FLOATING-POINT PROPOSALS FOR C2X

N2140
WG 14 - Markham
April 3-6, 2017

C FP group

FP proposals for C2x

- IEC 60559 is intended for a wide range of applications. Not all its features are suitable for all languages or implementations – hence some features are optional in IEC 60559
- Goal here …
  - **Summarize C support for optional features of IEC 60559 as specified in ISO/IEC TS 18661-3,4,5**
  - **Decide what should be further considered for C2x**
- TS 18661 proposals are for conditional (optional) features in C2x
- All parts of TS 18661 provide detailed changes to C11

CFP proposals for C2x

n2117 - TS 18661-3 - interchange and extended types

n2118 - TS 18661-4a - mathematical functions

n2119 - TS 18661-4b - reduction functions

n2120 - TS 18661-5a - evaluation format pragmas

n2121 - TS 18661-5b - optimization control pragmas

n2122 - TS 18661-5c - reproducible results

n2123 - TS 18661-5d - alternate exception handling

n2124 - rounding direction macro FE_TONEARESTFROMZERO

n2128 - Default rounding mode

TS 18661-3

n2117

Types and functions to support
IEC 60559 interchange and extended formats

IEC 60559 interchange formats

- IEC 60559:2011 specifies a “tower” of *interchange* formats
- Arbitrarily large wdiths (32x)
- For binary and decimal
- Balanced precision and range determined by width
- For exchange of FP data
- binary16, for GPU data, etc.
- Formats may be supported as
  - Arithmetic – with all standard operations
  - Non-arithmetic – with conversion operations

IEC 60559 extended formats

- IEEE specifies *extended* formats that extend its basic formats: binary32|64|128 and decimal64|128
- Have at least a specified precision and range
- For explicit wide evaluation
- Not for data exchange

TS 18661-3

- Three features
  - Interchange floating types
  - Extended floating types
  - Support for non-arithmetic interchange formats
- Full language and library support for interchange and extended floating types
- Conversion operations for non-arithmetic interchange formats represented in unsigned char arrays

TS 18661-3 – type structure extensions

interchange floating types: _Float*N*, _Decimal*N*
extended floating types: _Float*N*x, _Decimal*N*x

real floating types

- standard floating types: float, double, long double
- binary floating types: _Float*N*, _Float*N*x
- decimal floating types: _Decimal*N*, _Decimal*N*x

complex types

- float _Complex, double _Complex, long double _Complex
- _Float*N* _Complex, _Float*N*x _Complex

Imaginary types

- float _Imaginary, double _Imaginary, long double _Imaginary
- _Float*N* _Imaginary, _Float*N*x _Imaginary

TS 18661-3 – type structure unchanged

floating types

- real floating types
- complex types
- imaginary types

real types

- integer types
- real floating types

arithmetic types

- integer types
- floating types

TS 18661-3

- Standard binding for extension floating types with IEC 60559 formats, which are common extensions (e.g., float16, float128, float80)
- Facilitates exchange of FP data, without full support type
- Enables explicit wide evaluation, for robustness

TS 18661-4a

n2118

Functions to support

IEC 60559 mathematical operations

TS 18661-4a mathematical functions

- IEC 60559:2008 specifies a set of optional mathematical operations
- Many of these are already supported as <math.h> functions
- TS 18661-4 adds functions for the rest
- Does not require IEC 60559-specified correct rounding
- Names with cr prefixes reserved for correctly rounded verisons, e.g., crsin for correctly rounded sin function

TS 18661-4a mathematical functions

$\mathrm{asinpi}(x) = \arcsin(x) / \pi$

$\mathrm{acospi}(x) = \arccos(x) / \pi$

$\mathrm{atanpi}(x) = \arctan(x) / \pi$

$\mathrm{atan2pi}(y, x) = \arctan(y/x) / \pi$

$\mathrm{sinpi}(x) = \sin(\pi \times x)$

$\mathrm{cospi}(x) = \cos(\pi \times x)$

$\mathrm{tanpi}(x) = \tan(\pi \times x)$

$\mathrm{exp10}(x) = 10^x$

$\mathrm{exp2m1}(x) = 2^x - 1$

$\mathrm{exp10m1} = 10^x – 1$

TS 18661-4a mathematical functions

logp1(x) = $\log_e(x + 1)$

log2p1(x) = $\log_2(x + 1)$

log10p1(x) = $\log_{10}(x + 1)$

rsqrt(x) = $1/\sqrt{x}$

compound(x, n) = $(1 + x)^{n,}$, for int n

rootn(x, n) = $x^{1/n}$, for int n

pown(x, n) = x^n , for int n

powr(x, y) = x^y as $e^{y \times \ln(x)}$, for x in $[0, +\infty]$

TS 18661-4a mathematical functions

- Complete the set of exponential and logarithm functions for bases 2 and 10
- Include trigonometric functions based on units of pi
- Include commonly needed functions involving power and square root operations
- Supported entirely in <math.h> and <tgmath>

