Final Minutes for 20 Jun – 23 Jun, 2023

MEETING OF ISO/IEC JTC 1/SC 22/WG 14
WG 14 / N 3167

Dates and Times:

Tuesday,	20	June,	2023	13:30 – 17:00 UTC
Wednesday,	21	June,	2023	13:30 – 17:00 UTC
Thursday,	22	June,	2023	13:30 – 17:00 UTC
Friday,	23	June,	2023	13:30 – 17:00 UTC

Meeting Location

Teleconference

1. Opening Activities

1.1 Opening Comments (Keaton)

Keaton: this is my final meeting serving as Convenor. The new Convenor will be Robert Seacord.

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Notes
Aaron Ballman	Intel	USA	
Alex Celeste	Perforce Software	USA	prev. as Alex Gilding
Clive Pygott	LDRA Inc.	USA	WG23 liaison
David Keaton	Keaton Consulting	USA	Convener
David Svoboda	SEI/CERT/CMU	USA	Undefined Behavior SG Chair
David Vitek	Grammatech	USA	
Douglas Teeple	Plum Hall	USA	
Elizabeth Andrews	Intel	USA	
Fred Tydeman	Keaton Consulting	USA	INCITS/C Vice Chair
Freek Wiedijk	Plum Hall	USA	
Kuniho Kasugai	Woven Planet North America Inc	USA	
Rajan Bhakta	IBM	USA, Canada	INCITS/C Chair
Robert Seacord	Woven Planet North America Inc	USA	
Aaron Bachmann	Austrian Standards	Austria	Austria NB
Bill Ash	SC 22		SC22 manager
Chris Bazley	ARM	UK	
Corentin Jabot	Freelance	France	Guest
Dave Banham	BlackBerry QNX	UK	MISRA Liaison
Eskil Steenberg	Quel Solaar	Sweden	Sweden NB
Geoff Clare	The Open Group	UK	Invited Guest
Jakub Lukasiewicz	Motorola Solutions Systems Polska	Poland	Poland NB
JeanHeyd Meneide	NEN	Netherlands	Netherlands NB
Jens Gustedt	INRIA/ICube	France	France NB
Joseph Myers	CodeSourcery / Siemens	UK	UK NB
Kayvan Memarian	University of Cambridge	UK	
Martin Uecker	Graz University of Technology	Austria	
Michael Wong	Codeplay	Canada, UK	WG21 Liaison
Miguel Ojeda	UNE	Spain	Spain NB
Nick Stoughton	Logitech	USA	SC22 Austin Group Liaison
Peter Sewell	University of Cambridge	UK	Memory Model SG Chair
Roberto Bagnara	BUGSENG	Italy	Italy NB, MISRA Liaison

1.3 Procedures for this Meeting (Keaton)

As usual, we hold straw polls instead of formal votes; guests may vote. All we do is make recommendations to our parent group.

Due to a rules change, this is the last joint meeting of INCITS-C and WG14 for the time being. Future meetings will be held separately.

1.4 Required Reading

1.4.1 ISO Code of Conduct

1.4.2 IEC Code of Conduct

1.4.3 JTC 1 Summary of Key Points [N 2613]

1.4.4 INCITS Code of Conduct

Reviewed without comments.

1.5 Approval of Previous Minutes

WG 14 Minutes [N 3116] (WG 14 motion): some typos reported, Seacord listed with incorrect affiliation. Bhakta moves to approve, Tydeman seconds. No objections.

WG14 Minutes [N 3113] (WG 14 motion): Tydeman moves, Ballman seconds. No objections.

INCITS/C Minutes [pl22.11-2023-00004] (INCITS/C motion): Seacord moves, Bhakta seconds. No objections.

INCITS/C Minutes [pl22.11-2023-00005] (INCITS/C motion): Pygott moves, Seacord seconds. No objections.

1.6 Review of Action Items and Resolutions

DONE: Keaton: contacted JTC-1 for feedback about the use of colour in the IS. They will not be attending this meeting and the discussion will continue after C23 is published without colour.

1.7 Approval of Agenda [N 3137]

(INCITS/C motion, WG 14 motion): Ballman moves, Tydeman seconds. No objections.

1.8 Identify National Bodies Sending Experts

Austria, Canada, France, Italy, Netherlands, Poland, Spain, Sweden, United Kingdom, United States.

The Linux Foundation and the Austin Group are also represented.

Kasugai present to observe but not able to officially represent Japan.

1.9 INCITS Antitrust Guidelines and Patent Policy

Guidelines were observed without discussion.

1.10 INCITS Social Media

Links were displayed, no discussion.

1.11 INCITS official designated member/alternate information

Check with Bhakta if unsure.

1.12 Note where we are in the C23 schedule [N 3132]

The schedule is now extremely tight, with only one week available for editing and editorial review. Volunteers are wanted to close-read the document for the review. The document must be submitted by the 12th. Once the ISO editing process and DIS ballot are in-flight we are no longer allowed to discuss C23, so the October meeting will be for the floating-point TS work. The January meeting will be for ballot resolution. Technically changes can still be made but require an FDIS, so ideally it will be a yes/no.

Volunteers for editorial review: Bhakta (on behalf of CFP), Gustedt, Wiedijk, Seacord, Ballman, Pygott.

2. Abbreviated Reports on Liaison and Collaboration Activities

2.1 ISO TS status

ISO no longer wants to allow TS to work as extension-specifications. This affects TS 6010 and has already impacted C++. SC22 are working with ISO to try to find a resolution that reflects the way TS are used in practice.

Ballman: what are they supposed to be for?

Keaton: describing work in progress changes to an IS. This is not consistent with past uses. All of SC22 is affected, not just WG14.

We *can* still publish the same kind of content we have been delivering as TS as ISO whitepapers, which are easier to publish and have more relaxed requirements. We would prefer to continue using the TS mechanism if possible.

No other reports.

3. Study Groups

No reports.

We note that the TS 6010 ballot was rejected by ISO because of the above rule change.

4. Future Meetings

4.1 Future Meeting Schedule

The October meeting will be used to work on floating-point, C-documents, the issue tracker, and all the other things that have been delayed by the focus on getting C23 finished.

16-20 October, 2023 Virtual 13:30-17:00 UTC (proposed for discussion of TS 18661 Parts 4 and 5, and group administration)

Jens: can we discuss new features there?

Keaton: ideally, no, because the DIS will be in progress.

This will be a virtual meeting since it is only expected to be administrative.

Seacord will take over as Convenor in September, before this meeting.

