Submitter:CFP group
Submission Date: 2020-08-03
Document: WG14 N2548
Title: N2548: intmax_t and math functions
Reference Documents: N2478, N2525

Summary

Due to issues raised by the WG14 committee with [u]intmax_t, several of the CFP related functions ([u]fromfp[x], compoundn, pown, rootn) should be changed to not use [u]intmax_t.

The WG14 consideration of removing [u]intmax_t, sparked the CFP discussion. We concluded that changing the interfaces to not use [u]intmax_t would be better anyway (even if [u]intmax_t were not removed). Introducing type [u]intmax_t into an expression via a function return type could have unpredictable negative performance implications.

In general, replace [u]intmax_t with the floating type used as the parameter.

Change

> 7.12.9.10 The fromfp and ufromfp functions
>
> Synopsis

```
1 #include <stdint.h>
  #include <math.h>
  intmax_t fromfp(double x, int round, unsigned int width);
  intmax_t fromfpf(float x, int round, unsigned int width);
  intmax_t fromfpl(long double x, int round, unsigned int width);
  uintmax_t ufromfp(double x, int round, unsigned int width);
  uintmax_t ufromfpf(float x, int round, unsigned int width);
  uintmax_t ufromfpl(long double x, int round, unsigned int width);
  #ifdef __STDC_IEC_60559_DFP__
  intmax_t fromfpd32(_Decimal32 x, int round, unsigned int width);
  intmax_t fromfpd64(_Decimal64 x, int round, unsigned int width);
  intmax_t fromfpd128(_Decimal128 x, int round, unsigned int width);
  uintmax_t ufromfpd32(_Decimal32 x, int round, unsigned int width);
  uintmax_t ufromfpd64(_Decimal64 x, int round, unsigned int width);
  uintmax_t ufromfpd128(_Decimal128 x, int round, unsigned int width);
  #endif
```

> Description
>
> 2 The **fromfp** and **ufromfp** functions round x, using the math rounding direction indicated by **round**, to a signed or unsigned integer, respectively, of **width** bits, and return the result value in the integer type designated by **intmax_t** or **uintmax_t**, respectively. If the value of the **round** argument is not equal to the value of a math rounding direction macro, the direction of rounding is unspecified. If the value of **width** exceeds the width of the function type, the rounding is to the full width of the function type. The **fromfp** and **ufromfp** functions do not raise the "inexact" floating-point exception. If x is infinite or NaN or rounds to an integral value that is outside the range of any supported integer type of the specified width, or if width is zero, the functions return an unspecified value and a domain error occurs.

to

> 7.12.9.10 The fromfp and ufromfp functions
>
> Synopsis

```
1 #include <stdint.h>
  #include <math.h>
  intmax_t double fromfp(double x, int round, unsigned int width);
  intmax_t float fromfpf(float x, int round, unsigned int width);
  intmax_t long double fromfpl(long double x, int round, unsigned int width);
  uintmax_t double ufromfp(double x, int round, unsigned int width);
  uintmax_t float ufromfpf(float x, int round, unsigned int width);
  uintmax_t long double ufromfpl(long double x, int round, unsigned int width);
  #ifdef __STDC_IEC_60559_DFP__
  intmax_t _Decimal32 fromfpd32(_Decimal32 x, int round, unsigned int width);
  intmax_t _Decimal64 fromfpd64(_Decimal64 x, int round, unsigned int width);
  intmax_t _Decimal128 fromfpd128(_Decimal128 x, int round, unsigned int width);
  uintmax_t _Decimal32 ufromfpd32(_Decimal32 x, int round, unsigned int width);
  uintmax_t _Decimal64 ufromfpd64(_Decimal64 x, int round, unsigned int width);
  uintmax_t _Decimal128 ufromfpd128(_Decimal128 x, int round, unsigned int width);
  #endif
```

