ISO / IEC JTC1 / SC22
Programming languages, their environments and system software interfaces
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC22
N 1418

AUGUST 1993

TITLE: Disposition of Comments Report for Second CD 10967-1: Information Technology - Programming languages, their environments and system software interfaces - Language independent arithmetic

SOURCE: Secretariat ISO/IEC JTC1/SC22

WORK ITEM: JTC1.22.28

STATUS: New

CROSS REFERENCE: Second CD10967-1 (N1263), N1325

DOCUMENT TYPE: Disposition of Comments Report

ACTION: For information to SC22 Member Bodies.
The final text of CD10967-1, prepared by WG11 will be forwarded to ITTF for DIS processing.

Address reply to: ISO/IEC JTC1/SC22 Secretariat
J.L. Côté
Treasury Board Secretariat
140 O'Connor St., 10th Floor, Ottawa, Ontario, Canada, K1A 0R5
Tel.: (613)957-2496 Telex: 053-3336 Fax: (613)957-8700

Response to International Comments on the Second Committee Draft of

ISO/IEC CD 10967-1.2
Language independent arithmetic --
Part 1: Integer and floating point arithmetic

As part of the second CD ballot, comments were received from three national bodies:

Netherlands (voting YES)
United Kingdom (voting YES)
United States (voting NO)

These comments are quoted below, along with the responses of SC22/WG11. As a result of these responses the US has changed its vote to YES.

Comment NL 1

Delete the Fortran 77 annex (clause B.5). The language described is no longer a standard.

Response: Accepted. Clause B.5 will be deleted. Fortran 90 will be referred to as Fortran. A note will be added to the remaining Fortran annex (B.6) relating its requirements to Fortran 77.

Comment UK 1

In the first goal, replace the two occurrences of "portability" with "predictability".

Response: The introduction has been completely rewritten based on text submitted by the UK delegate to WG11. The new introduction emphasizes the central importance of predictability, and explains the limits of portability.

Comment UK 2

Replace the second sentence of the first goal with: "The focus is on assuring users of single programs in specific programming languages of a documented level of accuracy of arithmetic operations on which they can rely, and the portability of such programs across platforms"

Response: See the response to UK 1.

Comment UK 3

In the second paragraph [of the introduction] replace "Our second goal" with "The second goal of this International Standard".

Response: See the response to UK 1.

Comment UK 4

LIA-1 should be accompanied, in an informative annex, by a specific

binding to IEC 559 (IEEE 754).

Response: After discussion with the UK representative, this was interpreted to mean that the relationship between LIA-1 and IEC 559 must be explained. The new annexes C and D compare the requirements of IEC 559 and LIA-1. In particular, annex C describes what is needed for an IEC 559 binding, and annex D describes what an IEC 559 system must do to satisfy LIA-1.

Comment UK 5

Definitions should be adapted so that they are consistent with IEC 559.

Clarification from the UK: Use the "significand model" both in the formalism and for emin, emax, exponent, and fraction.

Response: Subsequent investigation has shown that all SC22 programming language standards that present a model to the programmer use the fraction (rather than significand) model. An explanation of the relationship between the models, and the reason for LIA-1's choice, has been placed in the rationale. In light of this, the UK has withdrawn its suggestion.

Comment US 1.1

Restate the goal of the standard in terms that avoid use of the word "portable". Distinguish "portability" as something that can be preplanned as code is written or coincidental afterwards. Make clear that LIA-1 aims to enhance the former endeavor but will not significantly aid the latter. Make clear that the constraints imposed by LIA-1 are aimed at making the preplanned task easier (by requiring written documentation and provision of parameters and useful functions that can be economically implemented in software), but the constraints on hardware concern notification and value sets of datatypes only, and not operations on the datatypes.

Response: Accepted in spirit. The introduction has been completely rewritten to satisfy this comment along with UK 1 through 3. Note, the US has withdrawn the phrase starting "but the constraints ...".

Comment US 1.2

Acknowledge that in order to include Cray, the range of "bizarre" possible architectures is thereby increased enormously. The penalty such architectures must pay is increased documentation requirements and possible adverse public relations that arise from this documentation. (The aim is to illuminate rather than eliminate.)

Response: After discussion, the US has withdrawn this comment as moot. See the response to US 4.1.

Comment US 1.3

Make clear that LIA-1 intends to facilitate identification and illumination of deviations from IEEE 754. For other floating point datatypes, the LIA model provides a shorthand that makes documentation for model-conforming arithmetics easier to provide. For datatypes that do not fit the model (Cray particularly), conformance is still attainable provided the deviations are properly documented and the appropriate set of functions is supported.

Response: After discussion, the US has withdrawn the last sentence of this comment. The rest is addressed by new (and existing) material in the introduction and rationale.

Comment US 1.4

Emphasize there is no intent to define an "LIA machine". It therefore makes no sense to require programs to certify consistent or "correct" behavior on all conforming implementations.

Response: Accepted. Text has been placed in the scope clause.

Comment US 1.5

Make clear that the role of LIA-1 is to provide a bridge between a specific arithmetic implementation (such as IEEE 754) and a language standard. In order to be useful, it operates as a bridge between IEEE 754 AS WELL AS OTHER IMPLEMENTATIONS to bindings to specific languages.

Response: Accepted. WG11 believes that this is covered by other changes.

Comment US 1.6

Include an example of how to use the LIA-1 dynamically available parameters in a program that exploits them to choose an appropriate algorithm based on the available capabilities. This can be done in an annex in conjunction with an example of a specific binding.

Response: Accepted. A new annex (G) of examples has been added.