22-26 January, 2024 Hybrid (proposed for DIS 9899 ballot resolution)

INRIA invites the group to meet in Strasbourg. Show-of-hands suggests enough people would attend, so it is agreed.

4.2 Future Mailing Deadlines

Pre-Virtual-202310 – 15 September, 2023

Post-Virtual-202310 – 17 November, 2023

Pre-Hybrid 202401 – 15 December, 2023

Post-Hybrid 202401 – 23 February, 2024

5. Document Review

(Comments with no discussion, including those added to the mass-approval list, are not included in this section)

Tuesday

5.1 Use of colour in the DIS

Keaton: suggest rejecting all colour-related CD comments and moving them to the GitLab issues list (GB-086, GB-109, GB-145).

Bazley: what is a DIS?

Keaton: the Draft International Standard. We hope for it to be the final step and if so it goes straight to publication. There is a Final Draft IS if technical changes are needed.

5.2 Preferences for DE-066

Myers: prefer to move rather than remove the example.

Straw poll: (decision) Would WG14 like to resolve DE-066 by moving the last line and copying the second-last line from Example 2 to Example 1?

Result: 19-0-1

5.3 FR-132, FR-133, FR-134

Moved to technical discussion.

5.4 [N 3135] CFP editorial comment responses

US-201 was originated by Tydeman, who withdraws it.

Straw poll: (decision) Should WG14 accept US-201 against the recommendation of CFP?

Result: 0-17-2

Bhakta: this is not what is considered a "conditional feature" elsewhere.

Seacord: the user wants to diagnose non-standard pragmas, and needs the wording to provide a basis.

Bhakta: conditional features are for things like `_Complex`, but the Recommended Practice is still good to do.

Myers: this is using "unrecognized", which is distinct wording.

Straw poll: (decision) Should WG14 resolve US-087 by accepting the CFP alternate wording in N3135?

Result: 18-0-2

Straw poll: (decision) Should WG14 resolve US-117 by using the CFP wording from N3135?

Result: 18-0-0

Straw poll: (decision) Should WG14 resolve GB-181 by adding CFP’s N3135 comment to the function list (with the `wcstol` -> `wcstold` typo fixed)?

Result: 17-0-1

Straw poll: (decision) Should WG14 resolve GB-188 by accepting the proposed change, but correcting `isinfinite` to `isfinite`?

Result: 20-0-0

Straw poll: (decision) Should WG14 resolve US-200 with the clarification proposed by CFP in N3135 instead of the original comment?

Result: 15-0-4

5.5 Other editorial comments

US-105, GB-118 and US-220 taken first because they have additional commentary.

Bhakta: we brought this up before but someone objected.

Tydeman: no longer recall what the issue was.

Straw poll: (decision) Should WG14 resolve US-105 by accepting the comment and adding the word "mode" after the words "constant rounding"?

Result: 15-0-2

Straw poll: (decision) Should WG14 resolve GB-118 by accepting the comment but applying it to 7.12.16.2?

Result: 15-0-2

Straw poll: (decision) Should WG14 resolve US-220 by accepting the comment but removing the index link to page 610?

Result: 17-0-0

Bhakta: maybe file `__unsequenced__` as a GitLab issue instead? (Myers will file it)

Gustedt: this is partially automated by a script, but the entry point to page 610 is irrelevant.

Straw poll: (decision) Should WG14 resolve US-221 by accepting the comment but removing the index link to page 610?

Result: 14-0-0

Straw poll: (decision) Should WG14 accept ALL OTHER editorial comments (which excludes FR-132, FR-133, FR-134), using CFP's preferences for overlapping comments where listed in N3135?

Result: 15-0-1

Myers: pointing out the preference requirement for the decision.

Bhakta: does this override Myers’s change?

Keaton: this is for “all other” comments, so the change is preserved.

ALL editorial comments resolved.

5.6 Technical comments

Starting with the CFP recommended actions.

FR-132 (duplicate `<stdbit.h>` interfaces)

Gustedt: we should avoid repetition in the interfaces we added to `<stdbit.h>`. There is a lot of duplication at the moment. Including in this case an exactly identical specification.

Myers: disagree that these are the same. One of these is the other one, plus one; except for zero.

Gustedt: as I read it these are exactly the same.

Ballman: me too.

Celeste: prefer to wait for the author to be available to explain.

Myers: “most significant index” is a term of art defined in 7.18.1.

Celeste: so these functions count from opposite ends?

Keaton: this is confusing enough that we should wait for Meneide.

Bachmann: this is a late design comment, but only counting one way is clearer.

Bhakta: Jens, do you agree with that?

Gustedt: not sure, but would prefer to only count one way always.

Keaton: this needs more discussion on the Reflector.

Wiedijk: these functions match hardware intrinsics in practice. Each function matches some existing opcode.

(deferred for the author)

GB-010 (dollar in identifier)

Bhakta: we are repeating ourselves, we already accepted this.

Ballman: perhaps we should make a list to mass-accept the comments we have already seen before?

Keaton: no such list exists yet.

Myers: this came up in reviewing the document after applying CD1 comments.

Wiedijk: can we change the “te” column to “missing”? (no)

Celeste: this is really just a GitLab issue, then?

Myers: yes but we didn’t want to do so because this is the ballot period.

Keaton: we have a procedural requirement to vote.

Seacord: at least one subsequent comment (US-032) touches this, may create merge conflicts.

(deferring until Jabot is available)

US-036 (enum aliasing)

Myers: the proposed wording is asymmetric, similar to `char` – intentional?

Seacord: intended to add the underlying type of an enumeration as being compatible with it.

Myers: the previous version didn't even consider this. If everyone else is fine, so am I.

Straw poll: (decision) Should WG14 accept US-036?

Result: 14-0-3

GB-037 (composite type of unevaluated VLA)

Bazley: are we really adding another way to have UB?

Celeste: there is no way to make this situation well-defined, so it's not worse.

Straw poll: (decision) Should WG14 accept GB-037?

Result: 14-0-3

US-045 (extended integer constant expressions)

Myers: distinguish the cases of "other" forms, which may appear to violate constraints, like the implementation details of `offsetof`.

Ballman: changing the corner cases has a knock-on impact on the types of expressions.

Myers: other forms still aren't allowed if they violate an explicit constraint.

Ballman: this rolls the state back to how it was in C17; changes are needed, but this is not the right way. We have 35 years of implementation experience of this.

Tydeman: that was the result of a DR against C99, though?

Ballman: that doesn't matter if it wasn't integrated because it was unimplementable.

Celeste: would consider it unreasonable for constructs like `__builtin_same_type` *not* to be constant expressions.

