WG 14 N1769
2013/10/03

Issue from Szabolcs Nagy:

round to narrower
==========

The floating-point extension to C (ISO/IEC TS 18661) in section 14.5
introduces new operations that round the result to narrower type.
The text claims

14.5
"The operations that round to the same and wider formats are already
available by casting operands of the builtin operators (+,-,*,/) to
the desired type and by casting the fma and sqrt functions to the
desired type."

which is not true when FLT_EVAL_METHOD!=0, casting does not solve
the double rounding issues which the new functions of 14.5 intend to
solve. On such platforms the functions which take float_t and double_t
argumets may be used to get narrower, same or wider results.

For example when FLT_EVAL_METHOD==2, dividing float x and y to float,
double and long double precisions can be done by

float r = fdivl(x,y);
double r = ddivl(x,y);
long double r = x/y;

(of course for float add, sub, mul no special functions are needed
because the result can be exactly represented in double and long double
precisions, so there is no double rounding issue. But for double
precision add, sub, mul and div operations with double inputs the
special functions are needed.)

<end of issue from Nagy>

Analysis

The statement from 14.5 is, as Nagy claims, not true. In his example, wide evaluation might cause two rounding errors in

```
r = x / y;
```

so this is not a computation of the IEC 60559 float divide of float operands. The functions that round result to narrower type can be used as suggested. There is a similar problem for IEC 60559 operations that are implemented as functions if they are allowed to return results with extra range and precision.

Proposed solution

The following changes to C11 clarify requirements implicit in binding to IEC 60559 operations, and fix some incorrect or misleading explanatory text.

Add at the end of new F.3, Page 12 (in N1756) :

> [11] IEC 60559 requires operations with specified operand and result formats. Therefore, math functions that are bound to IEC 60559 operations (see Table 1) must remove any extra range and precision from arguments or results. If an operator (+, -, *, or /) has an evaluation format wider than the semantic type (5.2.4.2.2), then an appropriate function that rounds result to narrower type (7.12.13a) might be needed for a proper IEC 60559 operation. For example, ddivl(x, y) computes a correctly rounded double divide of float x by float y, whereas (double)((double)x / (double)y) might have two rounding errors if the evaluation format for the divide is wider than double.

Remove the (incorrect) last sentence of the first paragraph of 14.5:

> The operations that round to the same and wider formats are already available by casting operands of the built-in operators (**+**, **-**, *, **/**) to the desired type and by calling the **fma** and **sqrt** functions of the desired type.

In Table 1, Page 10 (in N1756), merge each of the six rows beginning with a formatOf operation into the row above it. For example, change:

addition	**+**	6.5.6
formatOf addition with narrower format	**fadd**, **faddl**, **daddl**	7.12.13a.1, F.10.10a

to:

addition	**+**, **fadd**, **faddl**, **daddl**	6.5.6, 7.12.13a.1, F.10.10a

This last change includes widening operations, so provides a more complete accounting of IEC 60559 operations. IEC 60559 uses “addition” to refer to all of these operations.