MINUTES FOR APRIL 2007
MEETING OF ISO/JTC1/SC22/WG14 AND INCITS J11
WG14/N1267

Meeting Location:

Monday, Wednesday, Thursday April 23, 25, & 26:

The Window Venue Hire
13 Windsor Street
Islington
London N1 8QG
UK
+44 (0)20 7288 7008
+44 (0)7977 049094

Tuesday 24th
London Art House
2-18 Britannia Row
Islington
London N1 8PA
UK
+44 (0)207 359 5195

Host:

Neil Martin (personal host, not BSI)

Host Contact information:
Neil B. Martin
TFJ Ltd
P. O. Box 506
Milton Keynes
MK7 8JD, United Kingdom
+44 (0)1908 645 717
+44 (0)8701 602 754 (Fax)

1. Opening activities

1.1 Opening Comments (Martin, Benito)

Neil Martin, TFJ Ltd, welcomed us to the Window Venue Hire. He apologized for the last minute change of venue, and for the fact that we are in a different venue for Tuesday. The Institute of Chartered Accounts is helping paying for this, since they bounced our original booking. Snacks and beverages will be available throughout the meeting, so breaks can be taken at any time. Rooms will be locked as required. Lunch is not provided.

NB, this meeting is hosted by TFJ Ltd, and not BSI.

1.2 Introduction of Participants/Roll Call

John Benito	Blue Pilot	USA	WG14 Convener
Randy Meyers	Silverhill Systems	USA	J11 Chair
Douglas Walls	Sun Microsystems	USA	HOD
Fred Tydeman	Tydeman Consulting	USA	
Arjun Bijanki	Microsoft	USA	
David Schwab	Oracle	USA	
P. J. Plauger	Dinkumware, Ltd	USA	
Tana L. Plauger	Dinkumware, Ltd	USA	
Nick Stoughton	Usenix	USA	
John Parks	Intel	USA	
Clark Nelson	Intel	USA	
Bill Seymour	self	USA	
Robert Seacord	CERT-CC	USA	
Tom Plum	Plum Hall	USA	
Christopher Walker	Dinkumware, Ltd	USA	
David Keaton	Self	USA	
Rich Peterson	Hewlett Packard	USA	
Keith Derrick	Plantronics	USA	
Derek Jones	Knowledge Software	UK	HOD
Neil Martin	TFJ Ltd	UK	
Joseph Myers	Code Sourcery	UK	
Lois Goldthwaite	Self	UK	(Thursday only)
Willem Wakker	ACE	Netherlands	HOD

Randy Marques	ATOS Origin	Netherlands	
Edison Kwok	IBM	CANADA	HOD

1.3 Procedures for this Meeting (Benito)

The Chair, Benito, announced the procedures are as per normal. Everyone is encouraged to participate in straw polls. INCITS J11 members are reminded of the requirement to follow the INCITS Anti-Trust Guidelines which can be viewed at http://www.incits.org/inatrust.htm.

All 'N' document numbers in these minutes refer to JTC1 SC22/WG14 documents unless otherwise noted.

1.4 Approval of Previous Minutes (N1198) (Hedquist)

Comments from Fred Tydeman:
1.2: Column alignment could be improved. Already done ... see update on Wiki.

Minutes approved as modified.

1.5 Review of Action Items and Resolutions (Stoughton)

ACTION: Ulrich Drepper and a small team to write a liaison report on threads to WG21, in response to Lawrence Crowl's presentation, N1196. JB, Nick, Tom Plum, and Bill Seymour to work on this with Ulrich.
OPEN

ACTION: Stoughton Editorial Group (Stoughton, Benito, PJ Plauger, Randy Meyers, Martyn Lovell, Ulrich Drepper) to edit N1193, TR 24731-2, as needed, and forward the revised document to the Convenor.
DONE

ACTION: Convenor to forward TR24731-2, as revised by the Stoughton Editorial Group, to SC22 for CD Registration.
DONE

ACTION: Convenor to change the C99 Rationale as proposed in N1189.
OPEN

ACTION: Tom Plum to notify WG21 as a liaison of a new macro, __STDC_MB_MIGHT_NEQ_WC_ , being added to C99: 6.10.8;p2.
DONE

ACTION: Edison Kwok and Editorial Group (PJ, Fred, Jim Thomas, Janis Johnson) to revise DTR 24732, and forward to Convenor.
DONE

ACTION: Convenor to forward DTR 24732, as revised by the Edison Kwok editorial group, to SC22 for DTR ballot.
OPEN

ACTION: Rich Peterson to submit paper for additional comments on TR 24732.
DONE

ACTION: Convenor to forward TR 18037, Support for Embedded Systems, to SC22 for publication, again.
DONE

1.6 Approval of Agenda (N1211)

Revised agenda approved.