Seacord: make them UB?

Ballman: I don't like that – they're needed for initialization, which is already long-broken. That kind of UB would get us yelled at.

Seacord: "this is an extension" is a UB category. What about "reserved for extensions"?

Bhakta: propose a different wording: "may accept other implementation-defined integer constant expressions".

Ballman: that's fine but assumed the existing sentence implies that.

Bhakta: perhaps it is implementation-defined *whether* they are an integer constant expression?

Ballman: the Committee intent was *not* to allow extensions, which is where we disagree.

Gustedt: can we delete the footnote?

Ballman: we should delete it.

Bhakta: so, why did WG14 decide this?

Tydeman: the decision was procedural – we did not believe that DRs could be technical.

Ballman: we want VLA status to be portable.

Bachmann: the outcome can also depend on the rounding mode. Casting a value from `float` to `int` could be problematic?

Bhakta: this doesn't depend on the rounding mode.

Bachmann: it would if we allowed composite floating expressions.

Myers: code can be *valid* in different ways depending on whether an expression is a constant or not.

Seacord: I would want usages to be flagged by `-pedantic`.

Ballman: wouldn't you get that?

Seacord: implementation-defined behaviour is allowed there.

Ballman: we would need some recommended practice then.

Gustedt: the current poll proposal is progress, we should keep moving forward.

Bhakta: the footnote should call out that there are two valid semantics here, rather than be deleted.

(Ballman takes homework to produce clearer final wording)

DE-046/FR-073 (forward parameter declarations)

Uecker: we brought a new paper for forward-declared parameters. It stil has wording issues and we may need to defer the entire feature.

Bhakta: I want at least *some* solution, because right now C23 breaks existing code.

Myers: I want forward declaration of parameters, but don't think this solution is ready.

Celeste: it is possible to write extremely "surprising" code using forward declarations by hiding and repeating side effects. Jens's solution is adequate for the fix, but don't think it is a good fit for the language overall.

Ballman: agree that we don't have time to fix this before DIS. Clang wants this too, not just for VLAs but for bounds checking as well.

Gustedt: agree that it may be too late to add this.

Bachmann: I do not want to wait another decade for this feature!

Ballman: *only* KnR-style functions are broken by C23. Which have very few users.

Uecker: we do want to explore bounds-safety better,

Keaton: we have the option of saying that we intend to address this in future.

Ballman: bounds safety wants to do member lookup, which is its own issue. We would hope for a design space that can solve the problem in common.

Bazley: I do like Martin's examples, they provide a simple solution. I would like less syntax though.

Ballman: we know there are wording issues in this paper.

Bhakta: can we fix that later?

Ballman: the Clang community doesn't like the GCC style, and is going forward with the FR-073 syntax for bounds checking.

Seacord: so… we have *no* experience?

Celeste: implementation experience is not a problem, user experience is.

Bazley: broad concerns.

Wednesday

(continuing the CFP technical comments to allow time to reply)

(no objections raised)

DE-001 (nature of UB)

Uecker: this proposal comes from the UB Study Group, clarifying the different interpretations of UB. Abstract vs. concrete – with not everything becoming invalid in the latter case. There is consensus that observable behaviour that occurs before the UB is triggered should be safe. Users have asked for this, and it only adds a NOTE.

Gustedt: agree with this, it is a good change. Talk of "subsequent" and so on, but UB can also refer to non-specific operations like a deadlock, so would prefer to use "happens after".

Myers: this names observable behaviour but then redefines it with a different meaning. Instead of "i.e." the NOTE should refer directly to 5.1.2.3.

Ballman: how exactly does this match practice? Agree with the goal, but it doesn't seem to. These changes will be difficult to make.

Seacord: MSVC does this, and the other major vendors don't.

Bachmann: code supersedes implementations. Convenience is not the issue.

Uecker: we looked at MSVC, but received the reply that it was a bug. Especially Microsoft's `volatile` is known to be buggy.

Ballman: we have to admit reality though, a user NOTE that doesn't describe behaviour is not helpful.

Wiedijk: strong agreement with Aaron – this is a huge and fundamental change and expect it to give optimizer folks a very hard time.

Straw poll: (decision) Should WG14 accept DE-001, substituting "happens after" for "subsequent" and referencing 5.1.2.3 instead of "i.e. ..."?

Result: 9-5-4

Bhakta: I believe this is unimplementable.

Ballman: I agree and am only happy with it being a non-normative NOTE.

Steenberg: the least we could do is make it observable. There are lots of places where I is not knowable. The compiler is allowed to undo work *at* the UB, but it must have done the work in order to reach it.

DE-002/FR-238 (integrate pointer provenance)

Gustedt: provenance is covered by TS 6010, which ISO doesn't want to exist. Original votes for TS 6010 were based on the assumption that it was an option. We agree that this is a lot of work, but the patches to integrate the TS into C17 already exist.

Seacord: a rule change is annoying, but there are other paths forward. This is too much.

Celeste: the scale of the work being demanded of the editor is unreasonable. There is not enough time to do it.

Ballman: we told implementers we wanted feedback on the TS, so we sent the wrong message.

Myers: the document referenced is not the latest version; we have no new experience; the editing is against C17; there is no public fix document; there are new functions in C23 that need to be considered. The actual fixes would be significantly different from this document.

Gustedt: this is exaggerating the incompatibility. We have a common problem with saying there will be a TS that then won't.

Seacord: I could accept adding it as a Future Direction. (no wording for this)

Bhakta: we don't yet *know* the TS won't happen, and we can still publish the content as a whitepaper. Without the experience, which was the whole point, the entire reason for the document is side-stepped.

Straw poll: (decision) Should WG14 accept DE-002 and FR-238?

Result: 2-11-5

DE-027/GB-028 (composite type of structures and unions)

Ballman: why is the added bullet in change 1 needed?

Uecker: the rule is not there yet, we imply it exists without specifying it.

Gustedt: structure and union types are not considered "derived", so aren't covered by that.

Seacord: unclear what "same type" means if the types are different.

Keaton: if we are applying a change we need the document.

Uecker: that's purely editorial.

Bhakta: agree, make that edit after accepting the comment.

Seacord: would want to change "such a type" to "one of these types".

Ballman: does "an enumerated" in 3a mean we get to pick? Can it be any type?

Uecker: it would have to fulfil all the other requirements.

Ballman: so it doesn't matter which one is picked? (no)

Seacord: can we just fix it up instead of making two changes?

Keaton: needs homework.

Straw poll: (opinion) Does WG14 prefer change 3a in N3122?

