**Proposal for C2Y**
**WG14 N3647**

| | |
|---|---|
| **Title:** | Semantic rules for constant evaluation |
| **Author, affiliation:** | C FP group |
| **Date:** | 2025-06-23 |
| **Proposal category:** | Editorial |
| **Reference:** | N3550 |

## Background

In email [cfp-interest 3419], Vincent Lefèvre noted that 6.6.1 #17 says

> The semantic rules for the evaluation of a constant expression are the same as for nonconstant expressions.[122)]

and he commented "this cannot be true for status flags, as they cannot be raised at translation time."

One could argue that an implication that status flags are raised, or a statement that they are not raised, at translation time is inconsequential, since C doesn't provide a way to access status flags at translation time.

However, a clear contradiction to 6.6.1 #17 is in 6.6.1 #5:

> … If a floating expression is evaluated in the translation environment, the arithmetic range and precision shall be at least as great as if the expression were being evaluated in the execution environment.

Note that this license is limited by the fact that evaluation methods characterized by the evaluation method macros (5.3.5.3.3) apply to both execution time and translation time evaluations (see footnote 117 in 6.6.1). Thus, using more range and precision for translation time evaluation is not permitted under the standard specified evaluation methods.

The suggested change below simply qualifies 6.6.1 #17 with "unless explicitly stated otherwise" to remove the contradiction.

Alternatively, 6.6.1 #17 could be deleted, because the semantic rules for evaluation of expressions don't say they are just for execution time. However, #17 might be a helpful in highlighting a significant fact, and it provides an anchor for footnote 122.

**Suggested Change**

Change 6.6.1 #17 to:

> The semantic rules for the evaluation of a constant expression are the same as for nonconstant expressions, unless explicitly stated otherwise.[122)]