1.7 Information on Next Meeting. (Plum)

Kona in October, following the C++ meeting. Should reserve quickly if not done yet. October 8-11 (Monday-Thursday).

1.8 Identification of National Bodies (Benito)

US, UK, Canada, Netherlands.

1.9 Identification of J11 voting members (Tydeman)

15 voting members present. Plantronics is a new member (non voting).

2. Liaison Activities

2.1 J11 (Walls, Meyers)

2.2 WG11 (Wakker)

Not much to report. LIA will be revised. LID is ready for publication. Document is being converted to MS Word, and this is causing substantial delay.

2.3 WG14 (Note: Cover current status of TR 18037 and TR 24731-1.) (Benito)

TR24731-1 (bounds checking) is in publication. TR 18037 (embedded) has had all defects addressed. SC 22 secretariat is requiring a NWI ballot for revising this, which is against the JTC 1 procedures.

2.4 J16/WG21 (Note: C++ consider TC I, TC II and TC III - all defects that are closed.)

WG21 met last week in Oxford. Very overloaded. Adding extra meetings to meet the goal of publishing a revision of the standard in 2009. Threading support (i.e. memory model) looks like it will be palatable to C. Actual threads will be library level, and modeled closely on POSIX pthreads (though Windows threads could also be used for implementing). Still issues around thread cancellation.

Special Math (in C++ this is in TR1, in C it is a separate TR) is going to move to its own IS. We should take appropriate action to move our TR to be a part of this IS.

Clark Nelson proposed new text for Sequence Points, and if we revise our standard, we should align with this new wording.

2.5 FSG (Free Standards Group) (Stoughton)

Nick posted document to wiki (now N1230). FSG has merged with OSDL to become the Linux Foundation. LSB TC1 is in preparation.

2.6 OWG:Vunerability (Plum)

Motivation from two sources: safety critical community (Ada leads the way), and software security community (e.g. Homeland Security, NSA, CERT-CC)

Safety is well understood, special tools exist. Spark-Ada.

Security has different perspective.

Overall direction is still uncertain.

2.7 Other Liaisons

None

3. Defect report status

(***Note:*** *Cover what should be in TC III*. Also cover the N1124 change to fwprintf that was not included in TC II) (Benito)

10:30 - 10:45 - Morning break -

4. Potential defect reports

N1203, Fred Tydeman (USA); Missing semantics of comparision macros. This is a defect. Becomes DR334.

N1204, Fred Tydeman (USA); _Bool bit-fields. This is a defect. Becomes DR335.

N1206, Nick Stoughton (USA); TMP_MAX. This is a defect. Becomes DR336.

N1207, Nick Stoughton (USA); stdio.h macro definition problems. This is a defect. Becomes DR337.

N1208, Rich Peterson (USA); C99 seems to exclude **indeterminate value** from being an uninitialized register. This is a defect. Becomes DR338.

N1220, Joseph Myers (UK); Variably modified compound literals. This is a defect. Becomes DR339.

N1221, Joseph Myers (UK); Composite types for variable-length arrays. This is a defect. Becomes DR340.

N1222, Joseph Myers (UK); [*] in abstract declarators. This is a defect. Becomes DR341.

N1223, Joseph Myers (UK); VLAs and conditional expressions. This is a defect. Becomes DR342.

N1224, Joseph Myers (UK); Initializing qualified wchar_t arrays. This is a defect. Becomes DR343.

(SC22WG14.11300) Preprocessor DR from WG21: this is a defect. ACTION Clark Nelson to write up preprocessor issue with casts from WG21 in DR format. Will become DR344.

Derek Jones introduced a potential defect from Rob Arthan (UK) on parameter scopes (see http://wiki.dinkumware.com/twiki/pub/Wg14london/Documents/arthandr.htm). This is a defect. Becomes DR345.