Result: 6-4-10

(weak consensus, against author’s preference)

DE-029/FR-030 (enumerated type conversions)

Uecker: we prefer Jens’s wording.

Gustedt: this is clearer for users, and what everybody does.

Straw poll: (decision) Should WG14 resolve DE-029 and FR-030 by accepting FR-030?

Result: 18-0-2

FR-132, FR-133, FR-134 continued (bit interfaces)

Meneide: I can accept the first comment, and keep “first leading one”.

Bhakta: so Joseph and I read these as different, were we wrong? These functions seem to be locating the bit starting at opposite ends of the value – they do not return the same value, and they do something different.

Meneide: that means the problem applies to the other one then.

Celeste: that would seem to be a different bug than FR-132?

Bazley: it was unclear which end is being counted from, and we needed context.

Meneide: I prefer to reject FR-133 and FR-134, the functions are not the same. There *is* a direct relationship between them, and one implies the other. Hardware varies in which one is the native intrinsic, so the API should not bias the user one way or the other, instead standardizing the whole set and the architecture can provide all. The user can select the optimal one if they know it exists. It’s not that we don’t know the relationship, but that we don’t want to impose the correct base.

Bachmann: relatedly, most people use “count” – can we adopt names that put this in, making it clearer what the functions do? Or is it too late?

Bazley: should we even offer these APIs? “C is not a large language” and so on.

Ballman: we do standardize practice though.

Meneide: the comment is essentially right but it identifies the wrong API. We should be consistent and remove redundancies.

Wiedijk: “first trailing one” refers to a different bit? This is not the same?

Meneide: I don’t think we should remove that. Allowing similar functions to coexist reduces the API bias. I am happy to reject all three comments.

Gustedt: I do not agree that compilers wouldn’t be able to rewrite this.

Meneide: even the major vendors like MSVC made no attempt to rewrite these until this year, so any optimizations here are new. Last time, Krause told us he does *some* work, but that having the names for the user intent makes optimization much easier than trying to detect patterns. We should acknowledge the small compilers, and even the big vendors have trouble here. We should let users request these directly and spend the optimization budget on harder tasks.

Straw poll: (decision) Should WG14 resolve FR-132 by removing section 7.18.14?

Result: 7-5-8 (not consensus to accept)

Meneide: it's safer not to remove something if we're not completely confident it is identical.

Straw poll: (decision) Should WG14 accept FR-133?

Result: 0-12-8

Straw poll: (decision) Should WG14 accept FR-134?

Result: 0-14-7

FR-135 (UB in bit ceiling)

Meneide: providing the next power of 2 for an input means the 65th bit is not representable in a 64-bit target, etc. In hardware, shift-truncate would produce a zero. If overflow is an error, it could trap. We argue that UB is bad, but zero is a natural result. C++ chose to make this UB so that it would become an error in `constexpr` evaluation, which we do not have yet.

Gustedt: UB is just wrong here. To wrap or trap doesn't require UB.

Celeste: UB is also not needed to future-proof against misuse inside `constexpr`.

Ballman: are there any architectures that *wouldn't* shift-truncate, or are we just making work for people?

Meneide: wouldn't disadvantage TI or ARM.

Ballman: we got rid of non-two's-complement anyway.

Bhakta: the current text allows for zero, but also other ways to detect the error. Normally we try to maintain C++ compatibility.

Seacord: specifying a behaviour *is* compatibility with C++.

Bazley: I have never heard a user describe UB as useful.

Straw poll: (decision) Should WG14 accept FR-135?

Result: 15-3-4

DE-047/GB-048 ("declaration declares identifiers")

Uecker: these comments are mostly agreeing – attempting to clarify that nested sub-contexts are not included. Substantially the same outcome. These are also related to DE-049 and GB-050, as we've accidentally broken `__auto_type` declarations.

Gustedt: I prefer GB-050 – it has the same understanding with direction to avoid the weird cases. Extra clarification is good. Implementations can have extensions to `auto` as well.

Uecker: I am happy with the wording and it addresses the issue. We should standardize practice. This fails that declared intention and I don't think the Constraint still makes sense.

Myers: GB-069 is also related although not in this paper. Same category of issue.

Keaton: hybrid wording?

Straw poll: (decision) Should WG14 resolve DE-047 and GB-048 by accepting GB-048?

Result: 13-0-3

Straw poll: (opinion) Would WG14 like to remove 6.7 Declarations p5?

Result: 5-5-7

Ballman: does GB-050 allow for the existing practice in DE-049? This is working code and we created an incompatibility!

Seacord: the opinion poll is tied but you can redo it if you want.

Bhakta: we know of one implementation only, not all compilers.

Myers: some cases are not incompatible with C++, which disallows definitions in `sizeof` etc.

Steenberg: we don't have to adopt everything an implementation does. This feature mostly comes from C++ and the strict subset is still compatible with C++.

Gustedt: I am against removing it because of the burden for smaller implementers.

Keaton: so move it to Semantics?

Gustedt: that creates an arbitrary distinction between Constraints and Semantics and adds another UB. All cases except for paragraph 5 are already UB (no diagnostic required). It's not unreasonable but would need a new paper to propose it, therefore homework.

Bazley: I did not intend to vote to add UB! Agree that we don't need everything from C++.

Keaton: we have an obligation to be compatible where appropriate. We need to be careful not to be too gratuitous in incompatibility, and should have a reason.

Steenberg: there is a difference between going *against* C++, and not going *as far* as C++.

Ballman: there are also technical arguments against accepting it.

Myers: this is intended to allow shorter code, but not to allow code that couldn't previously be expressed at all. That turned out to be more of a concern for deduced-return -types and scope escape (no longer proposed). Does this allow anything that couldn't be expressed?

Celeste: we did not find there was any extra complexity for implementation. Disallowing surprises is more MISRA's jurisdiction.

Myers: non-local escape of types is the interesting part.

Ballman: it creates an internal incompatibility which can allow this.

Gustedt: there is precedent from disallowing definitions within `offsetof`.

Straw poll: (decision) Should WG14 resolve DE-049 and GB-050 by accepting GB-050?

Result: 4-3-8 (not consensus to accept)

Ballman: would it be reasonable to convert this to Recommended Practice?

Celeste: having a Constraint is not a terminal error, implementations can still choose to accept it if it is what users really want.

Bhakta: many modes allow for suppression of warning messages.

Uecker: this came up as a user, moving from GCC to C23 caused errors. So we do have the user experience saying that we have broken a feature.