Description

2 The **fromfp** and **ufromfp** functions round x, using the math rounding direction indicated by **round**, to a signed or unsigned integer, respectively. ~~**, of width bits, and return the result value in the integer type designated by intmax_t or uintmax_t, respectively.**~~ **If width is nonzero and the resulting integer is within the range**

```
   [-2**(width-1), 2**(width-1) - 1], for signed
   [0, 2**width - 1], for unsigned
```

the functions return the integer value (represented in floating type). Otherwise, if width is zero or x does not round to an integer within the range, the functions return a NaN (of the type of the x argument, if available), else the value of x, and a domain error occurs. If the value of the **round** argument is not equal to the value of a math rounding direction macro **(7.12)**, the direction of rounding is unspecified. ~~**If the value of width exceeds the width of the function type, the rounding is to the full width of the function type.**~~ The **fromfp** and **ufromfp** functions do not raise the "inexact" floating-point exception. ~~**If x is infinite or NaN or rounds to an integral value that is outside the range of any supported integer type of the specified width, or if width is zero, the functions return an unspecified value and a domain error occurs.**~~

And insert another example after the existing example:

5 EXAMPLE Unsigned integer wrapping is not performed in

```
ufromfp(-3.0, FP_INT_UPWARD, UINT_WIDTH) /* domain error */
```

Also, in F.10.6.10:

Change:

1 The fromfp and ufromfp functions raise the "invalid" floating-point exception and return an unspecified value if the floating-point argument x is infinite or NaN or rounds to an integral value that is outside the range of any supported integer type of the specified width.

to:

1 The fromfp and ufromfp functions raise the "invalid" floating-point exception and return ~~**an unspecified value**~~ **a NaN** if the **argument width is zero or if the** floating-point argument x is infinite or NaN or rounds to an integer value that is outside the range ~~**of any supported integer type of the specified width**~~ **determined by the argument width (see 7.12.9.10)**.

Change

7.12.9.11 The fromfpx and ufromfpx functions

Synopsis

```
1 #include <stdint.h>
  #include <math.h>
  intmax_t fromfpx(double x, int round, unsigned int width);
  intmax_t fromfpfx(float x, int round, unsigned int width);
  intmax_t fromfplx(long double x, int round, unsigned int width);
  uintmax_t ufromfpx(double x, int round, unsigned int width);
  uintmax_t ufromfpfx(float x, int round, unsigned int width);
  uintmax_t ufromfplx(long double x, int round, unsigned int width);
  #ifdef __STDC_IEC_60559_DFP__
  intmax_t fromfpd32x(_Decimal32 x, int round, unsigned int width);
  intmax_t fromfpd64x(_Decimal64 x, int round, unsigned int width);
  intmax_t fromfpd128x(_Decimal128 x, int round, unsigned int width);
  uintmax_t ufromfpd32x(_Decimal32 x, int round, unsigned int width);
  uintmax_t ufromfpd64x(_Decimal64 x, int round, unsigned int width);
  uintmax_t ufromfpd128x(_Decimal128 x, int round, unsigned int width);
  #endif
```

Description

to

7.12.9.11 The fromfpx and ufromfpx functions

Synopsis

```
1 #include <stdint.h>
```

```
#include <math.h>
~~intmax_t~~ double fromfpx(double x, int round, unsigned int width);
~~intmax_t~~ float fromfpfx(float x, int round, unsigned int width);
~~intmax_t~~ long double fromfplx(long double x, int round, unsigned int width);
~~uintmax_t~~ double ufromfpx(double x, int round, unsigned int width);
~~uintmax_t~~ float ufromfpfx(float x, int round, unsigned int width);
~~uintmax_t~~ long double ufromfplx(long double x, int round, unsigned int width);
#ifdef __STDC_IEC_60559_DFP__
~~intmax_t~~ _Decimal32 fromfpd32x(_Decimal32 x, int round, unsigned int width);
~~intmax_t~~ _Decimal64 fromfpd64x(_Decimal64 x, int round, unsigned int width);
~~intmax_t~~ _Decimal128 fromfpd128x(_Decimal128 x, int round, unsigned int width);
~~uintmax_t~~ _Decimal32 ufromfpd32x(_Decimal32 x, int round, unsigned int width);
~~uintmax_t~~ _Decimal64 ufromfpd64x(_Decimal64 x, int round, unsigned int width);
~~uintmax_t~~ _Decimal128 ufromfpd128x(_Decimal128 x, int round, unsigned int width);
#endif
```

