IEC 60559-3 ANNEX UPDATE

N2578
WG 14 – online meeting
October 2020

C FP group

Background

- Goal: C support for interchange and extended formats specified in IEC 60559:2011 (and 2020).
- ISO/IEC TS 18661-3 “Interchange and extended types” published 2015.
- Recast as C2X annex in N2342.
- Presented to WG14. See slide deck N2374.
- Approved by WG14 for C2X.
- Rebased to then current C2X draft in N2405.
- Incorporation into C2X not completed.
- Updated in N2579 (based on C2X draft N2478) …

Changes in update

- X.2.1 #1 and X.2.3 #1 Revised tables of type parameters to use C instead of IEC 60559 model parameters.

  Editorial.

Binary interchange format parameters

Parameter	binary16	binary32	binary64	binary128	binaryN (N ≥ 128)
N, storage width in bits	16	32	64	128	N a multiple of 32
p, precision in bits	11	24	53	113	$N - \text{round}(4\times\log_2(N)) + 13$
emax, maximum exponent e	16	128	1024	16384	$2^{(N-p-1)}$
emin, minimum exponent e	−13	−125	−1021	−16381	3 − emax

Changes in update

- X.2.3 Added requirement that extended types of the same radix be ordered.

> [2] … The set of values of **_Float32x** is a subset of the set of values of **_Float64x**; the set of values of **_Float64x** is a subset of the set of values of **_Float128x**. The set of values of **_Decimal64x** is a subset of the set of values of **_Decimal128x**. …

For usual arithmetic conversions and tgmath rules, this reduces the number of cases where undefined behavior is possible.

Not an IEC 60559 requirement (maybe an oversight).

Changes in update

- X.8 Removed the lists of identifiers per header and conditional macros. Now just

> [1] The identifiers added to library headers by this annex are defined or declared by their respective headers only if the macro **__STDC_WANT_IEC_60559_TYPES_EXT__** is defined (by the user) at the point in the code where the appropriate header is first included.

Utility didn't justify the lengthy lists.

Conditionalities specified in relevant subclauses.

Changes in update

- Clarified relevant conditionalities, in various places. Examples:
  - Added to annex synopses:

```
#define __STDC_WANT_IEC_60559_TYPES_EXT__
```

  - X.11.3 Regarding encoding functions for non-arithmetic formats, added:

> … Support for these formats is an optional feature of this annex. Implementations that do not support non-arithmetic interchange formats need not declare the functions in this subclause.

Changes in update

- X.2.2 #2 Made support for binary16 optional. Previously required at least as non-arithmetic format.

  How many and which short floating-point formats will be needed is unclear.

- X.5 #6 Added determination for the quantum exponent of floating constants of decimal floating type.

> [6] The quantum exponent of a floating constant of decimal floating type is the same as for the result value of the corresponding **strtod***N* or **strtod***N***x** function (X.12.2) for the same numeric string.

Missing in N2405 -- oversight.

Changes in update

- X.10 Added missing **<fenv.h>** specification for new types.

  Missing in N2405 - oversight.

> [2] The same floating-point status flags are used by floating-point operations for all floating types, including those types introduced in this annex, and by conversions for IEC 60559 non-arithmetic interchange formats.

> [3] … Likewise, both the dynamic rounding direction mode accessed by **fe_dec_getround** and **fe_dec_setround** and the **FENV_DEC_ROUND** rounding control pragmas apply to operations for all the decimal floating types, including those decimal floating types introduced in this annex, and to conversions for radix-10 non-arithmetic interchange formats.

Changes in update

- X.11.2 Added prototypes for the mathematical function families from TS 18661-4a.

  Missing in N2405 - oversight.

```
…
_FloatN rsqrtfN(_FloatN x);
_FloatNx rsqrtfNx(_FloatNx x);
_DecimalN rsqrtdN(_DecimalN x);
_DecimalNx rsqrtdNx(_DecimalNx x);
…
```

Changes in update

- X.3 #2 and #3 Changed so that wide evaluation does not specify a type for the evaluation format that is different from the semantic type if the two types have the same values.

