WG 14 N1839

WG14 CFP meeting minutes for the meeting of 2014/05/13

9:00 AM PDT / 12:00 PM EDT:

Attendees: Jim, Rajan, Fred, David, Mike, Ian

New agenda items:
None.

Old action items:
Jim: Work with John Benito and David Keaton to get the parts 2, 3 and 4 issued for ballot. - Done
Jim: Part 4: Divide it up and assign (or open to assign) reviewers to parts of the document. - Done
Jim: Part 4: Give the document to Joseph and ask if he has any issues with it. - Done (no issues from Joseph)
David: Part 5: Complete exception specification with the full syntax dealing with scope and sub-exceptions. Include a discussion document with reasons choices and alternatives. - Partially done (more of an outline. Sent on 2014/05/12). Keep open.

Next Meeting:
June 12th (Thursday), 2014, 12:00 EST, 9:00 PDT
Same teleconference number.

New action items:
David: Part 5: (From last meeting): Complete exception specification with the full syntax dealing with scope and sub-exceptions. Include a discussion document with reasons choices and alternatives. - Partially done (more of an outline. Sent on 2014/05/12). Keep open.
David: Part 5: For fast_subnormal, change to "allows abrupt underflow" -> "allows (but not requires) abrupt underflow"
David: Part 5: Put in exception category in the sub-exceptions as the prefix. Ex. FE_DIV_ZERO -> FE_DIVBYZERO_DIVIDES
David: Part 5: Add in FE_INVALID_OTHER and FE_DIVBYZERO_OTHER
David: Part 5: OPTFLAG -> MAY_FLAG, NOFLAG -> NO_FLAG, SUBSTITUTEXOR -> SUBSTITUTE_XOR

Discussion:
The wiki links at the top point to the ballot documents. There are Word versions as well as PDF versions.
Editorial changes are being kept in live documents by Jim for the next version as needed.

Part 1: Sent off for publication.

Part 2: Submitted to ISO for DTS ballot.

Part 3: Submitted to ISO for PDTS ballot.

Part 4: (http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1835.pdf)

Submitted to ISO for PDTS ballot.

Part 5: (Email from David sent 2014/04/14)
  Took parts from C11 and IEEE 754 and parts 1 and 2 to develop the framework/outline sent in the email.
  FENV_WIDTH pragma: none means no preferred width.
    Jim: What is the difference between none and float?
    David: No difference. Generic should be standard (re floating types).
      There does need to be a difference since you can have float64, and float128 only since none would then be float64 and not float32.
    Jim: It is legitimate to add float64 to double. The FLT_EVAL_METHODs and DEC_EVAL_METHOD defines the evaluation method but do not prescribe the evaluation method. The pragma should match up with those macros.
      Perhaps have the pragma act as is if it sets the macro.
    Ian: Is width relevant to anything beyond x86?
    David: Yes, if you want to treat floats as doubles like pre-ANSI C.
    Jim/Ian: The pragmas should affect which lexical block they are associated with.
    What the numbers mean (used for *_EVAL_METHOD) are not obvious.
    Fred: What represents none? 0 is float.
    Jim: The user would have to use 0 or -1 since the EVAL_METHOD macro would have been set to 1 for David's case.
    Fred: How do we define the decimal widths?
    Jim: We'd have to name the pragma's differently (split them up).
    The pragma should change the macro value.
    David: The pragma should not accept -1 as an argument.
    Ian: Does adding in the half precision affect anything?
    Rajan: Covered by the value 16 (from Part 3).
    The pragma names will be FLT_EVAL_METHOD and DEC_EVAL_METHOD now.
  Underflow optimization:
    Abrupt vs value changing
    Jim: Existing pragmas use ON/OFF and do we want to use that vs ENABLE/DISABLE?
    David: Difference between allowing an optimization for example, but not requiring it.
    Jim/Ian: The existing pragmas use ON/OFF but have the same semantics. They are ENABLE/DISABLE.
    David: The distinction should be there.
    Jim: The distinction must be listed in the spec. For the name however we should discuss if we want it to be inconsistent.
    Rajan: Keep consistency if possible with C11. Everyone: Agreed.
    *ToDo: David: For fast_subnormal, change to "allows abrupt underflow" -> "allows (but not requires) abrupt underflow"
    Jim: The value changing optimizations switch is a coarse knob to set them all or off for all?
    David: In general yes, but with some exceptions. A list of optimizations will need to be given.
    The global switch will be for all, and we can have smaller ones for individual optimizations.
  reproducible vs value changing opt (Chapter 10 and 11 of IEEE):
    Reproducible means gives same results. Value changing does not have to have the same results.
    i.e. Disabling value changing optimizations does not necessarily give you reproducible results. Ex. the transcendental functions.
    Jim: Do you see the compiler diagnosing non-reproducible operations?
    David: It could. Ex. no correctly rounded versions of trance dental functions so it is not necessarily reproducible.
    Jim: It could for example replace a normal call with a cr prefixed call.
      Note that reproducible doesn't mean any operation has to be reproducible. We need to

spell out what extent reproducible means (ex. transcendental, order of operations, intermediate values). If it cannot be made so, the implementation should diagnose it.

We can require an implementation to say if it can't support a particular pragma.

We can have a minimal set that everything has to support, and leave the others optional.

Jim: Should FLT_EVAL_METHOD of zero require evaluating to the type?

David: reproducibility would require eval method = 0.

exception handling:

Jim: Can each sub-exception have the group exception macro name as it's base like the current invalid ones?

*ToDo: David: Put in exception category in the sub-exceptions as the prefix. Ex. FE_DIV_ZERO -> FE_DIVBYZERO_DIVIDES

Jim: If you have log sub-exception, why not acos or others?

David: Maybe have one for logb, and one for section 9 functions for both divzero and invalid. Note acos(-2) is invalid sqrt, but not obvious without looking into it. Perhaps say invalid_other that is intended to apply to recommended library functions?

Don't expect the sub-exceptions to be used, but in the cases where it is used it is invaluable.

Perhaps we just leave out the functions?

*ToDo: David: Add in FE_INVALID_OTHER and FE_DIVBYZERO_OTHER

control flow:

record: Used for debuggers.

Jim: We can say what it is intended to be used for, but no behaviour change from a C point of view.

OPTFLAG - Can be read as optimization flag.

*ToDo: David: OPTFLAG -> MAY_FLAG, NOFLAG -> NO_FLAG, SUBSTITUTEXOR -> SUBSTITUTE_XOR

Substitute scope:

Ex. f / g -> f' / g' being done in a loop with the pragma outside it. Note that f' / g' could also be an exception.

goto can be done, with a goto back, but where to go back to?

Ian: The C++ style try/catch can do it.

Rajan: Use something like a function exceptionName(f/g, f'/g') syntax.

Mike: Like the ternary operator in C.

Like an if statement with an empty true case.

ieee_substitute(p/q, FE_INVALID_DIV, p'/q') // Macro

Or

ieee_substitute(p/q, FE_INVALID_DIV) ? p/q : p'/q'

Up the the optimizer to know if the calculation needs to be redone.

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative for Canada