Proposal for C2x
WG14 N3035

Title:	_BitInt Fixes
Author, affiliation:	Aaron Ballman, Intel Philipp Klaus Krause, Albert-Ludwigs-Universität Freiburg Elizabeth Andrews, Intel
Date:	2022-07-21
Proposal category:	Bug Fixes
Abstract:	Offers several minor corrections to the wording for bit-precise integer types.

_BitInt Fixes

Reply-to: Aaron Ballman (aaron@aaronballman.com),
Philipp Klaus Krause (krauseph@informatik.uni-freiburg.de)
Elizabeth Andrews (elizabeth.andrews@intel.com)
Document No: N3035
Date: 2022-07-21

Summary of Changes

N3035

- Adds prose about what was voted in at the Jul 2022 meeting
- Updated the [u]intmax_t exemption based on the latest working draft
- Cannot use a bit-precise type as the underlying type of an exact-width type, [u]intptr_t, or [u]intmax_t

N2960

- Prohibit using a bit-precise integer type as a compatible type for an enumeration

N2946

- Original proposal

Introduction and Rationale

After adding bit-precise integer types [N2763] to C23, some minor issues with the wording were discovered. This paper attempts to correct the issues known at this time.

The proposed changes in this document are orthogonal and should be voted on separately by the committee for inclusion in C23. All proposed wording in this document is a diff from WG14 N2912. Green text is new text, while ~~red~~ text is deleted text.

Already Adopted Changes

Several fixes were presented, and some were adopted, at the July 2022 WG14 meeting and are retained here purely for the historical record. The following fixes were voted in to C23:

- Clarifications to Integer Promotions
- Clarify Increment/Decrement Behavior

The remainder of this proposal covers modifications which required additional changes.

Exempt [u]intmax_t From Being Large Enough to Represent a Bit-Precise Integer

The current wording requires `intmax_t` and `uintmax_t` to be large enough to hold any value of an arbitrary bit-precise integer width. This could be a burden on implementations wanting to support a large width for bit-precise integers, and on users of `intmax_t` and `uintmax_t`. We propose to drop the requirement.

Proposed Straw Poll

Does WG14 want to drop the requirement of [u]intmax_t being able to hold any value of a bit-precise integer for C23 as in N3035?

Proposed Wording

Modify 7.21.1.5p1:

The following type designates a signed integer type capable of representing any value of any signed integer type with the possible exceptions of signed bit-precise integer types and of signed extended integer types that are wider than long long…

Which Types in stdint.h Should be Allowed to be a Bit-Precise Integer Type?

The current wording disallows the use of bit-precise integer types for the types `[u]intN_t`, but is unclear on [u]`intptr_t` and [u]`intmax_t`. The intent of the original proposal for bit-precise integer types was to disallow their use for all types defined in stdint.h.

On the other hand, the rationale for disallowing their use was that bit-precise integers narrower than `int` behave differently with regards to integer promotion vs. other types. So, it could make sense to only disallow the use of bit-precise integer types for stdint.h types narrower than `int`. This would allow implementations to easily provide e.g. an efficient [u]`intptr_t` type when `int` is 16 bits but pointers are 24 bits without needing support for a 24-bit integer type other than the bit-precise type.

It is worth noting that 7.21p4 has a blanket prohibition on stdint.h types being defined in terms of a bit-precise integer type.

At the July 2022 meeting, WG14 took opinion polls on whether the exact width types should be prohibited from being defined as a bit-precise type or not, and there was weak consensus to prohibit them.

Proposed Straw Poll

Does WG14 want to disallow types in stdint.h from being defined as a bit-precise integer for C23 as in N3035?

Proposed Wording

Modify 7.21.1.4p1:

The following type designates a signed integer type, other than a bit-precise integer type, with the property that…

…

The following type designates an unsigned integer type, other than a bit-precise integer type, with the property that…

Modify 7.21.1.5p1:

The following type designates a signed integer type, other than a bit-precise integer type, capable of representing any value of any signed integer type…

Type Compatibility With Enumerations

The standard currently allows an enumerated type to be compatible with “a signed integer type, or an unsigned integer type”, which currently allows for an enumerated type to be compatible with a bit-precise integer type. However, bit-precise integer types do not undergo integer promotion, and (considering C90 DR013) it is unspecified whether the composite type of an enum and its compatible integer type is the enum or the compatible integer type. So we think it is best to restrict enumerated types such that they cannot be compatible with a bit-precise integer type in order to reduce surprising behavior for users involving integer promotions.

Note, this fix is only required if WG14 does not adopt N3029 Improved Normal Enumerations. That paper makes the same changes as proposed here.

Proposed Straw Poll

Does WG14 wish to adopt N3035 Type Compatibility With Enumerations into C23?

Proposed Wording

Modify 6.7.2.2p5:

Each enumerated type shall be compatible with char, a ~~signed~~standard integer type, or an ~~unsigned~~extended integer type. The choice of type is implementation-defined, but shall be capable of representing the values of all the members of the enumeration.

Acknowledgements

We would like to recognize the following people for their help with this work: Jens Gustedt and Joseph Myers.

References

[N2763]
Adding a fundamental type for N-bit integers. Ballman, et al. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2763.pdf