5. Decimal Floating-point

(N1212, N1213, N1214, N1215, N1216, N1217, N1218, N1219, N1227, N1228) (Kwok, Peterson)

N1212:

A set of minor edits from HP.

1. Point 1: non-controversial.

2. Point 2 will move to rationale.

3. Point 3: will use a consistent name throughout.

4. Point 4: will use 200704L for next edit.

5. Point 5. Should use “*SUBNORMAL_MIN” to match C++ usage.

6. Point 6 is being withdrawn by submitter.

7. Point 7: Agreed we should defer to 754R here.

8. Point 8: Agreed there is a deficiency here.

9. Point 9: Remove the troublesome sentence? NOT AGREED ... leave this open for now and revisit. DR318 is relevant to this discussion.

10. Point 10: Agreed.

11. Point 11: Editorial, OK.

12. Point 12: Agreed.

13. Point 13: Agreed.

14. Point 14: Agreed.

15. Point 15: Agreed.

N1213.

Two problems from HP.

PJ objects to removal of _Decimal32. It is widely used in the embedded community, particularly for signal processing (Problem 1). Derek Jones agrees that there are many classes of applications that have very restricted range or precision requirements. Rich Peterson requests at least a statement in the rationale expressing that IEEE recommend the 32 bit format only for storage, not computation. Would prefer words in the main TR itself. A non-normative note. Second para of problem 1 still has one thing that should be done: *remove "; and for other values of DEC_EVAL_METHOD, they are otherwise implementation-defined".*

Problem 2. Not a problem, no changes required.

N1214

HP problem with floating-point model for decimal
Problem 1: Agreed ... we should stick to our model.

Problem 2: Agreed.

Implement the changes described in N1214. One typo: *where s, b, e, p, and fk are as in 5.2.4.2.2.* Should be *where s, b, e, p, and f[k] are as in 5.2.4.2.2.*

N1215

printf problems from HP.

Paper proposes changing the handling of %g. However, discussions leading up to this meeting has modified this proposal to use %z. This format should not have leading zeroes for the exponent. PJ argued for using (re-using?) %a (%Da) for this ("actual"). TR already prohibits hexadecimal output of decimal floating numbers. Use the term "subnormal" not "denormal" as appropriate (page 2, first para, change "c is adjusted through the subnormal range accordingly,..."). Agreed. ACTION Fred to provide additional (potentially troublesome) examples for the Decimal FP input functions.

N1216

HP problems with TTDT

The proposal in N1216 removes the concept of TTDT. Edison believes that this is not necessary. Lot of discussion; does not look like consensus to remove TTDT. Soften the language to not *require* breakage (as it does now) when mixing unsuffixed floating point constants. Using a #pragma will lead to issues with C++. ACTION Bill Plauger and Rich Peterson to propose new words for N1216, unsuffixed floating point numbers and TTDT.

N1217

Terminology inconsistencies.

Agreed. Willem: we need to check the existing standard to ensure we don't use the terms "format" and "encoding". Rich: C standard uses the term "value representation", not "encoding". ACTION: Derek Jones to make a pass through the C standard itself to ensure that the terms "format" and "encoding" are not used in any inconsistent manner. This is *not* tied to this paper.

N1218

Obtaining the quantum is important; previous meeting asserted that it was possible to get, but we now believe this assertion to be wrong. Agreed to add these functions.

N1219

Encode/decode. PJ suggests that these are esoteric and incomplete; are they worthwhile? Serialization (e.g. using printf/scanf) is the only portable way to get floating point numbers in and out. Rich pointed out that these functions are described in 754R. But we don't have every single function from 754R anyway. No consensus for this paper.

6. Decimal Floating-point continued.

N1227

Miscellaneous edits 1-4 are all accepted. Edits 5 and 6 are subsumed by the changes in N1215, ensuring that that document uses the term “subnormal”.

Comments requiring further discussion:

1. Discussed in Portland and agreed there, but missed the minutes. ldexp() will be given the same treatment that frexp() gets for base 10.
2. General feeling is that the given expression should be a constraint violation. The TR does not currently say this. Fix.the TR
3. The current words match the binary description, and most people feel it is right. However, the proposed words are not wrong either. No change.
4. This is correct. Words about overflow should be removed.
5. See N1214. (answer to question is yes, these are all different cohorts of the same value). Also add recommended practice along the lines of that in the main standard (6.4.4.2).
6. See N1228 for further details.