Myers: there is the option to make paragraphs 5 and 12 implementation defined. We did this for the additional declarator forms, with Recommended Practice to behave the way C++ does. Implementation-defined acceptance with Recommended Practice to follow C++ precedent. It would need homework but would be consistent.

Ballman: this is reasonable. Constraints tell users something is a bad idea, and we have sensible users. Normally we allow good code, and *maybe* bad code.

(homework to Uecker)

US-077 (`_Generic` constants)

Myers: the wording is wrong – whether it's constant doesn't matter, the controlling expression and the unused branches are relevant here. Wish to change this.

Ballman: I will take homework. I thought the constant-expression status mattered if the generic selection expands to a VLA with an evaluated size. We're dealing with the "obviously unevaluated" parts which is not what the change offers. It's still obviously unevaluated even if it has side effects.

Seacord: get rid of all the bullets and just say "unevaluated"?

Myers: this is not a simple runtime property, it needs to be trivial and compile-time. We could add a definition for "obviously unevaluated".

Gustedt: I am sympathetic to Seacord, but not now – this has wide utility.

Ballman: agreed, I like that but we can't get it right quickly.

DE-079 (`typeof` evaluation)

This was also fixed in CD1 and not applied.

Bhakta: do all compilers do this?

Ballman: only ICC seems to reject it, of the compilers that implement `typeof` – everyone else either evaluates or doesn't support `typeof` at all.

Straw poll: (decision) Should WG14 accept DE-079?

Result: 12-0-3

DE-089 (obsolete integers as null pointers)

Myers: not clear that this is an improvement. `if (p != 0) ...` is harmless, the conversions are a different matter.

Celeste: not convinced this will ever pass out of common usage. If adopted this should go in the main text.

Ballman: similarly, it sends an odd signal if we can't get rid of it.

Bhakta: agree that usage is common; but we never did conclude what Obsolescent means, so this could be valid.

Gustedt: it aims to create a feedback loop – naming something Obsolescent doesn't always work, but if we don't say anything, nothing will change.

Ballman: we cannot reasonably add this as an on-by-default warning.

Steenberg: can we say this is discouraged?

Ballman: the Standard shouldn't be opinionated, unless we try to split out unsafe code.

Seacord: if we narrowed it down to the problematic case it would make sense.

Ballman: we would need to identify that specifically.

Straw poll: (decision) Should WG14 accept DE-089?

Result: 3-13-1 (rejected)

FR-098 (`assert` depends on `NDEBUG` for correctness)

Myers: it has always been expected that the argument may not be valid as an expression in non-debug mode, such as calls to debug-only functions that are missing entirely with `NDEBUG`. So this imposes a new requirement.

Gustedt: the new text would allow that?

Bazley: agree with Joseph, this completely changes some debug statements.

Bhakta: previously we also accepted the exact opposite comment, CA-5 in CD1.

Straw poll: (decision) Should WG14 accept FR-098?

Result: 1-11-4 (rejected)

GB-108 (default argument promotions to `printf`)

Myers: o define the type in `<inttypes.h>` the same way as the direct formats does require implementation changes, but making the conversions explicit is better.

Ballman: I didn't see where any changes would be needed.

Myers: if the implementation uses a promoted format for a wider type, it will change to use the exact un-promoted type.

Ballman: that shouldn't impact the output?

Myers: it can’t, if the user passes a value of the type they named.

Gustedt: so this *is* a normative change.

Straw poll: (decision) Should WG14 accept GB-108?

Result: 11-0-6

DE-126 (unnecessary UB in specification of `va_arg`)

Gustedt: this loses the requirement to be an object type.

Myers: that would still be a runtime undefined behaviour.

Bhakta: this is “not worse” than the text which just adds a star.

Celeste: changing one UB to another UB is not worse either.

Gustedt: we don’t say it must be complete, but “object” is a change that we added.

Thursday

Unanimous consent to strike “shall” from DE-001.

GB-101/US-032 (`$` and `@`)

Jabot: the current wording is for implementation wanting `$` in their identifiers. We did *not* add it to the basic character set, so the universal character name remains allowed. Clang will crash if you try to use it! The easiest thing is to align with C++ and just put `$`, `@` and `` ` `` in the basic character set, instead of treating them alongside it. People do want to use the universal character name in string literals and we wanted to allow them to continue to do so.

Ballman: unaware of why we would want a distinction. This is also very useful for a shared lexer with the C++ frontend.

Celeste: this is an elegant solution in itself even without the C++ justification. Divergence however would be much worse.

Myers: this adds the universal character names *and* changes the definition of the basic character set. This doesn’t address that `$` is allowed in identifiers, and may need additional wording. The previous principle was that it would have been UB, but removing that removes the permit for the extension.

Jabot: agree that these are different issues.

Celeste: if it’s not UB, can we add them as punctuators or other syntax elements that are unused by the rest of the grammar?

Myers: that would make them syntax errors. It would still need text to allow use in an identifier without a diagnostic. The agreed text is predicated on them *not* being members of the basic character set.

Jabot: this is not changing that, it’s just something that can appear in a source file. The implementation should document the extended identifier grammar. I do not want to document implementation extensions in the Standard. There is no wording to prohibit what implementations do now, as an extension. It *could* be in the grammar and unused, there is no need to change it.

Ballman: if we change the wording we need a homework paper.

Myers: changing UB to a constraint violation is sometimes good, but this is extension space, and expected to be permitted, so more consideration is needed.

Keaton: you need the history – when `$` was originally proposed for use in identifiers, UB was important and the users considered it a “congratulatory diagnostic” of the UB.

Gustedt: UAX#31 has the `$` treated similarly to `_` in `XID_Start`, and we aim to align with that.

Ballman: if it’s a constraint violation, it is still extension space if the invalid code is defined as valid, even though we “must” warn about it.

Celeste: users do not want this to be taken away. It is important to communicate the intent to let them use it.

Jabot: we don’t want to actively encourage usage either.

Gustedt: usage is a reality. We should just make it implementation-defined whether `$` is an identifier character.

Seacord: Alternative 1 may not have been the best choice.

Myers: if we accept this, Alternatives 2 or 3 would work better. We should adopt those if we adopt this.

Bhakta: can we revisit the fact that we rejected those in favour of Alternative 1? (Yes)

Bazley: please clarify the intent behind the alternatives?

Seacord: Alternative 2 implies a requirement.

Myers: they are based on wording which is no longer up to date. We need a homework.

Celeste: I explicitly dislike Alternative 2.

Jabot: so here are no conflicts between the two papers? Even if we allow it as an identifier, we don’t want to allow the universal character name. So the design spaces are related but not overlapping.

