P1 CR for obsolescing `DECIMAL_DIG`

WG 14 N2254
2018-05-11
C FP Group

TS 18661-1 CR 20
==

Reference Document: CR501, N2211, N2253**,** TS 18661-1

Subject: changes for obsolescing DECIMAL_DIG

Summary

N2211 described changes in C11 and TS 18661 to remove references to **`DECIMAL_DIG`**, which CR501 is expected to obsolesce. The changes that apply to C11 are collected in N2253 as an update to the suggested TC in CR501. The changes that apply to TS 18661-1 compose the CR in this document. The remaining change is for TS 18661-3, which will be covered by a CR in a subsequent document.

Suggested Technical Corrigendum

In 7.1, omit:

Change footnote 361) from:

> 361) If the minimum-width IEC60559 extended format (64 bits of precision) is supported, **`DECIMAL_DIG`** shall be at least 21. If IEC 60559 double (53 bits of precision) is the widest IEC 60559 format supported, then **`DECIMAL_DIG`** shall be at least 17. (By contrast, **`LDBL_DIG`** and **`DBL_DIG`** are 18 and 15, respectively, for these formats.)

to:

> 361) If the minimum-width IEC 60559 binary64-extended format (64 bits of precision) is supported, **`DECIMAL_DIG`** shall be at least 21. If IEC 60559 binary64 (53 bits of precision) is the widest IEC 60559 format supported, then **`DECIMAL_DIG`** shall be at least 17. (By contrast, **`LDBL_DIG`** and **`DBL_DIG`** are 18 and 15, respectively, for these formats.)

In 10.1, change:

After F.5#2, insert:

> [2a] The **`<float.h>`** header defines the macro
>
> **`CR_DECIMAL_DIG`**

if and only if **`__STDC_WANT_IEC_60559_BFP_EXT__`** is defined as a macro at the point in the source file where **`<float.h>`** is first included. If defined, **`CR_DECIMAL_DIG`** expands to an integral constant expression suitable for use in **`#if`** preprocessing directives whose value is a number such that conversions between all supported types with IEC 60559 binary formats and character sequences with at most **`CR_DECIMAL_DIG`** significant decimal digits are correctly rounded. The value of **`CR_DECIMAL_DIG`** shall be at least **`DECIMAL_DIG`** + 3. If the implementation correctly rounds for all numbers of significant decimal digits, then **`CR_DECIMAL_DIG`** shall have the value of the macro **`UINTMAX_MAX`**.

[2b] Conversions of types with IEC 60559 binary formats to character sequences with more than **`CR_DECIMAL_DIG`** significant decimal digits shall correctly round to **`CR_DECIMAL_DIG`** significant digits and pad zeros on the right.

[2c] Conversions from character sequences with more than **`CR_DECIMAL_DIG`** significant decimal digits to types with IEC 60559 binary formats shall correctly round to an intermediate character sequence with **`CR_DECIMAL_DIG`** significant decimal digits, according to the applicable rounding direction, and correctly round the intermediate result (having **`CR_DECIMAL_DIG`** significant decimal digits) to the destination type. The "inexact" floating-point exception is raised (once) if either conversion is inexact. (The second conversion may raise the "overflow" or "underflow" floating-point exception.)

In F.5#2c, attach a footnote to the wording:

The "inexact" floating-point exception is raised (once) if either conversion is inexact.

where the footnote is:

*) The intermediate conversion is exact only if all input digits after the first **`CR_DECIMAL_DIG`** digits are **`0`**.

to:

Replace the content of F.5 with:

[1] The **`<float.h>`** header defines the macro

`CR_DECIMAL_DIG`

if and only if **`__STDC_WANT_IEC_60559_BFP_EXT__`** is defined as a macro at the point in the source file where **`<float.h>`** is first included. If defined, **`CR_DECIMAL_DIG`** expands to an integral constant expression suitable for use in **#if** preprocessing directives whose value is a number such that conversions between all supported IEC 60559 binary formats and character sequences with at most **`CR_DECIMAL_DIG`** significant decimal digits are correctly rounded. The value of **`CR_DECIMAL_DIG`** shall be at least M + 3, where M is the maximum value of the T**`_DECIMAL_DIG`** macros for IEC 60559 binary formats. If the implementation correctly rounds for all numbers of significant decimal digits, then **`CR_DECIMAL_DIG`** shall have the value of the macro **`UINTMAX_MAX`**.

[2] Conversions of types with IEC 60559 binary formats to character sequences with more than **`CR_DECIMAL_DIG`** significant decimal digits shall correctly round to **`CR_DECIMAL_DIG`** significant digits and pad zeros on the right.

[3] Conversions from character sequences with more than **`CR_DECIMAL_DIG`** significant decimal digits to types with IEC 60559 binary formats shall correctly round to an intermediate character sequence with **`CR_DECIMAL_DIG`** significant decimal digits, according to the applicable rounding direction, and correctly round the intermediate result (having **`CR_DECIMAL_DIG`** significant decimal digits) to the destination type. The "inexact" floating-point exception is raised (once) if either conversion is inexact. (The second conversion may raise the "overflow" or "underflow" floating-point exception.)

[4] The specification in this subclause assures conversion between IEC 60559 binary format and decimal character sequence follows all pertinent recommended practice. It also assures conversion from IEC 60559 format to decimal character sequence with at least *T*_**`DECIMAL_DIG`** digits and back, using to-nearest rounding, is the identity function, where *T* is the macro prefix for the format.

[5] Functions such as **`strtod`** that convert character sequences to floating types honor the rounding direction. Hence, if the rounding direction might be upward or downward, the implementation cannot convert a minus-signed sequence by negating the converted unsigned sequence.

In F.5#3, attach a footnote to the wording:

The "inexact" floating-point exception is raised (once) if either conversion is inexact.

where the footnote is:

*) The intermediate conversion is exact only if all input digits after the first **`CR_DECIMAL_DIG`** digits are **`0`**.