N1228

This paper introduces the gcc extension *typeof* to help determine if a value is decimal or binary. Alternative is *radix* operator from 754R. A TR is probably not the right place to sneak in *typeof* as a fix for one small issue! It will likely be discussed under the revision discussion later this week.

Shouldn't do anything half-baked. This issue may need compiler or library magic for fpclassify() to return the right answer. “Do it right or don't do it at all.”

This paper could be reconsidered if (and only if) both *typeof* and Decimal FP are added to a revision of the standard.

7. Special Math

(N1182, N1185) (Plauger)

N1182 has passed CD registration ballot. Some comments received. N1185 lists these comments.

1. Missing a rationale. Yes, true. There should be a brief rationale. TR is based on ISO 31. Should include things like why we removed the hyper-geometric functions. Rationale would be a separate document. There are also a couple of functions that don't exactly follow ISO 31; this deviation should be explained. ACTION PJ to produce a rationale for N1182 before the FDPTR ballot.

2. Should the m argument on the **Associated Laguerre**, **Associated Legendre**, and **Spherical Legendre** be implementation-defined for values >= 128? ACTION Chris Walker to investigate if the m argument should be implementation defined in the functions mentioned in item 2 of N1185. Add if required. --- Yes, it should be implementation defined for m>128.
3. Specify a macro that would hide the new functions — something along the lines of __STDC_WANT_MATH_SPEC_FUNCS__. Yes, we should follow this practice.
4. Should there be a reference to **J** in 2.1.10 and 2.1.19? Add a reference.
5. Should there be a reference to **N** in 2.1.21? Same as above.
6. Should Jerome Spanier and Keith B. Oldham: *An Atlas of Functions*, Hemispere Publishing Corp, 1987 be added to the Bibliography? Not normative. Might add confusion if formulas are different in some way. Useful reference book. Lots of discussion. Straw poll – 2-8. Should not add this. Perhaps add extra bibliography to rationale. Split bibliography into normative references and bibliography. Bibliography will be in the rationale.
7. 2.1.2 *associated Legendre functions*, should (-1)**M be added to definition? PJ believes not. Sentence or two in rationale explaining why not.
8. 2.1.2 *associated Legendre functions*, should for odd M then |x|<=1 be added? Yes. The constraint on the range of x is wrong for associate Legendre. Should be |x| <=1.
9. 2.1.3 *beta function*, what happens when X and/or Y is a zero or negative integer? The value is currently not constrained. Would make sense to add some implementation-defined constraints. ACTION Chris Walker to investigate sensible constraints for the beta function, and make recommendation for disposal of comment 9 in N1185.
10. 2.1.3 *beta function*, should the following text be added? [text in N1185]. No.
11. 2.1.4 *(complete) elliptic integral of the first kind*, should 0<=k<=1 be added? Yes, this range constraint should be added.
12. 2.1.4 *(complete) elliptic integral of the first kind*, should K(1.) is +INF be added? This follows clearly from the formula.
13. 2.1.5 *(complete) elliptic integral of the second kind*, in *Atlas of Functions* has E(k) not E(k,pi/2). Document is based on ISO 31, not on the Atlas of Functions. There is variation in the literature on this definition.
14. 2.1.5 *(complete) elliptic integral of the second kind*, should 0<=k<=1 be added? Yes, this is an omission.
15. 2.1.5 *(complete) elliptic integral of the second kind*, should E(1.) is +1. Not pointing out special cases.