Description

Also, in F.10.6.11:

Change:

> The fromfpx and ufromfpx functions raise the "invalid" floating-point exception and return an unspecified value if the floating-point argument x is infinite or NaN or rounds to an integral value that is outside the range of any supported integer type of the specified width.

to:

> The fromfpx and ufromfpx functions raise the "invalid" floating-point exception and return ~~**an unspecified value**~~**a NaN** if the floating-point argument x is infinite or NaN or rounds to an integral value that is outside the range ~~**of any supported integer type of the specified width**~~**determined by the argument width (see 7.12.9.11)**.

The three function families compoundn, rootn, pown have the simple change of replacing **intmax_t** with **long long int**.

Change

```
7.12.7.2 The compoundn functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double compoundn(double x, intmax_t n);
  float compoundnf(float x, intmax_t n);
  long double compoundnl(long double x, intmax_t n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 compoundnd32(_Decimal32 x, intmax_t n);
  _Decimal64 compoundnd64(_Decimal64 x, intmax_t n);
  _Decimal128 compoundnd128(_Decimal128 x, intmax_t n);
  #endif
```

to

```
7.12.7.2 The compoundn functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double compoundn(double x, ~~intmax_t~~long long int n);
  float compoundnf(float x, ~~intmax_t~~long long int n);
  long double compoundnl(long double x, ~~intmax_t~~long long int n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 compoundnd32(_Decimal32 x, ~~intmax_t~~long long int n);
  _Decimal64 compoundnd64(_Decimal64 x, ~~intmax_t~~long long int n);
  _Decimal128 compoundnd128(_Decimal128 x, ~~intmax_t~~long long int n);
  #endif
```

Change

```
7.12.7.6 The pown functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double pown(double x, intmax_t n);
  float pownf(float x, intmax_t n);
  long double pownl(long double x, intmax_t n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 pownd32(_Decimal32 x, intmax_t n);
  _Decimal64 pownd64(_Decimal64 x, intmax_t n);
  _Decimal128 pownd128(_Decimal128 x, intmax_t n);
```

```
  #endif
```

to

```
7.12.7.6 The pown functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double pown(double x, intmax_t long long int n);
  float pownf(float x, intmax_t long long int n);
  long double pownl(long double x, intmax_t long long int n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 pownd32(_Decimal32 x, intmax_t long long int n);
  _Decimal64 pownd64(_Decimal64 x, intmax_t long long int n);
  _Decimal128 pownd128(_Decimal128 x, intmax_t long long int n);
  #endif
```

Change

```
7.12.7.8 The rootn functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double rootn(double x, intmax_t n);
  float rootnf(float x, intmax_t n);
  long double rootnl(long double x, intmax_t n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 rootnd32(_Decimal32 x, intmax_t n);
  _Decimal64 rootnd64(_Decimal64 x, intmax_t n);
  _Decimal128 rootnd128(_Decimal128 x, intmax_t n);
  #endif
```

to

```
7.12.7.8 The rootn functions
Synopsis
1 #include <stdint.h>
  #include <math.h>
  double rootn(double x, intmax_t long long int n);
  float rootnf(float x, intmax_t long long int n);
  long double rootnl(long double x, intmax_t long long int n);
  #ifdef __STDC_IEC_60559_DFP__
  _Decimal32 rootnd32(_Decimal32 x, intmax_t long long int n);
  _Decimal64 rootnd64(_Decimal64 x, intmax_t long long int n);
  _Decimal128 rootnd128(_Decimal128 x, intmax_t long long int n);
  #endif
```

For all of the above, also update the prototypes in Annex B.