Comment US 2.1 [MAJOR]

Change the conformance concept to apply to datatypes, not to systems. In particular, decouple integer and floating point conformance. In order to do this, it must be made clear that the integer types implied in Section 4.2 need not be conforming types.

Response: Accepted. However, note that conformance is relative to a set of datatypes, since there must be conversion operations between conforming types.

Comment US 2.2

Tighten emax and emin parameter constraints to make all floating point parameters representable in their own type.

Response: Accepted. The new constraints are "p <= emax <= r^p-1" and "p-2 <= -emin <= r*p-1".

Comment US 3.1 [MAJOR]

Extend the recognized variants of the integer type to add "modular" to "bounded" and "unbounded". The latter two behave as at present. The modular variant is like bounded but does all arithmetic modulo (maxint - minint + 1). It does not overflow, but does remain subject to "undefined" for zero divisors.

Response: Accepted. See clause 5.1. However, the term "modulo" will be used.

Comment US 4.1 [MAJOR]

Include less regular architectures (such as Cray) in the following way. For each of the six major operations (add, subtract, multiply, divide, scale, and convert), require three parameters:

round_error (a floating point number that represents the maximum number of units in the last place that a normalized result can be in error),

input_perturb (a floating point number that represents the maximum number of units in the last place that operands would have to be perturbed separately in order to obtain a precise result), and

model_flag (a Boolean that specifies whether the properties of the version 4.0 LIA floating point model hold).

If model_flag is true, then a program can rely on maximum rounding_error being less than 1 ulp, all existing axioms being true, and can further interrogate the parameter round_style whose values come from the set {round to nearest ties to even, round to nearest ties away from zero, round towards zero, other}. If model_flag is false, then the axioms need not be true, but ALL DEVIATIONS MUST BE DOCUMENTED.

In making these changes, WG11 should also consider the possibility that the overflow and/or underflow threshold for each operation may not be equal to the representation thresholds defined by the model and may additionally be different from one operation to the next.

Response: After much discussion, WG11 decided
(1) the less regular architectures mentioned above will not conform,
(2) an annex will be added to explain how programming language standards can relax the LIA-1 requirements if they consider it necessary.
The US concurred. See the new annex B.

Comment US 4.2 [MAJOR]

Add a Boolean flag to indicate whether the datatype is IEEE 754 conforming (formats AND arithmetic).

Response: Accepted. See new clause 5.2.9.

Comment US 4.3 [MAJOR]

Correct the typographical error in the discussion of underflow that implies that IEEE 754 is still not supported due to divergent requirements on underflow detection.

Clarification from the US: The typo referred to is a missing clause in the definition of result_F. As a consequence, the current definition forbids the value (fmin_N minus .5ulp) from rounding to fmin_N without causing an underflow. Since this is permitted by IEC 559, LIA-1 should permit it as well.

Response: Accepted. The definition of result_F has been reworded to correct this.

Comment US 5.1

Codify the various uses of Boolean types in the standard to make clear that the implementation must support some encoding of "true" and "false" in some datatype.

Response: Accepted. See clause 5.0.

Comment US 5.2

Provide text that explicitly enumerates all hardware constraints.

Response: The US has withdrawn this comment. Note that the LIA-1 can be implemented entirely in software if desired.

Comment US 5.3

Define div and rem twice in Section 4.1 in order to avoid introducing the function rnd_I.

Response: Accepted. See clause 5.1.3.

Comment US 5.4

Restructure Section 4.2, reduce overloaded nomenclature, try to remove the functions (rnd_F, add_F*, and result_F) that contribute to the exposition but are not required of implementations, and clarify the roles of any such "helper" functions that remain.

Response: Accepted. A definition of "helper function" has been added. Helper functions have been separated from required operations and parameters as much as practical. A new explanatory clause 5.2.3 has been added. Clauses 5.2.4 through 5.2.6 have been reordered and explanatory material added. Clause 5.2.8 has been extracted from the general discussion of rounding. It was not feasible to remove rnd_F, add_F*, and result_F, although one axiom for add_F* could be simplified slightly.

Comment US 5.5

Reword the "range of translations" discussion to show how unary and binary operations may be composed in larger expressions -- use A*B + C*D as the example.

Response: Accepted. The intent will be clarified by phrasing the requirement in terms of language permitted expression transformations, rather than "translations". A non-exhaustive list of transformations will be added for further clarification.

Comment US 5.6

Try to remove terms such as "prompt", "hard-to-ignore", "error" and "exceptional" from the discussion of notification.

Response: Rejected. The terms will be retained. A note clarifying "hard-to-ignore" will be added. "Error" will be used only for for "delta from the correct value", and "exceptional" will only be used in the phrase "exceptional value". "Significant event" will be defined for 5.1.3. All uses of the offending terms have been examined for consistency; many uses have been changed. The definitions have been clarified.

Comment US 5.7

Make changes to the informative annexes to reflect changes to the normative text. Add clarifications to answer specific U.S. public review comments that indicated confusion. Annex B need not be extend to provide suggested bindings to IEEE 754.

Response: The US has withdrawn this comment. However, all normative changes

will be propagated to the annexes as appropriate. IEEE 754 bindings will not be added to the suggested bindings annex.

Comment US 5.8

Add text to enumerate and summarize all normative requirements.

Response: Rejected. Clarifying sentences have been added at critical points to clarify and separate "must supply" items from helper definitions.

Comment US 6.1

In view of the long lead times entailed, SC22 should start the process now of creating bindings between specific languages and the LIA-1 and IEEE 754 standards.

Response: WG11 concurs, but this will have no effect on the text.