16. 2.1.6 *(complete) elliptic integral of the third kind*, should **effects** use k and nu? Yes ... typo here. There is probably a global sweep required to ensure such the relationship between arguments and greek letters is correct.
17. 2.1.6 *(complete) elliptic integral of the third kind*, *Atlas of Functions* has (1+nu*sin(theta)**2) in denominator; not (1-nu*sin(theta)**2). Same as 13.
18. 2.1.6 *(complete) elliptic integral of the third kind*, should it be PI(nu,k) instead of PI(nu,k,pi/2)? Add a cross reference to 2.1.13 and eliminate the repeated definition.
19. 2.1.6 *(complete) elliptic integral of the third kind*, should |k| <= 1 be added? Yes.
20. 2.1.6 *(complete) elliptic integral of the third kind*, should PI(nu,1.) is sign(nu)*INF; PI(-1,k) is INF *not sure of its sign* be added? No ... we picked one of a variety of definitions.
21. 2.1.6 *(complete) elliptic integral of the third kind*, Why is C function f(k,nu), while math function is f(nu,k)? Seems like the C function argument order should be swapped. This follows a stylistic decision made early on in the process. No change required.
22. 2.1.7 *regular modified cylindrical Bessel functions*, should the note ... be added? Add to the rationale, not to the TR.
23. 2.1.7 *regular modified cylindrical Bessel functions*, should the text ... be added? No. This is not a text book. Uniformity of style.
24. 2.1.8 *cylindrical Bessel functions (of the first kind)*, should the text ... be added? Uniformity of style. A simple knowledge of advanced math and 20 years of experience is all that is needed.
25. 2.1.8 *cylindrical Bessel functions (of the first kind)*, should ... be replaced by ...? No, this follows a direct request from Fermi Lab to restrict the range. Rationale required.
26. 2.1.8 *cylindrical Bessel functions (of the first kind)*, should (x/2)**nu before the summation sign in the math formula be removed? We could, but it is there for uniformity of style. Style is generally not to remove the common factors.
27. 2.1.9 *irregular modified cylindrical Bessel functions*, should the note ... be added? Maybe to the rationale.
28. 2.1.9 *irregular modified cylindrical Bessel functions*, should K(0.) is +INF be added? No.
29. 2.1.13 *(incomplete) elliptic integral of the third kind*, should the formula in the denomerator be changed to (1 + nu*sin(phi)**2)? No.
30. 2.1.13 *(incomplete) elliptic integral of the third kind*, should the arguments be reorder to nu, k, phi to match the math function? No.

31. 2.1.14 *exponential integral*, should the formula be changed from Ei(-INF) = integral from -INF to x of exp(t)dt/t? Stylistic. No.
32. 2.1.14 *exponential integral*, should the following be added Ei(-INF) is 0, Ei(0) is -INF, Ei(+INF) is +INF? No.
33. 2.1.15 *Hermite polynomials*, should following be added ...? No. Style.
34. 2.1.17 *Legendre polynomials*, why |x| <= 1? Functions are defined for all x. This is an intentional restriction.
35. 2.1.18 *Riemann zeta function*, zeta(0.5) does not match the *Handbook of Mathematical Function*. This definition clearly needs fixing. There is a hole here.
36. 2.1.18 *Riemann zeta function*, should zeta(1.) is +INF be added? No.

Additional points: John noted that he left the C++ clause names in there because he didn't have time to alter the LaTex macros that add them and believes they are useful. Everyone believes these are good and useful, and we should consider doing them elsewhere.

ACTION: Bill, Fred, & JB to review changes made to N1182 and the rationale when the updated document is available.

ACTION: Convener to forward the updated document for PDTR ballot when ready.

NB The next ballot will not be the last ... there will be another ballot after that.

Bill also wants to add an implementation-defined range to the beta() function. No objection.

On the IS/TR question. Most people agree that this is a useful approach. It is not Language Independent.

8. Possible defects in ISO/IEC TR 24731-1

(N1210) (Seacord)
Some file I/O issues from CERT-CC.

Issue 1. Exclusive open access through fopen_s(). Add a new mode character 'x'. Some people suggest adding this to part 2. We haven't succeeded in our goal for the fopen_s function. Is it a defect? Does not fit the part 2 scope. Do a DR for this and immediately publish rationale.

Issue 2. There is a problem relating to the fact that C does not define directories. The POSIX mkstemp() function is the closest to what is being requested (but returns an fd, not a FILE *). Cover this with text in the rationale.

Issue 3. There is no requirement for non-predictability of filenames. Could be recommended practice, not normative. Even non-predictability is not perfect. A recommended practice could easily be added to the rationale right now without any problems. General consensus is that this is the best approach for this issue.

ACTION Randy Meyers, Robert Seacord and Nick Stoughton to write additional rationale for the issues raised in N1210.

9. CERT C Programming Language Secure Coding Standard (DRAFT)

(N1209) (Seacord)

Arising from discussions following the managed string library proposal, Robert has prepared this document describing “good programming practices”. Idea is to provide both rules and recommendations. Could comply to rules. Recommendations are generally more useful, but not enforceable.