TS 18661-4b

n2119

Functions to support

IEC 60559 reduction operations

TS 18661-4b reduction functions

- IEC 60559:2008 specifies a set of optional reduction operations
- TS 18661-4 supports them as <math.h> functions

TS 18661-4b – sum reductions

Sum reduction functions on vectors p and q of length n

double reduc_sum(size_t n, const double p[static n]);
computes $\Sigma_{i=0,n-1} p_i$

reduc_sumabs computes $\Sigma_{i=0,n-1}|p_i|$

reduc_sumsq compute $\Sigma_{i=0,n-1} p_i^2$

reduc_sumprod computes $\Sigma_{i=0,n-1} p_i \times q_i$

TS 18661-4b – scaled product reductions

Scaled product reduction functions on vectors p and q of length n

```
double scaled_prod(size_t n,
            const double p[static restrict n],
            intmax_t * restrict sfptr);
```

computes product pr of the n members of array p and scale factor sf, such that $pr \times b^{sf} = \Pi_{i=0,n-1} p[i]$, where b is the radix of the type

scaled_prodsum computes pr and sf, such that

$pr \times b^{sf} = \Pi_{i=0,n-1}(p[i] + q[i])$

scaled_proddiff computes pr and sf, such that

$pr \times b^{sf} = \Pi_{i=0,n-1}(p[i] - q[i])$

TS 18661-4b reduction functions

- Reductions are among the most widely used numerical computations
- Allow implementations to take advantage of platform-specific performance features to compute reductions
- Avoid intermediate overflow and underflow
- The scaled product functions can avoid overflow and underflow where the scaled product itself is an intermediate computation
- Supported entirely in <math.h>

TS 18661-5a

n2120

Evaluation format pragmas to support IEC 60559 preferredWidth attributes

TS 18661-5a evaluation format pragmas

- IEC 60559:2008 recommends preferredWidth attributes for users to specify the format for evaluating expressions, at a block level
- TS 18661-5 supports them as evaluation format pragmas in <fenv.h>
- Form and scope like other floating-point pragmas in C11
- Allow user tradeoffs for precision, performance, or reproducibility

TS 18661-5a evaluation format pragmas

- #pragma STDC FENV_FLT_EVAL_METHOD *width*
  for standard and binary types
- *width* reflects a possible value of FLT_EVAL_METHOD macro (which characterizes default evaluation)
- Required support for *width* values -1, 0, and DEFAULT
- Other *width* values may be supported
- Similar FENV_DEC_EVAL_METHOD for decimal types
- Required support for decimal *width* values -1, 1, and DEFAULT

TS 18661-5b

n2121

Pragmas to support

IEC 60559 optimization attributes

TS 18661-5b optimization pragmas

- IEC 60559:2008 recommends attributes for users to allow or disallow certain value-changing optimizations
- TS 18661-5 supports these attributes as optimization pragmas in <fenv.h>
- Form and scope like other floating-point pragmas in C11
- Pragmas allow but do not require the optimizations
- Enable user to tradeoff predictability and performance

TS 18661-5b optimization pragmas

Allow/disallow value-changing optimizations (transformations)

#pragma STDC FENV_ALLOW_... *on-off-switch*

where … is one of

- VALUE_CHANGING_OPTIMIZATION allows all the following, which can also be allowed separately
- ASSOCIATIVE_LAW
- DISTRIBUTIVE_LAW
- MULTIPLY_BY_RECIPROCAL

  A / B = A x (1/B)

TS 18661-5b optimization pragmas

- ZERO_SUBNORMAL

  allow replacing subnormal operands and results with 0
- CONTRACT_FMA

  contract (compute with just one rounding) A x B + C
- CONTRACT_OPERATION_CONVERSION

  e.g., F = D1 * D2 and F = sqrt(D)
- CONTRACT

  all contractions

  equivalent to FP_CONTRACT pragma in <math.h>

TS 18661-5c

n2122

Pragma to support

IEC 60559 reproducible-results attribute

TS 18661-5c reproducible results

- IEC 60559:2008 recommends an attribute for users to request results that are reproducible on all supporting implementations
- TS 18661-5 supports this attribute with a pragma in <fenv.h> and with guidelines for reproducible code
- Form and scope like other floating-point pragmas in C11
- #pragma FENV_REPRODUCIBLE *on-off-default*

