# C9X Floating-Point Proposal—Overview

**WG14/N510 X3J11/95-111 (Draft 12/21/95)**

Jim Thomas
Taligent, Inc.
10201 N. DeAnza Blvd.
Cupertino, CA 95014-2233
jim_thomas@taligent.com

The four papers

Floating-Point Arithmetic—C9X Edits (N511)
Mathematics (Enhanced) <math.h> (N512)
Floating-Point Environment <fenv.h> (N513)
Annex X: IEEE Standard Floating-Point Arithmetic (N514)

collectively constitute a detailed proposal to integrate the "Floating-Point C Extensions" chapter of X3J11's *Numerical C Technical Report* (X3/TR-17:199x) into C9X, with modifications according to directions approved by SC22/WG14.

## Purpose

The proposal addresses two long-standing needs of programmers: (1) more predictable floating-point arithmetic for all implementations, and (2) a standard language binding for features of IEEE arithmetic.

Vagaries of floating-point arithmetic have plagued programmers and users since its inception. And they still do, even though hardware floating-point is now largely standardized. When IEEE binary floating-point standard 754 became an official standard in July 1985, 26 months before the radix-independent standard 854, several IEEE implementations were already shipping. Now virtually all new floating-point implementations conform to the IEEE standards—at least in format, if not to the last detail. Although these standards have been enormously successful in influencing hardware implementation, many of their features, including predictability, remain impractical or unavailable for use by programmers. The IEEE standards do not include language bindings—a cost of delivering the basic standard in a timely fashion. The ANSI C committee attempted to remove conflicts with IEEE arithmetic, but did not specify IEEE support. Expediencies of programming language implementation and optimization can deny the features offered by modern hardware. In the meantime, particular companies have defined their own IEEE language extensions and libraries; not surprisingly, lack of portability has impeded programming for these interfaces.

## History

The Numerical C Extensions Group, NCEG, at its initial meeting in May 1989, identified support for IEEE floating-point arithmetic as one of its focus areas and organized a subgroup to produce a technical report. The ensuing effort benefited from the considerable C language and IEEE floating-point expertise associated with NCEG and X3J11. It included individuals with substantial experience with language extensions (albeit proprietary) for IEEE floating-point. And, following after the standardization of

C, it had a stable, well defined base for its extensions. Thus there has been a unique opportunity to solve this problem.

NCEG/X3J11.1 and X3J11 mailings have included thirteen drafts of FPCE and two drafts of the four papers constituting the specific C9X proposal. NCEG's FP/IEEE subgroup, NCEG, X3J11, and numerous other interested parties have reviewed the drafts to varying degrees. Proprietary extensions for IEEE support have provided prior art for many features. Both developmental and commercial compilers and libraries have implemented the specification.

Major milestones include

| Date | Milestone |
|---|---|
| May 92 | FPCE draft distributed for preliminary review to various professional organizations, including X3J11, WG14, X3J16/WG21, X3T2, and X/Open |
| Jan 93 | FPCE draft distributed for public comment |
| Jun 93 | FPCE draft forwarded by NCEG/X3J11.1 to X3J11 as its FP/IEEE technical report |
| Jun 94 | FPCE draft accepted by X3J11 as part of its technical report on numerical extensions |
| Mar 95 | Final version of FPCE, for X3J11 technical report |
| Aug 95 | FPCE recast for C9X integration (four papers) |

**Substantive changes from FPCE**

Widest-need is no longer one of the defined options for expression evaluation. *Rationale*: Lack of prior art.

In `<float.h>`, the `FLT_EVAL_METHOD` macro now replaces FPCE's `_MIN_EVAL_FORMAT` and `_WIDEST_NEED_EVAL` macros. *Rationale*: Two macros are no longer needed, since widest-need was removed.

The implementation can now set `FLT_EVAL_METHOD` to indicate that its method of expression evaluation is defined by the implementation or indeterminate, rather than one of the standard defined methods. *Rationale*: Eases acceptance while still promoting standard-defined methods.

Macros replace pragmas as the means for uttering translation directives. *Rationale*: Macros are preferred because of general limitations of pragmas and the desire not to require standard pragmas.