> **1** evaluate operations and constants, whose semantic type comprises a set of values that is a strict subset of the values of **double**, to the range and precision of **double**; evaluate all other operations and constants to the range and precision of the semantic type;

Not substantive for evaluation formats, but changes _t types which are defined as the type specified for the evaluation format. E.g., with **FLT_EVAL_METHOD** 1, **_Float64_t** was **double**, now is **_Float64_t**.

Changes in update

- X.11 #7 Added an example showing _t types, as determined by specification of evaluation formats.

Type	IEC 60559 format
_Float16	binary16
float, _Float32	binary32
double, _Float64, _Float32x	binary64
long double, _Float64x	80-bit binary64-extended
_Float128	binary128

	t type as determined by FLT_EVAL_METHOD m							
_t type \ m	0	1	2	32	64	128	33	65
_Float16_t	float	double	long double	_Float32	_Float64	_Float128	_Float32x	_Float64x
float_t	float	double	long double	float	_Float64	_Float128	_Float32x	_Float64x
_Float32_t	_Float32	double	long double	_Float32	_Float64	_Float128	_Float32x	_Float64x
double_t	double	double	long double	double	double	_Float128	double	_Float64x
_Float64_t	_Float64	_Float64	long double	_Float64	_Float64	_Float128	_Float64	_Float64x
long_double_t	long double	long double	long double	long double	long double	_Float128	long double	long double
_Float128_t	_Float128	_Float128	_Float128	_Float128	_Float128	_Float128	_Float128	_Float128

Changes in update

- X.11.3.1.1 (and elsewhere) Changed restrict pointer parameters to arrays with static restrict size.

```
void strtoencf N(unsigned char encptr[restrict static N/8],
                 const char * restrict nptr, char ** restrict endptr);
```

Changes in update

- X.11.3.2.1 #4 Added an example using **f32encf16** and **decodef32** for correctly rounded conversion from binary16 non-arithmetic format to **float**.

```
#define __STDC_WANT_IEC_60559_TYPES_EXT__
#include <math.h>
unsigned char b16[2];      // for input binary16 datum
float f;                   // for result
unsigned char b32[4];
_Float32 f32;
// store input binary16 datum in array b16
...
f32encf16(b32, b16);
decodef32(&f32, b32);
f = f32;
...
```

Changes in update

- X.12.2 #3 Added requirement that, for implementations supporting both binary and decimal types and at least one (binary or decimal) non-arithmetic interchange format, the strto functions for decimal types and formats accept input strings of hexadecimal form and correctly round for enough hexadecimal digits to represent all radix-2 types and formats.

  Supports IEC 60559 requirement for conversions. See new X.12.2 example.

> [3] For implementations that support both binary and decimal floating types and a (binary or decimal) non-arithmetic interchange format, the **strtod***N* and **strtod***N***x** functions (and hence the **strtoencdecd***N* and **strtoencbind***N* functions in X.12.4.2) shall accept subject sequences that have the form of hexadecimal floating numbers and otherwise meet the requirements of subject sequences (7.22.1.6). …

Changes in update

- X.12.2 #3 Added requirement continued …

… Then the decimal results shall be correctly rounded if the subject sequence has at most *M* significant hexadecimal digits, where $M \geq \lceil (P\text{-}1)/4 \rceil + 1$ is implementation defined, and *P* is the maximum precision of the supported binary floating types and binary non-arithmetic formats. If all subject sequences of hexadecimal form are correctly rounded, *M* may be regarded as infinite. If the subject sequence has more than *M* significant hexadecimal digits, the implementation may first round to *M* significant hexadecimal digits according to the applicable rounding direction mode, signaling exceptions as though converting from a wider format, then correctly round the result of the shortened hexadecimal input to the result type.

Changes in update

- X.12.2 #4 Added an example using **strfromencf128** and **strtod128** for correctly rounded conversion from binary128 non-arithmetic format to **_Decimal128** type.

```
#define __STDC_WANT_IEC_60559_TYPES_EXT__
#include <stdlib.h>
#define MAXSIZE 41              // > intermediate hex string length
unsigned char b128[16];         // for input binary128 datum
_Decimal128 d128;               // for result
char s[MAXSIZE];
// store input binary128 datum in array b128
...
strfromencf128(s, MAXSIZE, “%a”, b128);
d128 = strtod128(s, NULL);
...
```

Changes in update

- X.12.2 #4 Example continued ….

