Proposal for C2X
WG14 N2558

Title:	**Annex B with prototype forms**
Author, affiliation:	**C FP group**
Date:	**2020-08-06**
Proposal category:	**Editorial**
Reference:	**N2478**

This note suggests a simpler library summary B.11 for **`<math.h>`** vs the one in N 2478.

The suggested summary does not include the statements of conditionality (dependencies on feature and WANT macros) that are in B.11 in the N 2478 C2X draft. Also, it does not include restrictions on forms, e.g., *Type*`_t` is not defined for all types. This information is in the relevant library subclauses.

This approach gives a much simpler and, we believe, generally more useful summary.

With this formulation, support for the TS 18661-3 annex is a modest extension to the library summary. In the suggested change below, the last three rows of the table are for the TS 18661-3 annex and the interfaces from the annex are indicated by 2-space indentation.

It is intended that the C2X editor will columnize the list and remove indentation if this approach is accepted.

Similar changes for **`<complex.h>`**, **`<float.h>`**, and **`<stdlib.h>`** are not shown here.

Suggested change:

Replace B.11 with:

B.11 Mathematics `<math.h>`

In the list below, *Type*, *StdType*, *DecType*, *BinType*, *FS* , and *MS* represent floating types and their associated function and macro suffixes:

real floating types	*Type*	*FS*	*MS*
standard floating types *StdType*	**`float`**	**`f`**	**`F`**
	`double`	none	none
	`long double`	**`l`**	**`L`**
decimal floating types *DecType*	**`_Decimal`***N*	**`d`***N*	**`D`***N*
	`_Decimal`*N***`x`**	**`d`***N***`x`**	**`D`***N***`X`**
binary floating types *BinType*	**`_Float`***N*	**`f`***N*	**`F`***N*
	`_Float`*N***`x`**	**`f`***N***`x`**	**`F`***N***`X`**

The symbol *N* (or *M*) represents a type width.

