensure that character sequences for infinities and NaNs are interpreted as required in IEC 60559:2011 even in locales where the mapping between uppercase and lowercase letters is not the same as in the C locale. Changes to C11: In 7.22.1.3#3, change "INF or INFINITY, ignoring case" to "INF, inf, INFINITY or infinity, ignoring case". Change "NAN or NAN(n-char-sequence_opt), ignoring case in the NAN part" to "NAN, nan, NAN(n-char-sequence_opt) or nan(n-char-sequence_opt), ignoring case in the NAN or nan part". In 7.22.1.3#4, change "INF or INFINITY" to "INF, inf, INFINITY or infinity". Change "NAN or NAN(n-char-sequence_opt)" to "NAN, nan, NAN(n-char-sequence_opt) or nan(n-char-sequence_opt). In 7.29.4.1.1#3, change "INF or INFINITY" to "INF, inf, INFINITY or infinity". Change "NAN or NAN(n-wchar-sequence_opt)" to "NAN, nan, NAN(n-wchar-sequence_opt) or nan(n-wchar-sequence_opt)". Change "NAN part" to "NAN or nan part". In 7.29.4.1.1#4, change "INF or INFINITY" to "INF, inf, INFINITY or infinity". Change "NAN or NAN(n-wchar-sequence_opt)" to "NAN, nan, NAN(n-wchar-sequence_opt) or nan(n-wchar-sequence_opt)".	The IEEE 754r committee strove to be consistent with C99 but decided that different character sets were out of its scope (though this decision is not stated in the floating-point standard). The specification in question is consistent with IEC 60559, considering its scope. Suggest no change.

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
GB	Page 12 lines 32-37	11		te	The description of the dynamic floating-point environment in C11, as amended, fails to make sufficiently clear what is or is not an object (C11 footnote 205 is not normative, and so cannot be used to that effect). (This is also an issue with C11 as it stands, and so could also be addressed through a TC.)	Line 37, at end of paragraph insert "The normative text in C11 describes various properties of the (dynamic) floating-point environment, but does not state what parts of it might be an object or objects; to clarify this regarding exceptions being raised more than once in an expression, relevant text is moved out of a footnote.". At the start of the changes to C11, insert: Move the contents of footnote 205 (C11 subclause 7.6) to the end of 5.1.2.3#2.	Agree in principle. As with other issues with the current C11, there's the question of whether it should be addressed though a TC or in the TS. Defer to WG14.
GB	Page 14 lines 3-9	11		ed	"FP_DFL_ENV" is a typo.	Change "FP_DFL_ENV" to "FE_DFL_ENV" in both places.	Agreed
GB	Page 15 lines 1-28	11		te	The interaction of constant rounding modes with inline functions should be explicitly specified.	On line 24, change "functions other than" to "functions, including inline functions, other than". (If some other semantics are desired, consideration would need to be given to the handling of floating-point constants in inline functions; the values for such constants can affect whether there is a constraint violation, which would cause its own problems if inline functions were to be affected by constant rounding modes from their callers.)	The proposed change is not needed, assuming it's given that inline functions must behave the same whether they're inlined or not? Suggest no change
GB	Page 15 lines 12-18	11		te	The new 7.6.1a paragraph 4 says "the mode specified by the dynamic floating-point environment, which is the dynamic rounding mode that was established either at thread creation or by a call to fesetround, fesetenv, or feupdateenv". But the new function fesetmode can also have the effect of changing the dynamic rounding mode.	Insert "fesetmode, " before "fesetround".	Agreed
GB	Page 15	11	Table 2	te	The definition of functions affected by constant rounding modes should be more explicit that the float and long double versions of functions listed are also included.	Line 27, at end insert: "An entry for a function in Table 2 includes the corresponding functions for all supported types (for example, acosf and acosl as well as acos).".	Agree in principle. Suggest to add the following after Table 2: "Each <math.h> functon listed in Table 2 indicates the family of functions of all supported types (for example, acosf and acosl as well as acos)."
GB	Page 15	11	Table 2	te	The table of function groups affected by constant	After "cbrt", insert ", hypot".	Agreed