FENV_ACCES	“on”
FENV_ALLOW_VALUE_CHANGING_OPTIMIZATION	“off”
FENV_FLT_EVAL_METHOD	0
FENV_DEC_EVAL_METHOD	1

TS 18661-5c reproducibility

Rules for reproducible code include

- Code translates into a sequence of IEC 60559 operations
- Use FENV_REPRODUCIBLE pragma
- Limit use of FP pragmas to reproducible states
- Do not use long double, extended floating, complex, or imaginary types
- Use part 3 interchange formats only among supporting implementations

TS 18661-5d

n2123

Pragma to support
IEC 60559 alternate exception handling

TS 18661-5d alternate exception handling

- IEC 60559 default exception handling
    - set exception flag(s)
    - return prescribed value
    - continue execution
- IEC 60559:2008 recommends attributes for users to specify alternate (non-default) methods for handling floating-point exceptions
- Intended to let users deal with exceptions without having to know the details
- TS 18661-5 supports these attributes with a pragma in <fenv.h>

TS 18661-5d alternate exception handling

#pragma STDC FENV_EXCEPT *except-list action*

except-list a comma-separated list of

exception macro names:

FE_DIVBYZERO, FE_INVALID, …

and FE_ALL_EXCEPT

and optional sub-exception designations:

FE_INVALID_ADD	inf - inf
FE_INVALID_MUL	inf * 0
FE_INVALID_SNAN	signaling NaN operand
FE_DIVBYZERO_LOG	log(0)

etc.

TS 18661-5d alternate exception handling

action one of

- DEFAULT

  IEC 60559 default handling

- NOEXCEPT

  like default but no flags set

- OPTEXCEPT

  like default but flags may be set

- ABRUPT

  only for “underflow”, IEC 60559-defined abrupt underflow shall occur, unlike ALLOW_ZERO_SUBNORMAL where zeroing may occur

TS 18661-5d alternate exception handling

The following change flow of control

action one of (cont.)

- BREAK

  terminate compound statement associated with pragma, ASAP*

  *ASAP – for performance, the objects, flags, dynamic modes, and library states that would be changed at any point if the compound statement ran to completion are indeterminate or unspecified

TS 18661-5d alternate exception handling

action one of (cont.)

These work together

- TRY

  A designated exception may be handled (ASAP) by a compound statement associated with a CATCH action

- CATCH

  Code to handle designated exceptions

TS 18661-5d alternate exception handling

action one of (cont.)

These work together

- DELAYED_TRY

  After associated compound statement completes, a designated exception may be handled by a compound statement associated with a DELAYED_CATCH action.

- DELAYED_CATCH

  Code to handle designated exceptions

TS 18661-5d alternate exception handling

```
double d[n]; float f[n];
…
#pragma STDC FENV_EXCEPT TRY FE_DIVBYZERO,  FE_OVERFLOW
{

        for (i=0; i<n; i++) {
                f[i] = 1.0 / d[i];
        }
}
#pragma STDC FENV_EXCEPT CATCH FE_DIVBYZERO
{
        printf(“divide-by-zero\n”); }
}
#pragma STDC FENV_EXCEPT CATCH FE_OVERFLOW
{
        printf(“overflow\n”);
}
```

Rounding direction macro FE_TONEARESTAWAY

n2124

Macro to support

IEC 60559 optional rounding direction

Rounding direction macro FE_TONEARESTAWAY

- IEC 60559:2008 specifies rounding to nearest with ties away from zero
- The rounding direction is required for decimal, optional for binary FP
- Now in RISC V architecture for binary FP and should be expected to appear in HW
- Proposal supports it with an optional <fenv.h> macro FE_TONEARESTAWAY
- For use with the fegetround and fesetround functions and the FENV_ROUND pragma