Type`_t`
`HUGE_VAL`*MS*

INFINITY
DEC_INFINITY
NAN
DEC_NAN
SNAN*MS*
FP_INFINITE
FP_NAN
FP_NORMAL
FP_SUBNORMAL
FP_ZERO
FP_INT_UPWARD
FP_INT_DOWNWARD
FP_INT_TOWARDZERO
FP_INT_TONEARESTFROMZERO
FP_INT_TONEAREST
FP_FAST_FMA*MS*
FP_FAST_FADD*MS*
FP_FAST_DADD*MS*
FP_FAST_D*M***ADD***MS*
  FP_FAST_D*M***XADD***MS*
  FP_FAST_F*M***ADD***MS*
  FP_FAST_F*M***XADD***MS*
FP_FAST_FSUB*MS*
FP_FAST_DSUB*MS*
FP_FAST_D*M***SUB***MS*
  FP_FAST_D*M***XSUB***MS*
  FP_FAST_F*M***SUB***MS*
  FP_FAST_F*M***XSUB***MS*
FP_FAST_FMUL*MS*
FP_FAST_DMUL*MS*
FP_FAST_D*M***MUL***MS*
  FP_FAST_D*M***XMUL***MS*
  FP_FAST_F*M***MUL***MS*
  FP_FAST_F*M***XMUL***MS*
FP_FAST_FDIV*MS*
FP_FAST_DDIV*MS*
FP_FAST_D*M***DIV***MS*
  FP_FAST_D*M***XDIV***MS*
  FP_FAST_F*M***DIV***MS*
  FP_FAST_F*M***XDIV***MS*
FP_FAST_FFMA*MS*
FP_FAST_DFMA*MS*
FP_FAST_D*M***FMA***MS*
  FP_FAST_D*M***XFMA***MS*
  FP_FAST_F*M***FMA***MS*
  FP_FAST_F*M***XFMA***MS*
FP_FAST_FSQRT*MS*
FP_FAST_DSQRT*MS*

```
FP_FAST_DMSQRTMS
  FP_FAST_DMXSQRTMS
  FP_FAST_FMSQRTMS
  FP_FAST_FMXSQRTMS
FP_ILOGB0
FP_ILOGBNAN
FP_LLOGB0
FP_LLOGBNAN
MATH_ERRNO
MATH_ERREXCEPT
math_errhandling

#pragma STDC FP_CONTRACT on-off-switch
int fpclassify(real-floating x);
int iscanonical(real-floating x);
int isfinite(real-floating x);
int isinf(real-floating x);
int isnan(real-floating x);
int isnormal(real-floating x);
int signbit(real-floating x);
int issignaling(real-floating x);
int issubnormal(real-floating x);
int iszero(real-floating x);
Type acosFS(Type x);
Type asinFS(Type x);
Type atanFS(Type x);
Type atan2FS(Type x, Type y);
Type cosFS(Type x);
Type sinFS(Type x);
Type tanFS(Type x);
Type acospiFS(Type x);
Type asinpiFS(Type x);
Type atanpiFS(Type x);
Type atan2piFS(Type x, Type y);
Type cospiFS(Type x);
Type sinpiFS(Type x);
Type tanpiFS(Type x);
Type acoshFS(Type x);
Type asinhFS(Type x);
Type atanhFS(Type x);
Type coshFS(Type x);
Type sinhFS(Type x);
Type tanhFS(Type x);
Type expFS(Type x);
Type exp10FS(Type x);
Type exp10m1FS(Type x);
Type exp2FS(Type x);
Type exp2m1FS(Type x);
Type expm1FS(Type x);
Type frexpFS(Type value, int *y);
int ilogbFS(Type x);
Type ldexpFS(Type x, int p);
long int llogbFS(Type x);
Type logFS(Type x);
Type log10FS(Type x);
Type log10p1FS(Type x);
Type log1pFS(Type x);
Type logp1FS(Type x);
Type log2FS(Type x);
Type log2p1FS(Type x);
Type logbFS(Type x);
Type modfFS(Type value, Type *iptr);
Type scalbnFS(Type x, int n);
Type scalblnFS(Type x, long int n);
Type cbrtFS(Type x);
Type compoundnFS(Type x, long long int n);
Type fabsFS(Type x);
Type hypotFS(Type x, Type y);
Type powFS(Type x, Type y);
Type pownFS(Type x, long long int n);
Type powrFS(Type x, Type y);
Type rootnFS(Type x, long long int n);
Type rsqrtFS(Type x);
Type sqrtFS(Type x);
Type erfFS(Type x);
Type erfcFS(Type x);
Type lgammaFS(Type x);
Type tgammaFS(Type x);
Type ceilFS(Type x);
Type floorFS(Type x);
Type nearbyintFS(Type x);
Type rintFS(Type x);
long int lrintFS(Type x);
long long int llrintFS(Type x);
Type roundFS(Type x);
long int lroundFS(Type x);
long long int llroundFS(Type x);
Type roundevenFS(Type x);
Type truncFS(Type x);
Type fromfpFS(Type x, int round, unsigned int width);
Type ufromfpFS(Type x, int round, unsigned int width);
Type fromfpxFS(Type x, int round, unsigned int width);
Type ufromfpxFS(Type x, int round, unsigned int width);
Type fmodFS(Type x, Type y);
Type remainderFS(Type x, Type y);
StdType remquoFS(StdType x, StdType y, int *quo);
Type copysignFS(Type x, Type y);
Type nanFS(const char *tagp);
