Proposal for TS 18661 update
WG14 N2273

Title:	Min-max functions
Author, affiliation:	C FP group
Date:	2018-06-30
Proposal category:	New features
Target audience:	IEEE 754-201x, floating-point computation, data management

The 2018 revision of IEEE 754 replaces the minNum and maxNum operations, which are supported by the **`fmin`** and **`fmax`** functions in C, with new min-max operations. The IEEE 754 committee took this unusual step because it believed the minNum and maxNum operations were seriously flawed and problematic for their intended use. The new min-max operations will be optional in IEEE 754-2018, but only because the 2018 revision is avoiding additional requirements. The next revision is expected to make the new operations mandatory.

This is a proposal for updating TS 18661-1, TS 18661-2, and TS 18661-3 to support the new min-max operations in IEEE 754-2018 as **`<math.h>`** functions. Also, given the 754 thinking about the minNum and maxNum operations supported by **`fmin`** and **`fmax`**, this proposal removes the **`fminmag`** and **`fmaxmag`** functions from TS 18661.

In TS 18661-1, clause 8, in the operation binding table for F.3, replace:

minNum	**`fmin`**	7.12.12.3, F.10.9.3
maxNum	**`fmax`**	7.12.12.2, F.10.9.2
minNumMag	**`fminmag`**	7.12.12.5, F.10.9.5
maxNumMag	**`fmaxmag`**	7.12.12.4, F.10.9.4

with:

	`fmax`	7.12.12.2, F.10.9.2
	`fmin`	7.12.12.3, F.10.9.3
maximum	**`fmaximum`**	7.12.12.4, F.10.9.4
minimum	**`fminimum`**	7.12.12.5, F.10.9.4
maximumMagnitude	**`fmaximum_mag`**	7.12.12.6, F.10.9.4
minimumMagnitude	**`fminimum_mag`**	7.12.12.7, F.10.9.4
maximumNumber	**`fmaximum_num`**	7.12.12.8, F.10.9.5
minimumNumber	**`fminimum_num`**	7.12.12.9, F.10.9.5
maximumMagnitudeNumber	**`fmaximum_mag_num`**	7.12.12.10, F.10.9.5
minimumMagnitudeNumber	**`fminimum_mag_num`**	7.12.12.11, F.10.9.5

In TS 18661-1, replace 14.3 with:

14.3 Maximum and minimum functions

The functions in this subclause support IEC 60559 operations for determining maximum and minimum values.

Changes to C11:

After 7.12.12.3, add:

7.12.12.4 The `fmaximum` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum(double x, double y);
float fmaximumf(float x, float y);
long double fmaximuml(long double x,
      long double y);
_FloatN fmaximumfN(_FloatN x, _FloatN y);
_FloatNx fmaximumfNx(_FloatNx x, _FloatNx y);
_DecimalN fmaximumdN(_DecimalN x, _DecimalN y);
_DecimalNx fmaximumdNx(_DecimalNx x, _DecimalNx y);
```

Description

[2] The **`fmaximum`** functions determine the maximum value of their arguments. For these functions, +0 is considered greater than −0. These functions differ from the **`fmaximum_num`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fmaximum`** functions return the maximum value of their arguments.

7.12.12.5 The `fminimum` functions

Synopsis

[1]
```
#include <math.h>
double fminimum(double x, double y);
float fminimumf(float x, float y);
long double fminimuml(long double x,
      long double y);
_FloatN fminimumfN(_FloatN x, _FloatN y);
_FloatNx fminimumfNx(_FloatNx x, _FloatNx y);
_DecimalN fminimumdN(_DecimalN x, _DecimalN y);
_DecimalNx fminimumdNx(_DecimalNx x, _DecimalNx y);
```

Description

[2] The **fminimum** functions determine the minimum value of their arguments. For these functions, −0 is considered less than +0. These functions differ from the **fminimum_num** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum** functions return the minimum value of their arguments.

7.12.12.6 The `fmaximum_mag` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_mag(double x, double y);
float fmaximum_magf(float x, float y);
long double fmaximum_magl(long double x,
      long double y);
_FloatN fmaximum_magfN(_FloatN x, _FloatN y);
_FloatNx fmaximum_magfNx(_FloatNx x, _FloatNx y);
_DecimalN fmaximum_magdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fmaximum_magdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **fmaximum_mag** functions determine the value of the argument of maximum magnitude: **x** if |**x**| > |**y**|, **y** if |**y**| > |**x**|, and **fmaximum(x, y)** otherwise. These functions differ from the **fmaximum_mag_num** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fmaximum_mag** functions return the value of the argument of maximum magnitude.

7.12.12.7 The fminimum_mag functions

Synopsis

[1]
```
#include <math.h>
double fminimum_mag(double x, double y);
float fminimum_magf(float x, float y);
long double fminimum_magl(long double x,
      long double y);
_FloatN fminimum_magfN(_FloatN x, _FloatN y);
_FloatNx fminimum_magfNx(_FloatNx x, _FloatNx y);
_DecimalN fminimum_magdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fminimum_magdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **fminimum_mag** functions determine the value of the argument of minimum magnitude: **x** if |**x**| < |**y**|, **y** if |**y**| < |**x**|, and **fminimum(x, y)** otherwise. These functions differ from the **fminimum_mag_num** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum_mag** functions return the value of the argument of minimum magnitude.

