Update to N2108 suggested TC for CR501
WG 14 N2253

Submitter: C FP Group
Submission Date: 2018-05-11
Source: WG14
Reference Documents: DR 501, N2108, C11, N2211
Subject: Changes for obsolescing **`DECIMAL_DIG`**

Summary

N2108 suggested obsolescing **`DECIMAL_DIG`**, as part of the resolution of CR 501. This document updates the suggested CR in N2108 to eliminate references in C11 to **`DECIMAL_DIG`**, and to clarify. Changes below (along with changes to TS 18661) were identified in N2211.

Suggested Technical Corrigendum

In 7.31, add a subclause:

> ### 7.31.x Mathematics `<math.h>`
>
> Use of the **`DECIMAL_DIG`** macro is an obsolescent feature. A similar type-specific macro, such as **`LDBL_DECIMAL_DIG`** can be used instead.

In 5.2.4.2.2#11, in the bullet defining **`DECIMAL_DIG`**, attach a footnote to the wording:

> **`DECIMAL_DIG`**

where the footnote is:

> *) See ‘‘future library directions’’ (7.31.x).

In 5.2.4.2.2#14, change:

> [14] Conversion from (at least) **`double`** to decimal with **`DECIMAL_DIG`** digits and back should be the identity function.

to:

> [14] 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.

In 5.2.4.2.2#16, in the list of macro values in EXAMPLE 2, omit:

```
DECIMAL_DIG          17
```

In 5.2.4.2.2#16, at the end of EXAMPLE 2, 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.

In 7.21.6.1#13 and 7.29.2.1#13, change:

[13] 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:

[13] 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 $L \le D \le U$, with the extra stipulation that the error should have a correct sign for the current rounding direction.

In 7.22.1.3#9 and 7.29.4.1.1#9, change:

[9] 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 $L \le D \le U$. 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:

[9] 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 $L \le D \le U$. 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)

In 7.22.1.3 footnote 294 and 7.29.4.1.1 footnote 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.

In F.5, omit footnote 361:

> 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.)

The following change is needed only if TS 18661-1 (with CR 20) is not incorporated into C.

In F.5, replace::

> [1] Conversion from the widest supported IEC 60559 format to decimal with **DECIMAL_DIG** digits and back is the identity function.361)
>
> [2] 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.

with:

> [1] 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.

and renumber the subsequent paragraph.