Bazley: why *shouldn’t* C force support for `$`? It seems easy to do.

Jabot: it takes a design space away from the languages depending on C. Cool enough for C itself, but forces accommodation from others. We should not claim it at this point.

Ballman: so we are converging on resolving GB-010 with Alternative 3?

Keaton: there is no problem adopting one alternative only.

Celeste: use of the `$` marks a cultural boundary between portable and GNU C dialects.

Bhakta: this does make the diagnostic required in isolation, though.

Straw poll: (decision) Should WG14 accept US-032?

Result: 8-2-8 (clear consensus despite abstentions)

DE-027/GB-028 continued (composite structure and union types)

Uecker: change 4 now makes this explicit implementation-defined behaviour rather than unspecified.

Tydeman: so “one of” could be a new type, if the compiler wants?

Uecker: yes. I’m not happy about any of this “same type” wording anyway.

Bhakta: change 4 goes beyond what was asked for.

Uecker: the preference poll was close, so it is a compromise.

Ballman: this isn’t a backdoor, so much as an emergent bug.

Seacord: it seems to create a new type with a different range of values.

Uecker: there is no degree of freedom for the type – it has to *appear* the same.

Seacord: when do you want it to be an integer type?

Uecker: later declarations collect information in all of the composites.

Seacord: so this is a statement of existing praxis.

Bhakta: I support the change but this is not in the NB comment. We voted that out of the paper.

Keaton: this is a compromise between close positions in the opinion poll. It is OK. Thank you for being careful.

Straw poll: (decision) Should WG14 resolve DE-027 and GB-028 by accepting N3141, changing the typo "is the sames" to "is the same"?

Result: 14-0-3

DE-046 continued (forward parameter declarations)

Gustedt: we now have a compromise for both paths. This takes the GCC syntax, but restricts it to only integer types, which are the thing we actually need. The syntax now only allows typed identifiers, not full declarators, so forward-declaring arrays or pointers is no longer possible.

Uecker: so as not to break GCC, it is implementation-defined.

Bazley: I am worried about restrictions that would make this inconsistent. There are also no examples of comma-separated declarations?

Gustedt: that part of the GCC syntax is not included. This is a subset.

Myers: it solves some problems by not using declarator, but other parts of the Standard assume a declarator is present, such as the definition of the start of its scope, or the definition of an ordinary-identifier. Those things are now broken. There is a similar global impact check needed to that for `constexpr`, and we cannot do this check now. The problems in the wording *will* have consequences.

Ballman: I agree fully with that, and also this creates an inconsistency with C++, which will never add this feature. I would like a liaison group discussion. My main concern is the timing.

Jens: what about the precedent of declarations in `if` and `while`?

Bazley: I have ergonomic concerns that users will not understand this.

Celeste: there is no conflict with C++. I really *want* to vote yes, but we cannot seriously tell implementers “you know what we mean, ignore the wording”.

Bhakta: we need a fix for the code that we broke. We can break either the users or the implementers.

Straw poll: (decision) Should WG14 resolve DE-046 by accepting N3140?

Result: 6-10-2

Unanimous consent to apply as resolving FR-073 as well.

DE-049/GB-050 continued (“declaration declares identifiers”)

Uecker: N3144 takes Myers’s solution, and makes it implementation defined.

Bhakta: the wording in the last part doesn’t say *what* is implementation-defined. (this can be editorial)

Gustedt: this is non-normative and can be left to the editors.

Straw poll: (decision) Should WG14 resolve DE-049 and GB-050 by accepting N3144?

Result: 10-0-4

Keaton: so GB-050 was rejected and this resolves it.

US-045 continued (extended integer constant expressions)

Keaton: I would prefer “are expressions” for consistency with the existing text.

Ballman: fine for that to be editorial.

Bhakta: reconciling the footnote with the change – the footnote says it is implementation-defined, the main text says it is an extension.

Keaton: the footnote doesn’t say *why* it is an integer constant expression. The array isn’t the implementation-defined behaviour.

Gustedt: wouldn’t mind an editorial change, do mind adding more homework items.

Straw poll: (decision) Should WG14 resolve US-045 by accepting N3138, appending ", depending on whether `(int)+1.0` is an extended integer constant expression." to the footnote?

Result: 15-0-1

US-077 continued (`_Generic` constants)

Ballman: since “obviously unevaluated” is not a Term of Art, the wording is awkward, and there is an editorial change for “part of a `sizeof` or `alignof`”.

Straw poll: (decision) Should WG14 resolve US-077 and US-078 by accepting N3143?

Result: 13-0-3

GB-128 (diagnosing inappropriate `va_start` arguments)

Myers: this is intended to give implementations more freedom to diagnose when the argument is inappropriate according to C17 rules.

Gustedt: the specification as as a prototype, so this is not needed.

Celeste: implementations don't need permission to diagnose whatever they want here, or at least the intent was not to forbid them from doing so.

Myers: it would handle the cases where an unbalanced token sequence would otherwise break the code.

Straw poll: (decision) Should WG14 accept GB-128?

Result: 9-0-6

FR-131 (`[[unsequenced]] <stdbit.h>` functions)

Gustedt: although not a coordinated approach, there is no chance of conflict. It may also be worth extending to other interfaces, but those would change their semantics, whereas these were not there before.

Celeste: future proofing is not an issue for this change because it will impact user code either way if we change it.

Ballman: policy question – can implementations add or remove attributes?

Gustedt: Myers observes that adding them can change the semantics, so we should have a NOTE. As far as removing them, so far there is not much to remove.

Bachmann: is this part of the type? Would it compose in the ternary operator? (Yes)

Ballman: if they impact observable semantics, this is not a valid attribute.

Gustedt: they do *not* change the semantics of the program as-written by the user.

Ballman: then it should be a quality-of-implementation issue.

Gustedt: this lets the type of ternary results compose to the attributed type. QoI would make determining the intent harder. It's the *same issue* as composing a pointer to `restrict`.

Celeste: so the problem doesn't go away if we don't do this, it just manifests slower.

Bhakta: we can't change that right now, but `[[unsequenced]]` follows and fits in with precedent.

Straw poll: (decision) Should WG14 accept FR-131?

Result: 10-3-2

DE-137 (`typeof` inside `offsetof`)

Gustedt: it is very late to allow this given that compilers may have already changed.

Celeste: this was really the rationale?

Myers: yes, but I still don't want to allow this anyway.

Uecker: the language allows this everywhere, and now we have the compatibility rules that mean constructions will match.

