FPCE Features for C9X

N407, X3J11/95-008

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

Introduction

This is a high-level proposal for integrating "Floating-Point C Extensions" (FPCE), Chapter 5 of X3J11's Technical Report, into C9X. It classifies FPCE features as:

I. For inclusion in C9X, as normative
II. For inclusion in C9X, but separable for IEEE implementations
III. Not for inclusion in C9X

Its purpose is to gain X3J11 review and approval for the basic direction. A subsequent proposal will address the specification details to integrate the features.

Most of the features in group I are key to FPCE's main goals of improving predictability and supporting the IEEE floating-point standards. The few others, for example new math functions like acosh, round out basic services for numerical computation.

Features for IEEE support separate into groups I and II. Those that have reasonable implementations on non-IEEE systems and support code and data portability among all systems fall into group I. For example, the `isfinite` macro function reasonably always returns true on a system without infinities, and the availability of the macro on all systems allows programs to exploit infinities on systems that provide them, but still run on systems that don't. It's particularly important that these sorts of FPCE features be normative because they support facilities that are present on the great majority of systems.

Other IEEE-support features amount to direct IEEE binding and have no significance on systems without the pertinent facilities. These fall into group II. One example is the mapping between C types and IEEE types. Another is the specification for how the library functions handle special cases on IEEE systems. This sort of specification has no bearing on code and data portability between IEEE and non-IEEE systems, where the limits are already set by differences in the underlying systems.

This proposal does not address the question of whether group II features should be normative, albeit for IEEE implementations only. Specification for group II features should be in C9X, else accompany it, maybe as an appendix, because they are an essential complement to group I features in providing IEEE support.

Prior art recommends all FPCE features in groups I and II. Those in group III have not yet proven practical and useful enough for inclusion, or are superseded by the following proposals for groups I and II.

323

Group I

These FPCE features should be included in C9X as normative, with specification to follow the Technical Report except where noted.

Floating-point library <fp.h> (§4.3). Enhanced functionality for all implementations. Key for IEEE support and portability. Supersedes `<math.h>`.

<fp.h> overloading (§4.3.1). Facilitates portable efficient code. Programming convenience. Enables better semantics for complex arithmetic, e.g. cos(imaginary) = real. Requires translator support for many implementations. (Fallback: provide only double prototypes and rely on implementations to exploit standard library liberties in dealing with wide arguments and return values.)

Floating-point environment library <fenv.h> (§4.4). Key for IEEE support and portability. Trivial implementation for non-IEEE systems. Entails introducing concept of floating-point environment. Consider renaming `fegetexcept` and `fesetexcept`.

Expression evaluation methods (§2.3.1, 3.2.3). Stricter specification of expression evaluation. Key for predictability for all implementations. Add new macro `FP_EVAL_METHOD` to `<float.h>`, which evaluates to

0	float
1	double
2	long double
3	other

0-2 indicate the FPCE methods without widest need. 3 indicates a method deviating from non-widest-need FPCE methods in some way. One of 0-2 required. (Fallback: 0-2 strongly recommended). Supersedes TR's `_MIN_EVAL_FORMAT` and `_WIDEST_NEED_EVAL` (which existing FPCE implementations could continue to support).

Hexadecimal floating constants (§3.1.2.1, 4.2.1.2, 4.2.2.1). New floating constants, with I/O support. Useful for all implementations. Facilitates predictability.

I/O support for infinities, NaNs, and -0 (§4.2.1.2, 4.2.2.1). Key for IEEE support and code/data portability.

Translation directives. Recast pragmas as macros, retaining basic effects (per direction from X3J11). Include `fenv_access` and `fp_contract`, but not `fp_wide_xxx` directives.

fenv_access macro (§2). Key for IEEE support/optimization. Trivial implementation for non-IEEE systems.

fp_contract macro (§3.2.3.2). Key for predictability/efficiency on systems with fast contracted operators in hardware. Trivial implementation on others.

strtof, strtold, HUGE_VALF, HUGE_VALL (§4.2.1.2, 4.3). Useful for all implementations. `HUGE_VALF`, `HUGE_VALL` fix deficiency in current standard.

Translation-execution consistency (§3.1.2, 4.2.1.2). Key for predictability.

Dynamic FLT_ROUNDS (§2.3.1). Better fit for IEEE semantics, and for any system with rounding modes. No implementation required for other systems.

Constant expressions and initialization (§3.4, 3.5). Key for IEEE support. Useful on any system where constant expressions have side effects. No implementation required for other systems. (Fallback: move to group II.)

FP environment management (§2.1, 2.2). Key for IEEE support. Useful for any system with floating-point modes or flags. No implementation required for other systems. (Fallback: move to group II.)

Optimization guide (§B.2, B.5). Some of the items about expression transformations and the section on wide representation are important for predictability on all systems.

Documentation guidelines (§E). Recommend information for implementors to document. Promotes predictability and facilitates porting for all implementations. Include a brief list of what to document. Don't refer to "forthcoming" guide.

Group II

These FPCE features should be included in C9X, for implementations of IEEE standard floating-point arithmetic. Specification should follow the Technical Report except where noted.

<fp.h> for IEEE implementations (§F). For IEEE compatibility.

C-IEEE type mapping (§3.1.1). Key for IEEE support.

C-IEEE conversion mapping (§3.2.1, 3.2.2). Key for IEEE support.

C-IEEE operation mapping (§3.3.1, 3.3.2). Key for IEEE support.

Correctly rounded binary-decimal conversion (§4.2.1.2, 4.2.2.1). Important for predictability, particularly on IEEE systems where other operations are correctly rounded. (Move to group I?)

Optimization guide (§B). Important for IEEE support.

Conformance macro (§3.6). Important for portability of IEEE aware code. IEEE implementations predefine `__FP_IEEE__` indicating support for group I and II features. Replaces FPCE's `__FPCE_IEEE__`. `__FPCE__` not needed.

Other. Any IEEE specification for group I features that is deemed unsuitable for the general C9X document.

Group III

These features have not yet proven practical and useful enough for inclusion in the standard, or are superseded by changes mentioned above.

Widest need expression evaluation (§3.2.3.1). Would not be precluded. TR provides specification for interested implementors.

385

Wide parameter, return values, and parameters (§3.2.4). Would not be precluded. TR provides specification for interested implementors.

Pragmas. Replace with macros. See "Translation directives" above.

Predefined FPCE macro (§3.6). `__FPCE__` not needed as FPCE integrates into C.

Expression evaluation macros (§2.3.1). Superseded by `FP_EVAL_METHOD` . See item "Expression evaluation methods" above.