Document: WG14 N1302
Date: 2008-04-17

Various models used for the [type-generic FP] comparision macros of 7.12.14.

```
Integer args?  Mixed FP types?  Solution

yes            yes              int _isless(long double x, long double y);
                                #define isless(x,y) _isless(x,y)

                                This is the slowest as extra conversions
                                would be done if x and y are the same type
                                and not long double.  It is the easiest to
                                implement as no magic is required.

yes            yes              int _islessl(long double x, long double y);
                                int _islessd( double x,  double y);
                                int _islessf(  float x,   float y);
                                #define isless(x,y) _tg_isless(x,y)

                                This treats isless(x,y) just like pow(x,y)
                                in <tgmath.h>; they both use the same magic
                                of type-generic macros.

yes            yes              ?<  [[an operator in the language]]

                                This should be equivalent to the above
                                unless the hardware supports compares
                                between different FP types; in which case
                                it would be faster as it avoids an explicit
                                conversion.  Unless the above were inline
                                functions, this would be faster in that it
                                avoids the function call overhead.

                                The above 3 are most general and slowest.

                                The next 2 honor the last sentance of
                                7.12.14 in that they require the arguments
                                to be real floating types.  Since they
                                use compiler magic, they could result in
                                inline instructions and avoid a function
                                call.

no             yes              int isless(real-floating x, real-floating y);
                                #define isless(x,y) _cm_isless(x,y)
                                along with compiler magic

                                This is intermediate between the other two
                                models.  This is what Tydeman believes that
                                C99 requires.

no             no               int isless(real-floating x, real-floating y);
                                #define isless(x,y) _cm_isless(x,y)
                                along with compiler magic

                                This is the most restrictive and the fastest
                                (of the function based solutions) as no
                                conversions need be done.  I heard P.J.
                                Plauger favor this model.
```

IEEE-754 only covers comparisons between floating-point values and
comparisons involving mixed FP types shall be supported. So, C doing
IEEE-754 comparisons where one operand is an integer value and the
other is a floating-point value is an extension to IEEE-754.

The classification macros of 7.12.3 need similar compiler magic.
fpclassify() and isnormal() are the only macros that truely require

magic as they must use the type of the argument (and isnormal() can be done as fpclassify()==FP_NORMAL). The others can all be done using functions of long double parameters; at a cost in performance.