Use of **“%a”** for formatting assures an exact conversion of the value in binary format to character sequence. The value of that character sequence will be correctly rounded to **_Decimal128**, as specified above in this subclause. The array **s** for the output of **strfromencf128** need have no greater size than 41, which is the maximum length of strings of the form

[-]**0x***h***.***h*…*h***p**±*d*, where there are up to 29 hexadecimal digits *h* and *d* has 5 digits

plus 1 for the null character.

Changes in update

- X.13 #3 Refined the way tgmath resolution rules regard integer-type arguments.

The treatment of arguments of integer type in 7.25 is expanded to cases where another argument has extended type. Arguments of integer type are regarded as having type:

_Decimal64x	if any argument has a decimal extended type; otherwise
_Float32x	if any argument has a binary extended type; otherwise
_Decimal64	if any argument has decimal type; otherwise
double	

Avoids undefined behavior, e.g.,

```
#include <tgmath.h>
_Float32x f32x;
… pow(f32x, 2) …
```

had undefined behavior if **_Float32x** *and* **double** *are not ordered (by inclusion of values), now invokes* **powf32x**.

Changes in update

- X.13 #4 Enhanced the tgmath resolution rules for functions that round to narrower type, so that arguments of equivalent standard and binary type are used interchangeably.

```
_Float32 x, y;
double d;
…
y = f32add(x, d);
```

was undefined, now calls **f32addf64**.

Changes in update

- X.13 #4 Enhanced tgmath resolution rules continued ….

… The following specification uses the notation *type*1 ⊆ *type*2 to mean the values of *type*1 are a subset of (or the same as) the values of *type*2. The generic parameter type *T* for the function invoked by the macro is determined as follows:

— First apply the rules (for determining the corresponding real type of the generic parameters) in 7.25 for macros that do not round result to narrower type, using the usual arithmetic conversion rules in X.4.2, to obtain a preliminary type *P* for the generic parameters.
— If there exists a corresponding function whose generic parameters have type *P*, then *T* is *P*.
— Otherwise, *T* is determined from *P* and the macro prefix as follows:

…

Changes in update

- X.13 #4 Enhanced tgmath resolution rules continued ….

…

— For prefix **f***M*: If *P* is a standard or binary floating type, then *T* is **_Float***N* for minimum *N* > *M* such that *P* ⊆ *T*, if such a type *T* is supported; otherwise *T* is **_Float***N***x** for minimum *N* ≥ *M* such that *P* ⊆ *T*, if such a type *T* is supported. Otherwise (if no such **_Float***N* or **_Float***N***x** is supported or *P* is a decimal floating type), the behavior in undefined.

…

Changes in update

- X.13 #4 Updated and expanded the example.

pow(f32x, n) **pow32x**

Macros that round result to narrower type …

fsub(d, ld)	**fsubl**
dsub(d, f32)	**dsubl**
fmul(dc, d)	undefined
ddiv(ld, f128)	**ddivl** if **_Float128** ⊆ **long double**, else undefined
f32add(f64x, f64)	**f32addf64x**
f32xsqrt(n)	**f32xsqrtf64**
f32mul(f128, f32x)	**f32mulf128** if **_Float32x** ⊆ **_Float128**, else **f32mulf32x** if **_Float128** ⊂ **_Float32x**, else undefined

…

Changes in update

- ~~X.6 #3 Made **_Float16**, **_Float32**, **_Float64** be subject to default argument promotions like **float** is.~~

> ~~[3] The *default argument promotions* (6.5.2.2) for functions whose type does not include a prototype are expanded so that arguments that have type **_Float16**, **_Float32**, or **_Float64** are promoted to **double**.~~

~~*Allows these types to be printf arguments.*~~

~~*Synchronizes with C++ proposal for extension floating types.*~~