Bhakta: I hate the idea, but there is a use case, and it is consistent.

Bazley: the original proposal was to add UB – shouldn't it be to add a diagnostic?

Uecker: this reduces the UB surface area.

Celeste: UB *is* the constraint violation tool for Section 7 at the moment, which generally doesn't specify Constraints.

Straw poll: (decision) Should WG14 accept DE-137?

Result: 9-0-6

GB-141 (inconsistent behaviour in `printf` length specifier)

Gustedt: musl do not oppose opion 1, but there is a lack of data. Therefore maybe enforcing it is not a good idea.

Bhakta: we are a non-POSIX platform and are no biased towards option 1.

Seacord: this may conflict with 7.23.6.1p9?

Myers: don't think so.

Straw poll: (opinion) Does WG14 prefer Option 1 to Option 2 in GB-141?

Result: 6-3-6

(deferred, Austin Group is not present)

C23 vs C24

Myers: here was the beginning of the discussion on he Reflector about the version number.

Gustedt: I don' mind either way but it is a business decision and we should know ASAP.

Celeste: I think it is more useful for the version to describe when WG14 stopped changing the Standard, than whenever it was published. Anyway, the community is already calling it C23.

Ballman: agree, and that's what C++ does.

Seacord: maybe some of us have already written it down...

Wiedijk: I do not want another sentence like "C17 is sometimes referred to as C18".

Bhakta: echoing Alex and Aaron, the number should be based on what we do; *but* officially we should support ISO in this. It is however a version macro not a publication date.

Myers: there is a reasonable argument that we're *not* done until DIS finishes.

Gustedt: like Seacord, I want to be able to name it in my book!

Bazley: "C23" is a nice name. Seriously, that is a point for adoption.

Seacord: it is an arbitrary number and the copyright doesn't need to match the year anyway. Copyright lies all the time like this.

Wiedijk: I strongly prefer C24. Does ISO have any opinion?

Keaton: no. Whatever we call it isn't the "real" title of ISO 9899 anyway.

Myers: there is precedent set by C++ 97/98.

Friday

GB-141 continued (inconsistent behaviour in `printf` length specifier)

Clare: this is a corner case that musl handles differently to everyone else. All similar cases write `NUL` here, and we would like it to be consistent. POSIX can overrule this in C17, if we know C23 will change to match. The options are to require consistency, or just allow it, but POSIX will require it.

Gustedt: musl are OK with option 1. I didn't check the non-POSIX implementations and don't know if option 1 is safe for them. Although I prefer 1, option 2 is safer at this phase.

Seacord: this may conflict with 7.23.6.1p9?

Bhakta: I checked with two other non-POSIX implementations and option 1 is fine for them.

Ballman: I don't think I conflicts. I prefer option 1.

Seacord: paragraph 9 is hard to read but OK, if we assume the first sentence, there is no conflict in the rest of the paragraph.

Ballman: it implies the rest is required.

Straw poll: (decision) Should WG14 resolve GB-141 by accepting Option 1?

Result: 16-0-0

GB-159 (`mktime`)

Clare: arising from previous comments. Much discussion but mostly not about C.

Bhakta: is this from wraparound on the year?

Clare: yes, when the year is `INT_MAX`.

Bhakta: normally we don't show an error on wraparound, but we do have the `ckd_` functions now.

Clare: those features wouldn't be in POSIX, and anyway they require the calculation to be done user-side.

Myers: `ckd_` doesn't help users, who would have to reimplement the logic themselves. It might be useful within the implementation but doesn't help the user.

Bhakta: the feature could have been added to `mktime`?

Bazley: aren't there other functions that can return indistinguishable errors? A user would never expect error values to also be return values.

Celeste: it's better to have the information than to not. We can change to better options later.

Bachmann: I don't see a problem with `errno`.

Clare: POSIX requires a non-standard value here, so we didn't see it as appropriate. And there is existing practice to do it like this.

Bachmann: C doesn't require any fixed value? And `errno` can be extended.

Clare: we didn't think that ISO would want it, and there is no practice for that.

Keaton: we do usually say what `errno` is set to.

Gustedt: I am concerned by this late normative change. musl doesn't do it.

Myers: the interface is less problematic than `errno` would be because it is not external state – it's no *as* bad as `errno`, we do need the same change to `time_gm` though. Previous issues like UTC/DST weren't relevant, but this is.

Seacord: I don't see why we *can't* add an `errno`. Every technical comment is a possible normative change.

Bachmann: it is not breaking anything, and changing for C23 is expected.

Bazley: if -1 is a valid return value, why is it documented as an error? Can it be simplified so only one place needs to be checked?

Gustedt: technical changes are made during this phase, but this has not been seen before. This is not high risk, but also was not on *anyone's* radar, and is totally new. Normally we would reject it on those grounds.

Clare: agree about `time_gm` – sorry, we missed it.

Straw poll: (decision) Should WG14 accept GB-159?

Result: 3-6-5

Straw poll: (opinion) Would WG14 like to accept GB-159 in C23 with further changes for `time_gm`?

Result: 6-3-7 (sentiment in favour to pursue in a future edition)

Seacord: I don't like setting up an incompatibility, this is inviting a DIS comment…

(N3147 written over the break)

Straw poll: (decision) Should WG14 resolve GB-159 by accepting N3147 (identical text to Reflector message SC22WG14.23725)?

Result: 9-1-8

GB-163 (`fgetwc`)

Clare: the POSIX change was made to `fgetwc` but not to the corresponding `put`.

Gustedt: so the difference is in the error indicator.

Bhakta: a difference between put and get is nice to fix for symmetry, but not technically needed.

Straw poll: (decision) Should WG14 accept GB-163?

Result: 13-1-2

GB-010 continued (dollar in identifier)

Gustedt: I think this is elegant. It should be added to Annex D as well.

Seacord: changes are redundant, we are not changing Unicode.

Gustedt: the Annex is informal, descriptive, additional. Just as we describe the use of `_`, we need to describe the use of `$`.

Straw poll: (decision) Should WG14 resolve GB-010 by accepting N3145, and adding the sentence "Additionally, implementations may add the character `U+0024`, `DOLLAR SIGN`, or `$`, to the set of permitted Start and Continue characters. " immediately before the final sentence of D1 p1?

Result: 15-0-1

6. Clarification Requests

The previous queue of clarification requests has been processed.

7. Other Business

Straw poll: (opinion) Would WG14 prefer to change the informal name of the C Standard edition to C24?

Result: 4-12-2 (sentiment to keep the name C23)

Tydeman: will there be another public draft Standard?

