Document: WG14/N1216
Date: 2007/03/26
Project: TR 24732
Authors: Jim Thomas, Rich Peterson
Reply to: Rich Peterson <Rich.Peterson@hp.com>

Subject: problems with TTDT

This paper notes that use of a translation time data type (TTDT) is incompatible with Annex F and 754R, and recommends removing TTDT specification from WG14/N1201 (TR 24732 draft of 2006/11/10).

C99 Annex F guarantees that source floating-point constants up to DECIMAL_DIG digits are correctly rounded and that translation-time floating-point arithmetic is as per ISO/IEC 60559 (IEEE 754), provided the evaluation is to float or double, or long double if it too is an ISO/IEC 60559 type. This means that constant initialization, under those conditions, is predictable and reproducible to the last bit.

To satisfy both Annex F and the current TR, the implementation would have to chose a TTDT with the precision of its correct-rounding threshold (>= DECIMAL_DIG). For implementations whose long double type is the common 128-bit version of IEEE-754 double-extended, DECIMAL_DIG is at least 36, which exceeds the 34-digit precision of _Decimal128. So _Decimal128 is clearly not wide enough to be a suitable candidate for the TTDT in those implementations.

Regardless of the number of digits used in the TTDT, translation-time arithmetic would be incorrect per Annex F. For example, for FLT_EVAL_METHOD equal 0 or 1, Annex F guarantees the binary value of double x = 0.1 + 0.2 to be 0x1.3333333333334p-2, but adding 0.1 + 0.2 in a TTDT to get 0.3 before converting to binary would yield 0x1.3333333333333p-2. Thus, Annex F and TTDT are incompatible. Moving to TTDT would introduce a silent change in

```
#include <stdio.h>
#include <float.h>
void launch_missle() { printf("duck\n"); }
int main() {
#if defined(__STDC_IEC_559__) && (FLT_EVAL_METHOD==0 ||
FLT_EVAL_METHOD==1)
        if (0.1 + 0.2 != 0x1.3333333333334p-2) launch_missle();
#endif
}
```

A pragma to interpret unsuffixed floating constants as _Decimal64 would address the usability problems mentioned in 7.1 in a manner similar to what was was done in C99 for generic floating constants.

Suggested TR changes:

Page 14, 7.1, last paragraph, insert new sentence at the end:

A pragma to interpret unsuffixed floating constants as _Decimal64 could address the usability problem mentioned here in a manner similar to what was was done in C99 for generic floating constants.

Remove 7.1.1.