WG 14 N2211

Submitter: C FP Group
Submission Date: 2018-03-16
Source: WG14
Reference Documents: DR 501, N2108, C11, TS 18661-1, TS 18661-2
Subject: Changes for obsolescing DECIMAL_DIG

Summary

N2108 suggested obsolescing **`DECIMAL_DIG`**, as part of the resolution of DR 501. The purpose of this paper is to identify places in C11 and TS 18661 that currently reference **`DECIMAL_DIG`** and to suggest changes that eliminate these references.

C11 5.2.4.2.2

Change:

> Conversion from (at least) **`double`** to decimal with **`DECIMAL_DIG`** digits and back should be the identity function.

to:

> Conversion between real floating type and decimal character sequence with at most *T*`_DECIMAL_DIG` digits should be correctly rounded, where *T* is the macro prefix for the type. This assures conversion from real floating type to decimal character sequence with *T*`_DECIMAL_DIG` digits and back, using to-nearest rounding, is the identity function.

Omit:

> If a type wider than **`double`** were supported, then **`DECIMAL_DIG`** would be greater than 17. For example, if the widest type were to use the minimal-width IEC 60559 double-extended format (64 bits of precision), then **`DECIMAL_DIG`** would be 21.

C11 7.21.6.1 and 7.29.2.1

Change:

> For **`e`**, **`E`**, **`f`**, **`F`**, **`g`**, and **`G`** conversions, if the number of significant decimal digits is at most **`DECIMAL_DIG`**, then the result should be correctly rounded.283) If the number of significant decimal digits is more than **`DECIMAL_DIG`** but the source value is exactly representable with **`DECIMAL_DIG`** digits, then the result should be an exact representation with trailing zeros. Otherwise, the source value is bounded by two adjacent decimal strings $L < U$, both having **`DECIMAL_DIG`** significant digits; the value of the resultant decimal string D should satisfy $L \le D \le U$, with the extra stipulation that the error should have a correct sign for the current rounding direction.

to:

> For **e**, **E**, **f**, **F**, **g**, and **G** conversions, if the number of significant decimal digits is at most the maximum value *M* of the *T*`_DECIMAL_DIG` macros (defined in **<float.h>**), then the result should be correctly rounded.283) If the number of significant decimal digits is more than *M* but the source value is exactly representable with *M* digits, then the result should be an exact representation with trailing zeros. Otherwise, the source value is bounded by two adjacent decimal strings $L < U$, both having *M* significant digits; the value of the resultant decimal string *D* should satisfy $L \leq D \leq U$, with the extra stipulation that the error should have a correct sign for the current rounding direction.

C11 7.22.1.3 and 7.29.4.1

Change:

> If the subject sequence has the decimal form and at most `DECIMAL_DIG` (defined in **<float.h>**) significant digits, the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than `DECIMAL_DIG` significant digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having `DECIMAL_DIG` significant digits, such that the values of *L*, *D*, and *U* satisfy $L \leq D \leq U$. The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.294)

to:

> If the subject sequence has the decimal form and at most *M* significant digits, where *M* is the maximum value of the *T*`_DECIMAL_DIG` macros (defined in **<float.h>**), the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than *M* digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having *M* significant digits, such that the values of *L*, *D*, and *U* satisfy $L \leq D \leq U$. The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.294)

C11 Footnotes 294 and 345

Change:

> `DECIMAL_DIG`, defined in **<float.h>**, should be sufficiently large that *L* and *U* will usually round to the same internal floating value, but if not will round to adjacent values.

to:

> *M* is sufficiently large that *L* and *U* will usually correctly round to the same internal floating value, but if not will correctly round to adjacent values.

C11 F.5

The following change is needed only if TS 18661-1, with the change further below to F.5, is not incorporated into C.

Change:

> Conversion from the widest supported IEC 60559 format to decimal with **DECIMAL_DIG** digits and back is the identity function.361)
>
> Conversions involving IEC 60559 formats follow all pertinent recommended practice. In particular, conversion between any supported IEC 60559 format and decimal with **DECIMAL_DIG** or fewer significant digits is correctly rounded (honoring the current rounding mode), which assures that conversion from the widest supported IEC 60559 format to decimal with **DECIMAL_DIG** digits and back is the identity function.

to:

> Conversions involving IEC 60559 formats follow all pertinent recommended practice. Conversion between any supported IEC 60559 format and decimal character sequence with *M* or fewer significant digits is correctly rounded (honoring the current rounding mode), where *M* is the maximum value of the *T*_**DECIMAL_DIG** macros (defined in **<float.h>**). Conversion from any supported IEC 60559 format to decimal character sequence with at least *T*_**DECIMAL_DIG** digits (for the corresponding *type*) and back, using to-nearest rounding, is the identity function.

C11 Footnote 361

Omit:

> If the minimum-width IEC 60559 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.)

TS 18661-1 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.)

TS 18661-1 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**.

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] … same as [2b]

[3] … same as [2c]

[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.

TS 18661-3 Clause 7

Omit:

In 5.2.4.2.2#11, change the bullet defining **DECIMAL_DIG** from:

— number of decimal digits, *n*, such that any floating-point number in the widest supported floating type with …

to:

— number of decimal digits, *n*, such that any floating-point number in the widest of the supported floating types and the supported IEC 60559 encodings with …