Meneide: next week, ASAP.

Bhakta: can we get that with and without colour? Want to ensure everything has been applied.

Celeste: is this the last public draft PDF?

Keaton: from this moment it **cannot be public**, including the editorial review.

Ballman: maybe if we had two documents, a "next working draft" so that one can be public? We have made a lot of changes and there *will* be divergence between the PDF and final Standard.

Keaton: **no.** We cannot have multiple versions of the Standard in-flight at once.

Wiedijk: this means there will be no public draft of C23 that includes the changes to UB time-travel.

Gustedt: can we make an early C26 draft after DIS?

Keaton: yes, we can start work right away if it passes.

8. Recommendations and Decisions reached

8.1 Editorial Decisions

Would WG14 like to resolve DE-066 by moving the last line and copying the second-last line from Example 2 to Example 1?

19 / 0 / 1

Should WG14 accept US-201 against the recommendation of CFP?

0 / 17 / 2

Should WG14 resolve US-087 by accepting the CFP alternate wording in N3135?

18 / 0 / 2

Should WG14 resolve US-117 by using the CFP wording from N3135?

18 / 0 / 0

Should WG14 resolve GB-181 by adding CFP's N3135 comment to the function list (with the `wcstol` -> `wcstold` typo fixed)?

17 / 0 / 1

Should WG14 resolve GB-188 by accepting the proposed change, but correcting `isinfinite` to `isfinite`?

20 / 0 / 0

Should WG14 resolve US-200 with the clarification proposed by CFP in N3135 instead of the original comment?

15 / 0 / 4

Should WG14 resolve US-105 by accepting the comment and adding the word "mode" after the words "constant rounding"?

15 / 0 / 2

Should WG14 resolve GB-118 by accepting the comment but applying it to 7.12.16.2?

15 / 0 / 2

Should WG14 resolve US-220 by accepting the comment but removing the index link to page 610?

17 / 0 / 0

Should WG14 resolve US-221 by accepting the comment but removing the index link to page 610?

14 / 0 / 0

Should WG14 accept ALL OTHER editorial comments (which excludes FR-132, FR-133, FR-134), using CFP's preferences for overlapping comments where listed in N3135?

15 / 0 / 1

8.2 Technical Decisions

Should WG14 resolve GB-005 by accepting the comment with the changes proposed by CFP in N3135, and using "produce" instead of "make"?

16 / 0 / 0

Should WG14 accept CA-022, with the change to the spelling of "floating-point" proposed in N3135?

13 / 0 / 3

Should WG14 accept US-006?

16 / 0 / 1

Should WG14 accept US-036?

14 / 0 / 3

Should WG14 accept GB-037?

14 / 0 / 3

Should WG14 accept DE-001, substituting "happens after" for "subsequent" and referencing 5.1.2.3 instead of "i.e. ..."?

9 / 5 / 4

Should WG14 accept DE-002 and FR-238?

2 / 11 / 5

Should WG14 resolve DE-029 and FR-030 by accepting FR-030?

18 / 0 / 2

Should WG14 resolve FR-132 by removing section 7.18.14?

7 / 5 / 8 (not consensus to accept)

Should WG14 accept FR-133?

0 / 12 / 8

Should WG14 accept FR-134?

0 / 14 / 7

Should WG14 accept FR-135?

15 / 3 / 4

Should WG14 resolve DE-047 and GB-048 by accepting GB-048?

13 / 0 / 3

Should WG14 resolve DE-049 and GB-050 by accepting GB-050?

4 / 3 / 8 (not consensus to accept)

Should WG14 accept GB-069?

8 / 0 / 4

Should WG14 accept GB-072?

13 / 0 / 2

Should WG14 accept GB-065?

16 / 0 / 0

Should WG14 accept DE-079?

12 / 0 / 3

Should WG14 accept DE-088?

10 / 0 / 5

Should WG14 accept DE-089?

3 / 13 / 1 (rejected)

Should WG14 accept DE-094?

5 / 6 / 5 (rejected)

Should WG14 accept FR-098?

1 / 11 / 4 (rejected)

Should WG14 accept GB-108?

11 / 0 / 6

Unanimous consent to strike “shall” from DE-001.

Should WG14 accept US-032?

8 / 2 / 8 (clear consensus despite abstentions)

Should WG14 resolve DE-027 and GB-028 by accepting N3141, changing the typo "is the sames" to "is the same"?

14 / 0 / 3

Should WG14 resolve DE-046 by accepting N3140?

6 / 10 / 2

Unanimous consent to apply as resolving FR-073 as well.

Should WG14 resolve DE-049 and GB-050 by accepting N3144?

10 / 0 / 4

Should WG14 resolve US-045 by accepting N3138, appending ", depending on whether `(int)+1.0` is an extended integer constant expression." to the footnote?

15 / 0 / 1

Should WG14 resolve US-077 and US-078 by accepting N3143?

13 / 0 / 3

Should WG14 accept DE-126, changing "a type name" to "an object type name"?

14 / 0 / 1

Should WG14 accept US-127?

14 / 0 / 1

Should WG14 accept GB-128?

9 / 0 / 6

Should WG14 accept FR-131?

10 / 3 / 2

Should WG14 accept DE-137?

9 / 0 / 6

Should WG14 accept GB-144?

13 / 0 / 1

Should WG14 accept GB-148?

14 / 0 / 0

Should WG14 accept all other technical comments agreed to by CFP in N3135?

(US-113, US-114, US-115, US-116, US-120, US-125, GB-152, US-195, US-203)

13 / 0 / 3

Should WG14 resolve GB-141 by accepting Option 1?

16 / 0 / 0

Should WG14 accept GB-159?

3 / 6 / 5

Should WG14 accept GB-163?

13 / 1 / 2

Should WG14 resolve GB-010 by accepting N3145, and adding the sentence "Additionally, implementations may add the character `U+0024`, `DOLLAR SIGN`, or `$`, to the set of permitted Start and Continue characters. " immediately before the final sentence of D1 p1?

15 / 0 / 1

Should WG14 resolve GB-159 by accepting N3147 (identical text to Reflector message SC22WG14.23725)?

9 / 1 / 8

8.3 Review of Action Items

No Action Items taken this meeting.

9. INCITS/C Business

(INCITS/C minutes are no longer to be combined with WG14 minutes)

10. Thanks to host

Thanks to ISO for providing Zoom capabilities.

11. Adjournment

(INCITS/C and WG14 combined motion)

Tydeman moves, Seacord seconds. No objections.

Adjourned.

Special thanks from all members of the Committee to David Keaton <3