Translation directives can now be applied to a compound statement, as well as to the subsequent part of a translation unit. *Rationale*: Avoids unnecessary restrictions on optimization and supports locality in source code.

FPCE's directives for allowing wide representation for function parameters, function returns, and local variables have been dropped. *Rationale*: This FPCE specification is awkward and use of the directives is highly system dependent; the need seems insufficient, given availability of the `float_t` and `double_t` types.

Wide character functions are now covered. *Rationale*: Brings the proposal up-to-date with the current C standard.

The predefined constant `__IEEE_FP__` replaces `__FPCE_IEEE`, and `__FPCE__` has been dropped. *Rationale*: Because the proposal is for changes to the standard and not for extensions, the FPCE name is not appropriate, and `__FPCE__` is not needed.

Matching of translation-time and execution-time conversion of decimal numbers is now recommended instead of required. *Rationale*: This requirement would be too great a burden for some implementations.

Overloading macros replace FPCE's overloading functions. Using `#undef` to remove a macro definition reveals a `double` version of the function, a la C90 `<math.h>`. Otherwise this change is cosmetic. *Rationale*: Macros allow a more natural specification and continue to support C90's unmasking scheme.

The names `fegetexceptflag` and `fesetexceptflag` replace FPCE's `fegetexcept` and `fesetexcept`. *Rationale*: This distinguishes these functions—which deal with the entire content of the flags, take an `fexcept_t*` argument, and are used as a pair—from the other exception functions.

Recommendations about type widths for non-IEEE implementations, and the recommendation for a warning when the recommendations for the widths are not met, have been dropped. *Rationale*: The needed doesn't seem sufficient, now that the diversity of implementations has loosened such expectations about types.

The requirement for a "warning" when a hexadecimal constant cannot be represented exactly has been been downgraded to a recommendation (for a non-fatal diagnostic). *Rationale*: The need does not seem sufficient to introduce warning requirements into the C standard, just for this purpose.

Certain recommendations for operator and function documentation have been dropped. *Rationale*: This will be left to LIA.

Certain specification related to the floating-point environment now covers just IEEE implementations. (This change does not affect the header `<fenv.h>`.) *Rationale*: This allows localizing most of the material about the floating-point environment in `<fenv.h>` and the IEEE annex, which is appropriate since the specification in question is essentially for IEEE implementations anyway.

## Acknowledgments

People who have made especially substantial contributions to this proposal include, in alphabetical order: Jerome Coonen, Bill Gibbons, David Hough, Rex Jaeschke, W. Kahan, Clayton Lewis, Stuart McDonald, Colin McMaster, Rick Meyers, David Prosser, and Fred Tydeman.

Others who have provided invaluable contributions, reviews, or administrative help include, in alphabetical order: Joel Boney, Norris Boyd, Larry Breed, Walter Bright, W. J. Cody, Elizabeth Crockett, Karen Deach, James Demmel, Fred Dunlap, Clive Feather, Yinsun Feng, Samuel Figueroa, Paul Finlayson, James Frankel, Scott Fraser, David Gay, Eric Grosse, Ron Guilmette, Doug Gwyn, Bill Homer, Kenton Hanson, Paul Hilfinger, Martha Jaffe, Bob Jervis, Ed Johnston, David Keaton, Earl Killian, David Knaak, John Kwan, Roger Lawrence, Tom MacDonald, Michael Meissner, Randy Meyer, Randy Meyers, Antonine Mione, Stephen Moshier, Adolfo Nemirovsky, Jon Okada, Conor O'Neill, Tom Pennello, Tim Peters, P.J. Plauger, Thomas Plum, Sanjay Poonen, Pat Ricci, Ali Sazegari, Roger Schlafly, Steve Sommars, Richard Stallman, Linda Stanberry, Gordon Sterling, Bill Torkelson, Douglas Walls, and Terrence Yee.

The author benefited from experience with numerics and languages groups at Apple Computer, Inc., developing the Standard Apple Numerics Environment (SANE) and its various language bindings, and at Taligent, Inc., implementing this specification.