7.12.12.8 The fmaximum_num functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_num(double x, double y);
float fmaximum_numf(float x, float y);
long double fmaximum_numl(long double x,
      long double y);
_FloatN fmaximum_numfN(_FloatN x, _FloatN y);
_FloatNx fmaximum_numfNx(_FloatNx x, _FloatNx y);
_DecimalN fmaximum_numdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fmaximum_numdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **fmaximum_num** functions determine the maximum value of their numeric arguments. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **fmaximum** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fmaximum_num** functions return the maximum value of their numeric arguments.

7.12.12.9 The fminimum_num functions

Synopsis

[1]
```
#include <math.h>
double fminimum_num(double x, double y);
float fminimum_numf(float x, float y);
long double fminimum_numl(long double x,
      long double y);
_FloatN fminimum_numfN(_FloatN x, _FloatN y);
_FloatNx fminimum_numfNx(_FloatNx x, _FloatNx y);
_DecimalN fminimum_numdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fminimum_numdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **fminimum_num** functions determine the minimum value of their numeric arguments. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **fminimum** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum_num** functions return the minimum value of their numeric arguments.

7.12.12.10 The `fmaximum_mag_num` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_mag_num(double x, double y);
float fmaximum_mag_numf(float x, float y);
long double fmaximum_mag_numl(long double x,
      long double y);
_FloatN fmaximum_mag_numfN(_FloatN x, _FloatN y);
_FloatNx fmaximum_mag_numfNx(_FloatNx x,
      _FloatNx y);
_DecimalN fmaximum_mag_numdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fmaximum_mag_numdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **`fmaximum_mag_num`** functions determine the value of a numeric argument of maximum magnitude. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **`fmaximum_mag`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fmaximum_mag_num`** functions return the value of a numeric argument of maximum magnitude.

7.12.12.11 The `fminimum_mag_num` functions

Synopsis

[1]
```
#include <math.h>
double fminimum_mag_num(double x, double y);
float fminimum_mag_numf(float x, float y);
long double fmaximum_mag_numl(long double x,
      long double y);
_FloatN fminimum_mag_numfN(_FloatN x, _FloatN y);
_FloatNx fminimum_mag_numfNx(_FloatNx x,
      _FloatNx y);
_DecimalN fminimum_mag_numdN(_DecimalN x,
      _DecimalN y);
_DecimalNx fminimum_mag_numdNx(_DecimalNx x,
      _DecimalNx y);
```

Description

[2] The **fminimum_mag_num** functions determine the value of a numeric argument of minimum magnitude. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **fminimum_mag** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum_mag_num** functions return the value of a numeric argument of minimum magnitude.

NOTE The **fmax** and **fmin** functions are similar to the **fmaximum_num** and **fminimum_num** functions, though may differ in which signed zero is returned when the arguments are differently signed zeros and in their treatment of signaling NaNs (see F.10.9.5).

After F.10.9.3 add:

F.10.9.4 The fmaximum, fminimum, fmaximum_mag, and fminimum_mag functions

These functions treat NaNs like other functions in **<math.h>** (see F.10). They differ from the corresponding **fmaximum_num**, **fminimum_num**, **fmaximum_mag_num**, and **fminimum_mag_num** functions only in their treatment of NaNs.

F.10.9.5 The fmaximum_num, fminimum_num, fmaximum_mag_num, and fminimum_mag_num functions

These functions return the number if one argument is a number and the other is a quiet or signaling NaN. If both arguments are NaNs, a quiet NaN is returned. If an argument is a signaling NaN, the “invalid” floating-point exception is raised (even though the function returns the number when the other argument is a number).

In TS 18661-1, clause 12, in the change to F.2.1, change:

[4] Any operator or **<math.h>** function that raises an "invalid" floating-point exception, if delivering a floating type result, shall return a quiet NaN.

to:

[4] Any operator or **<math.h>** function that raises an "invalid" floating-point exception, if delivering a floating type result, shall return a quiet NaN, unless explicitly specified otherwise.

In TS 18661-1, clause 16, change the replacement text:

> In 7.25#5, include in the list of type-generic macros: **roundeven**, **nextup**, **nextdown**, **fminmag**, **fmaxmag**, **llogb**, **fromfp**, **ufromfp**, **fromfpx**, **ufromfpx**, **totalorder**, and **totalordermag**.

to:

> In 7.25#5, include in the list of type-generic macros: **roundeven**, **nextup**, **nextdown**, **fmaximum**, **fminimum**, **fmaximum_num**, **fminimum_num**, **fmaximum_mag**, **fminimum_mag**, **fmaximum_mag_num**, **fminimum_mag_num**, **llogb**, **fromfp**, **ufromfp**, **fromfpx**, **ufromfpx**, **totalorder**, and **totalordermag**.

In TS 18661-2, clause 7, in the table "Preferred quantum exponents" in 5.2.4.2.2a#7, in the row for **fmin**, **fmax**, etc., replace:

> **fminmag**, **fmaxmag**

with:

> **fmaximum**, **fminimum**, **fmaximum_num**, **fminimum_num**, **fmaximum_mag**, **fminimum_mag**, **fmaximum_mag_num**, **fminimum_mag_num**

In TS 18661-3, subclause 12.3, in the change to 7.12.12, change:

```
_FloatN fmaxmagfN(_FloatN x, _FloatN y);
_FloatNx fmaxmagfNx(_FloatNx x, _FloatNx y);
_DecimalN fmaxmagdN(_DecimalN x, _DecimalN y);
_DecimalNx fmaxmagdNx(_DecimalNx x, _DecimalNx y);

_FloatN fminmagfN(_FloatN x, _FloatN y);
_FloatNx fminmagfNx(_FloatNx x, _FloatNx y);
_DecimalN fminmagdN(_DecimalN x, _DecimalN y);
_DecimalNx fminmagdNx(_DecimalNx x, _DecimalNx y);
```

to:

```
_FloatN fmaximumfN(_FloatN x, _FloatN y);
_FloatNx fmaximumfNx(_FloatNx x, _FloatNx y);
_DecimalN fmaximumdN(_DecimalN x, _DecimalN y);
_DecimalNx fmaximumdNx(_DecimalNx x, _DecimalNx y);

_FloatN fminimumfN(_FloatN x, _FloatN y);
_FloatNx fminimumfNx(_FloatNx x, _FloatNx y);
_DecimalN fminimumdN(_DecimalN x, _DecimalN y);
_DecimalNx fminimumdNx(_DecimalNx x, _DecimalNx y);
```

```
_FloatN fmaximum_magfN(_FloatN x,_FloatN y);
_FloatNx fmaximum_magfNx(_FloatNx x,_FloatNx y);
_DecimalN fmaximum_magdN(_DecimalN x,_DecimalN y);
_DecimalNx fmaximum_magdNx(_DecimalNx x,
    _DecimalNx y);

_FloatN fminimum_magfN(_FloatN x,_FloatN y);
_FloatNx fminimum_magfNx(_FloatNx x,_FloatNx y);
_DecimalN fminimum_magdN(_DecimalN x,_DecimalN y);
_DecimalNx fminimum_magdNx(_DecimalNx x,
    _DecimalNx y);

_FloatN fmaximum_numfN(_FloatN x,_FloatN y);
_FloatNx fmaximum_numfNx(_FloatNx x,_FloatNx y);
_DecimalN fmaximum_numdN(_DecimalN x,_DecimalN y);
_DecimalNx fmaximum_numdNx(_DecimalNx x,
    _DecimalNx y);

_FloatN fminimum_numfN(_FloatN x,_FloatN y);
_FloatNx fminimum_numfNx(_FloatNx x,_FloatNx y);
_DecimalN fminimum_numdN(_DecimalN x,_DecimalN y);
_DecimalNx fminimum_numdNx(_DecimalNx x,
    _DecimalNx y);

_FloatN fmaximum_num_magfN(_FloatN x,_FloatN y);
_FloatNx fmaximum_num_magfNx(_FloatNx x,_FloatNx y);
_DecimalN fmaximum_num_magdN(_DecimalN x,
    _DecimalN y);
_DecimalNx fmaximum_num_magdNx(_DecimalNx x,
    _DecimalNx y);

_FloatN fminimum_num_magfN(_FloatN x,_FloatN y);
_FloatNx fminimum_num_magfNx(_FloatNx x,_FloatNx y);
_DecimalN fminimum_num_magdN(_DecimalN x,
    _DecimalN y);
_DecimalNx fminimum_num_magdNx(_DecimalNx x,
    _DecimalNx y);
```

Also needed, in 5.3 of TS 18661-1, TS 18661-2, and TS 18661-3, in the lists of identifiers to be conditionally added to the standard headers in 7.12 and 7.25, are changes to replace **fmaxmag** and **fminmag** and their suffixed forms with **fmaximum**, **fminimum**, **fmaximum_num**, **fminimum_num**, **fmaximum_mag**, **fminimum_mag**, **fmaximum_mag_num**, and **fminimum_mag_num**, similarly suffixed.