Rounding direction macro FE_DEFAULT

n2128

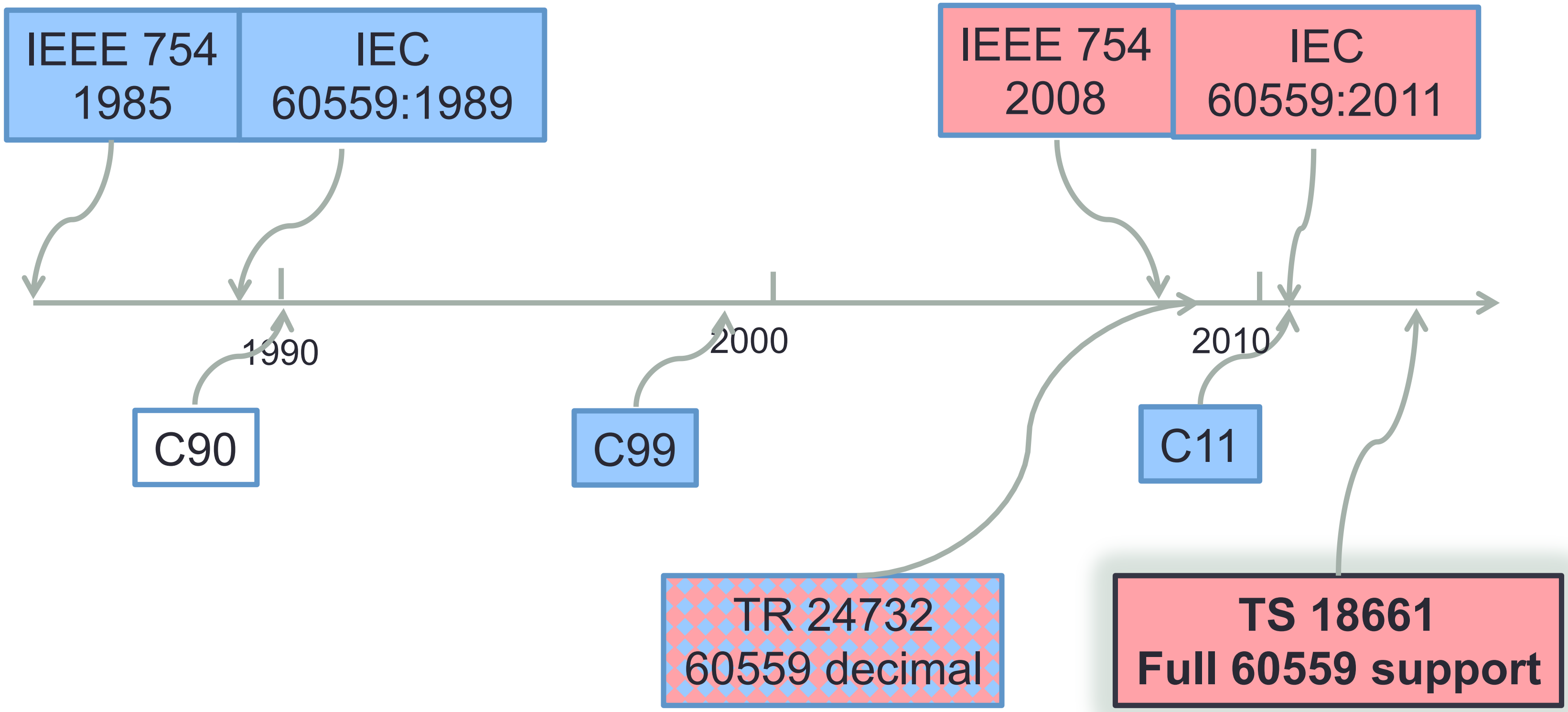
Macro for
default rounding direction

Rounding direction macro FE_DEFAULT

- C11 makes several references to "default rounding”
- There is no symbol for the default rounding direction
- FE_TONEAREST represents the default rounding mode for IEC 60559 implementations, but other implementations may have different defaults (e.g., IBM S/360 hex FP has FE_TOWARDZERO)
- Proposal adds macro FE_DEFAULT in <fenv.h> to represent the implementation’s default rounding direction

About TS 18661 – backup slides

Floating-point and C standards



IEEE 754 1985

IEC 60559:1989

IEEE 754 2008

IEC 60559:2011

1990

2000

2010

C90

C99

C11

TR 24732 60559 decimal

TS 18661 Full 60559 support

Background

Specify a C binding for IEEE 754-2008

- Work began 2009
- Under direction of ISO/IEC JTC1/SC22/WG14 – C
- Expertise in floating-point and language standards, compilers, libraries
- 754 adopted as international standard ISO/IEC/IEEE 60559:2011

Principles

- Support all of the current FP standard, as-is
- Specify as changes to C11
- Use existing C mechanisms, minimize language invention
- Develop specification in parts, to pipeline process
- Supersede TR 24732 (decimal)
- Allow support by free-standing C implementations
- Deliver an ISO/IEC Technical Specification

Status

- In five parts

  Required features in IEC 60559

  1 Binary floating-point arithmetic

  2 Decimal floating-point arithmetic

  Recommended features in IEC 60559

  3 Interchange and extended types

  4 Supplementary functions

  5 Supplementary attributes

- All parts published 2014-2016

Publications

- ISO/IEC TS 18661-1:2014, Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 1: Binary floating-point arithmetic
- ISO/IEC TS 18661-2:2015, Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 2: Decimal floating-point arithmetic
- ISO/IEC TS 18661-3:2015, Information technology — Programming languages, their environments and system software interfaces — Floating-point extensions for C — Part 3: Interchange and extended types
- ISO/IEC TS 18661-4:2015, Information Technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C — Part 4: Supplementary functions
- ISO/IEC TS 18661-5:2016, Information Technology — Programming languages, their environments, and system software interfaces — Floating-point extensions for C — Part 5: Supplementary attributes