Type nextafterFS(Type x, Type y);
Type nexttowardFS(Type x, Type y);
Type nextupFS(Type x);
Type nextdownFS(Type x);
int canonicalizeFS(Type *cx, const Type *x);
Type fdimFS(Type x, Type y);
Type fmaxFS(Type x, Type y);
Type fminFS(Type x, Type y);
Type fmaxmagFS(Type x, Type y);
Type fminmagFS(Type x, Type y);
Type fmaFS(Type x, Type y, Type z);
float faddFS(StdType x, StdType y);
double daddFS(StdType x, StdType y);
  _FloatM fMaddFS(BinType x, BinType y);
  _FloatMx fMxaddFS(BinType x, BinType y);
_DecimalM dMaddFS(DecType x, DecType y);
_DecimalMx dMxaddFS(DecType x, DecType y);
float fsubFS(StdType x, StdType y);
double dsubFS(StdType x, StdType y);
  _FloatM fMsubFS(BinType x, BinType y);
  _FloatMx fMxsubFS(BinType x, BinType y);
_DecimalM dMsubFS(DecType x, DecType y);
_DecimalMx dMxsubFS(DecType x, DecType y);
float fmulFS(StdType x, StdType y);
double dmulFS(StdType x, StdType y);
  _FloatM fMmulFS(BinType x, BinType y);
  _FloatMx fMxmulFS(BinType x, BinType y);
_DecimalM dMmulFS(DecType x, DecType y);
_DecimalMx dMxmulFS(DecType x, DecType y);
float fdivFS(StdType x, StdType y);
double ddivFS(StdType x, StdType y);
  _FloatM fMdivFS(BinType x, BinType y);
  _FloatMx fMxdivFS(BinType x, BinType y);
_DecimalM dMdivFS(DecType x, DecType y);
_DecimalMx dMxdivFS(DecType x, DecType y);
float ffmaFS(StdType x, StdType y, StdType z);
double dfmaFS(StdType x, StdType y, StdType z);
  _FloatM fMfmaFS(BinType x, BinType y, BinType z);
  _FloatMx fMxfmaFS(BinType x, BinType y, BinType z);
_DecimalM dMfmaFS(DecType x, DecType y, DecType z);
_DecimalMx dMxfmaFS(DecType x, DecType y, DecType z);
float fsqrtFS(StdType x);
double dsqrtFS(StdType x);
```

```
   _FloatM fMsqrtFS(BinType x);
   _FloatMx fMxsqrtFS(BinType x);
_DecimalM dMsqrtFS(DecType x);
_DecimalMx dMxsqrtFS(DecType x);
int isgreater(real-floating x, real-floating y);
int isgreaterequal(real-floating x, real-floating y);
int isless(real-floating x, real-floating y);
int islessequal(real-floating x, real-floating y);
int islessgreater(real-floating x, real-floating y);
int isunordered(real-floating x, real-floating y);
int iseqsig(real-floating x, real-floating y);
DecType quantizeFS (DecType x, DecType y);
_Bool samequantumFS (DecType x, DecType y);
DecType quantumFS (DecType x);
long long int llquantexpFS(DecType x);
void encodedecdN(unsigned char encptr[restrict static N/8],
      const _DecimalN * restrict xptr);
void encodebindN(unsigned char encptr[restrict static N/8],
      const _DecimalN * restrict xptr);
void decodedecdN(_DecimalN * restrict xptr,
      const unsigned char encptr[restrict static N/8]);
void decodebindN(_DecimalN * restrict xptr,
      const unsigned char encptr[restrict static N/8]);
int totalorderFS(const Type *x, const Type *y);
int totalordermagFS(const Type *x, const Type *y);
Type getpayloadFS(const Type *x);
int setpayloadFS(Type *res, Type pl);
int setpayloadsigFS(Type *res, Type pl);
   void encodefN(unsigned char encptr[restrict static N/8],
      const _FloatN * restrict xptr);
   void decodefN(_FloatN * restrict xptr,
      const unsigned char encptr[restrict static N/8]);
   void fMencfN(unsigned char encMptr[restrict static M/8],
      const unsigned char encNptr[restrict static N/8]);
   void dMencdecdN(unsigned char encMptr[restrict static M/8],
      const unsigned char encNptr[restrict static N/8]);
   void dMencbindN(unsigned char encMptr[restrict static M/8],
      const unsigned char encNptr[restrict static N/8]);
```