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
					rounding modes should include hypot.		
GB	Page 15	11	Table 2	te	For implementations of ISO 24747, the additional functions defined there should, by analogy with the C11 <math.h> functions, also be affected by constant rounding modes. (This document should consider interactions with all relevant ISO C extensions.)	At end of table, insert an entry for <math.h>, "All functions from ISO/IEC 24747, for implementations of that International Standard".	Defer to WG14
GB	Page 15 lines 19-21	11		te	The reference to "all floating-point operators and invocations of functions indicated in Table 2 below, for which macro replacement has not been suppressed" isn't clear about implicit conversions, which are neither operators nor function invocations.	After "operators", insert ", implicit conversions (including the conversion of a value represented in a format wider than its semantic type to its semantic type, as done by classification macros),".	Agreed
GB	Page 15 lines 19-23	11		te	The effects of the dynamic rounding mode on many of the <math.h> functions are implementation-defined (C11 F.10#10). It should be made clear here that for such functions, constant rounding modes place no more requirements on the functions than setting the dynamic rounding mode.	After "established by a call to fesetround", insert "; where the effect of the dynamic rounding mode on a function is implementation-defined, the same implementation definition applies to the constant mode as to when that mode is established by a call to fesetround".	7.6.1a#4 already has: "shall be evaluated according to the specified constant rounding mode (as though no constant mode was specified and the corresponding dynamic rounding mode had been established by a call to **fesetround**)". And F.10 says "Whether the functions honor the rounding direction mode is implementation-defined, unless explicitly specified otherwise." Suggest no change.
GB	Page 15 lines 25-26	11		te	Referring to "only the dynamic mode" isn't accurate since the called function might be in the scope of its own constant rounding mode.	After "only the dynamic mode", insert " and any constant mode in scope for the definition of the called function". Apply this also to the new text from the following comment if both are accepted.	This is about "invocations of functions" (not definitions of functions) in the scope of a constant rounding mode. Suggest no change.
GB	Page 15 lines 23-26	11		te	The reference to "Invocations of functions for which macro replacement has been suppressed" doesn't strictly cover uses of a function name that are not invocations (calls) of that function; in particular, taking the address of a function for a call somewhere	After "only the dynamic mode.", insert "Where the address of a function is taken in a context where macro replacement has been suppressed, calls using the resulting function pointer are affected by only the dynamic mode.". (Strictly this implies the previous	If "the address of a function is taken in a context where macro replacement has been suppressed" and the function is called "using the resulting

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
					else. It should be made clear that the resulting pointer is not bound to a constant rounding mode.	sentence, since all function calls in C are through function pointers, but it seems less confusing to make both statements explicitly.)	function pointer", then that amounts to an invocation of the function for which macro replacement has been suppressed, and is covered by the current specification. Suggest no change.
GB	Page 19 line 12	12		ed	The macro is called "isunordered", not "unordered".	Change "The unordered macro" to "The isunordered macro".	Agreed
GB	Page 23 lines 14-17	13		te	The width macros should not be required to have the same type as the type whose width they describe.	At the start of the changes to C11, insert: In 5.2.4.2.1#1, insert "the *_WIDTH macros, " before CHAR_BIT and MB_LEN_MAX". Before the additions of macros to <stdint.h>, insers: In 7.20.2#2, insert ", except for the *_WIDTH macros, " before "this expression shall have the same type".	Agreed, but suggest using width-of-type macros instead of *_WIDTH macros
GB	Page 23 lines 42-45	13		te	The width macros for intmax_t and uintmax_t should go in <stdint.h> (7.20.2.5), not <limits.h>.	Between lines 41 and 42, insert: In 7.20.2.5, insert the following bullets, each after the corresponding bullet for the same type:	Agreed
GB	Page 24 lines 1-11	13		te	For consistency, width macros should be provided for all the types for which limit macros are provided in <stdint.h>, including the exact-width types (given that limit macros are defined for them).	At start of page, insert: In 7.20.2.1, append – width of exact-width signed integer types INTN_WIDTH N – width of exact-width unsigned integer types UINTN_WIDTH N After line 11, insert: In 7.20.2.4, append – width of pointer-holding signed integer type INTPTR_WIDTH 16 – width of pointer-holding unsigned integer type UINTPTR_WIDTH 16 In 7.20.3#2, insert the following bullets, each after the	Agreed

