Proposal for C2X
WG14 N2570

Title: Feature and WANT macros for Annex F functions
Author, affiliation: C FP group
Date: 2020-08-03
Proposal category: Technical
Reference: N2478

Problem:

Synopses in Annex F in the current C2X draft use feature and WANT macros in an irregular way. For example, F.10.12.1 has

```c
#define __STDC_WANT_IEC_60559_BFP_EXT__
#include <math.h>
int totalorder(const double *x, const double *y);
int totalorderf(const float *x, const float *y);
int totalorderl(const long double *x, const long double *y);
#ifdef __STDC_IEC_60559_DFP__
int totalorderd32(const _Decimal32 *x, const _Decimal32 *y);
int totalorderd64(const _Decimal64 *x, const _Decimal64 *y);
int totalorderd128(const _Decimal128 *x, const _Decimal128 *y);
#endif
```

which doesn’t mention `__STDC_WANT_IEC_60559_DFP_EXT__` or `__STDC_WANT_IEC_60559_BFP__`.

Discussion:

For consistency with other uses of feature and WANT macros in synopses, this might be split into two frames, as

```c
#define __STDC_WANT_IEC_60559_BFP_EXT__
#include <math.h>
#ifdef __STDC_IEC_60559_BFP__
int totalorder(const double *x, const double *y);
int totalorderf(const float *x, const float *y);
int totalorderl(const long double *x, const long double *y);
#endif
```
The separate WANT macros `__STDC_WANT_IEC_60559_BFP_EXT__` and `__STDC_WANT_IEC_60559_DFP_EXT__` for IEC 60559 binary and decimal floating point appearances to be left over from the TS 18661 requirement for a WANT macro for all decimal interfaces. This requirement was dropped when parts 1 and 2 of the TS were incorporated into C2X, except for interfaces in Annex F. One WANT macro such as `__STDC_WANT_IEC_60559_EXT__` could guard all the interfaces in Annex F. If the user defines the macro, then which Annex F interfaces become available would depend on whether the implementation defines `__STDC_IEC_60559_BFP__` and/or `__STDC_IEC_60559_DFP__`. Then, the Annex F synopses could have the one frame:

```
#define __STDC_WANT_IEC_60559_EXT__
#include <math.h>
#ifdef __STDC_IEC_60559_BFP__
 int totalorderd32(const _Decimal32 *x, const _Decimal32 *y);
 int totalorderd64(const _Decimal164 *x, const _Decimal164 *y);
 int totalorderd128(const _Decimal128 *x, const _Decimal128 *y);
#endif

#define __STDC_WANT_IEC_60559_EXT__
#include <math.h>
#ifdef __STDC_IEC_60559_DFP__
 int totalorderd32(const _Decimal32 *x, const _Decimal32 *y);
 int totalorderd64(const _Decimal164 *x, const _Decimal164 *y);
 int totalorderd128(const _Decimal128 *x, const _Decimal128 *y);
#endif
```

Note that the only interfaces in question are for total order and payload functions, which seem unlikely to pose namespace problems. The interfaces in Annex G (`imaginary` and `_Imaginary_I`) are not guarded by a WANT macro. These considerations suggest a WANT macro for Annex F interfaces isn’t really needed.

The following suggested changes eliminate the WANT macro for Annex F interfaces and clean up the synopses in Annex F. The changes to not cover Annex B which will be discussed in a separate paper.

**Suggested changes:**

Delete F.1 #6:

> [6] This annex amends some standard headers with declarations or definitions of identifiers contingent on whether certain macros whose names begin with `__STDC_WANT_IEC_60559__` and end with `__EXT__` are defined (by the user) at the point in the code where the header is first included.
translation unit, the same set of such macros shall be defined for the first inclusion of all such headers.

Change F.5 #1:

[1] The `<float.h>` header defines the macro

```c
#define __STDC_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 it is first included. If defined, `__STDC_DECIMAL_DIG` which expands to a ...

In F.10.12.1 #1, change the Synopsis:

```c
#define __STDC_WANT_IEC_6059_BFP_EXT__
#include <math.h>
#ifdef __STDC_IEC_60559_BFP__
  int totalorder(const double *x, const double *y);
  int totalorderf(const float *x, const float *y);
  int totalorderl(const long double *x, const long double *y);
#endif
#ifdef __STDC_IEC_60559_DFP__
  int totalorderd32(const _Decimal32 *x, const _Decimal32 *y);
  int totalorderd64(const _Decimal64 *x, const _Decimal64 *y);
  int totalorderd128(const _Decimal128 *x, const _Decimal128 *y);
#endif
```