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
						corresponding bullet for the same type: – width of ptrdiff_t PTRDIFF_WIDTH 17 – width of sig_atomic_t SIG_ATOMIC_WIDTH 8 – width of size_t SIZE_WIDTH 16 – width of wchar_t WCHAR_WIDTH 8 – width of wint_t WINT_WIDTH 16	
GB	Page 27 lines 1-12	14.1.2		te	The new FP_INT_* macros should expand to integer constant expressions of type int and distinct values. (Once this is specified, generic C11 requirements mean they must also be usable in #if without that needing mentioning separately.)	At end of line 12, append: "They expand to integer constant expressions with type int and distinct values.".	Agree in principle. Suggest: "They expand to integer constant expressions with distinct values suitable for use as the second argument to the **fromfp**, **ufromfp**, **fromfpx**, and **ufromfpx** functions."
GB	Page 27 lines 33-34	14.1.2		te	"outside the range of integers of the specified width" assumes there is a single range for integer types of that width, which is not otherwise required by ISO C	Append "for any integer representation supported by the implementation" to that wording, with a footnote "For signed types, 6.2.6.2 permits three representations, which differ in whether a value of -(2^M) can be represented.".	Agree in principle. Suggest: "outside the range of any supported integer type of the specified width", with footnote "For signed types, 6.2.6.2 permits three representations, which differ in whether a value of $-(2^M)$, where M is the number of value bits, can be represented."
GB	Page 28 lines 26-33	14.1.2		te	The previous issue applies twice here as well.	Append the same text (without the footnote) in both places.	Agreed

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
GB	Page 30 lines 1-28	14.3		te	The generic specification of fmaxmag and fminmag does not specify the result when the arguments have equal magnitudes but opposite signs. This should follow the specification of minNumMag / maxNumMag to which these functions are, for Annex F implementations, bound.	Line 15, append: "If the arguments have equal magnitudes, the fmaxmag functions return the numeric value of their argument of maximum value.". Line 28, append: "If the arguments have equal magnitudes, the fminmag functions return the numeric value of their argument of minimum value.". Page 31, line 11 insert "fabs(x) == fabs(y) ? fmax(x, y) : " after "r = ".	Agree in principle. Suggest to change Definition to: "The **fmaxmag** functions determine the value of their argument whose magnitude is the maximum of the magnitudes of the arguments: the value of **x** if **|x| > |y|**, **y** if **|x| < |y|**, and **fmax(x, y)** otherwise." and to change sample implementation on Page 31 to: "**{** **double ax, ay, r;** **ax = fabs(x);** **ay = fabs(y);** **if (isgreater(ax, ay)) (void)canonicalize(&r, &x);** **else if (isgreater(ay, ax)) (void)canonicalize(&r, &y);** **else r = fmax(x, y);** **return r;** **}"**
GB	Page 33	14.5		te	The functions that round once to a narrower type will have efficiency varying widely depending on hardware support.. There should be macros to indicate whether these functions are efficiently supported, similar to FP_FAST_FMA.	Page 33, add at the start of the changes to C11: After 7.12#7, insert: The macros FP_FAST_FADD FP_FAST_FADDL FP_FAST_DADDL FP_FAST_FSUB FP_FAST_FSUBL	Agreed

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation
						FP_FAST_DSUBL FP_FAST_FMUL FP_FAST_FMULL FP_FAST_DMULL FP_FAST_FDIV FP_FAST_FDIVL FP_FAST_DDIVL FP_FAST_FFMA FP_FAST_FFMAL FP_FAST_DFMAL FP_FAST_FSQRT FP_FAST_FSQRTL FP_FAST_DSQRTL are optionally defined. If defined, they indicate that the corresponding function generally executes about as fast as, or faster than, the corresponding operation for the argument type (with result type the same as the argument type) followed by a separate conversion to the narrower type. (For FP_FAST_FFMA, FP_FAST_FFMAL and FP_FAST_DFMAL, the comparison is to a call to fma or fmal followed by a conversion, not to separate multiply, add and conversion.) If defined, these macros expand to the integer constant 1.	
GB	Page 41 lines 4-33	14.10		te	It seems unclear if the setpayload / setpayloadsig functions are permitted to raise the "invalid" exception if the specified payload value is a signaling NaN.	At ends of lines 15 and 30, insert “These functions raise no floating-point exceptions, even if pl is a signaling NaN.”.	Leaving this unspecified gives the implementation the usual flexibility for signaling NaN support, as intended. Suggest no change

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial

Template for comments and secretariat observations

Date: 11 Sep 2013	Document: **WG 14 N1754**	Project: 18661

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	FP teleconference group recommendation

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:ge** = general **te** = technical **ed** = editorial