Document is currently being built on a wiki (http://www.securecoding.cert.org), and there is a wg14 username that can edit.

Three criteria: severity of vulnerability, likelihood of attack, and remediation cost.

Goals are to:

- establish secure coding practices within an organization
    - may be extended with organization-specific rules
    - cannot replace or remove existing rules
- Train software professionals
- Certify programmers in secure coding
- Establish base-line requirements for software analysis tools
- Certify software systems

Nick encouraged Robert to continue to develop this, with the possible future goal of a type 3 TR in the future.

There is a level of concern ... Robert would prefer to get us to rip his document apart rather than just say “good work, keep going!” Possibly need smaller review groups to go through specific parts in detail. Especially looking to mine this document for future revision features. ACTION Tom, Nick, Mark, David to review N1209.

In particular, the INT, ARR, STR, MEM, FIO sections are well baked and worthy of mining for future revisions. Other sections need review for completeness.

There are several sources for the base material, and the idea is not to provide a subset of the language.

Joseph had a specific comment:

1. Compliant solution (signed) on page 110 has an overflow possibility.

Updates to the document should be made directly through the wiki, rather than by submitting written requests.

10. Defect Reports

TC3 is being prepared. Goal is to have all of the defects known about *at this meeting* should have been at least reviewed. What is to be included in TC3 is to be decided here. What does not get into TC3 will be targeted at a future revision. We will not run the defect report mechanism as such during a revision.

If any National Body is considering submitting a DR while a revision is under way should consider submitting a paper for the revision instead.

Clark Nelson was volunteered to run the DR review.

First look at the “review” status DRs.

DR 298

Validity of constant in **unsigned long long** range

This change clarifies the wording. The UINT64_C() macro is a macro, and its argument is a pre-processing token, and not an integer constant at this point. Perhaps we should use “a pp-number in the form of an unsuffixed integer constant” for the UINT64_C macro text (7.18.4 para 2). Derek believes this is wrong, since we only care about things is phase 5 or later at this point. Can we use other macros as arguments to this macro? Original words seem good enough. Agreed. Move to CLOSED.

DR 311

Definition of variably modified types

Submitter agrees current proposed words solve the problem adequately. Agreed. Move to CLOSED.

DR 325

Is an implementation permitted to return an empty string for **strerror()**?

No action required. Move to CLOSED.

DR 326

asctime() tm_year gt 9999

Note there is an editorial issue with “If any of the fields ... contain” (American v British English usage). The editor can handle this as he sees fit when applying this change. Move to CLOSED.

DR 327

Italicize definition of variable length array type, add forward references.

Almost entirely editorial. A revision project might want to consider rewriting the definition if VLA in more positive terms. Move to CLOSED.

DR 332

gets is generally unsafe.

Agreed. Move to CLOSED.

DR 333

Missing Predefined Macro Name

There is an error on the DR index page that has the wrong title for this defect. Move to CLOSED.

Open Defects

DR 314

There is a paper from Randy Meyers on this, N1226. The defect deals with structure tag compatibility across translation units. 6.2.7 para 1 deals with this issue.

Randy believes the current proposed answer to question 3 (Is an implementation required to accept compiling the three translation units above together into a program?) is wrong. The answer to Q2 is “Yes, undefined behavior”, and our previous reasoning was correct.

There is no requirement that struct tags themselves are some sort of global entity; it is quite possible to have two incompatible structures with the same name in two translation units that refer to different objects (see example 5 and 6).

The thing that is surprising in Q3 is, looking at TU2 ,g1 is dealing with a different set of types than g2. There appears to be nothing in the rules that should prevent this program from compiling and linking. However, it is unlikely that such a program *should* have defined behavior, and we need language to permit a compiler/linker to fail it.

It is important to allow example 7 from N1226, which deals with permitting overlapping namespaces between different modules.

It does seem to be late in the life of the standard to break things rather than go for fixing this in a revision. Joseph suggested the following words:

6.2.7 after paragraph 2 insert:
There shall exist a partition of all the structure and union types in the program into disjoint classes such that (a) if two types are in the same class, then they are compatible and (b) whenever two structure or union types are required to be compatible, including by (a), they are in the same class. If there does not exist such a partition, the behavior is undefined.

Nick suggested that the Committee response to Q3 should be: “The committee believes there should be no such requirement in the Standard, and are considering this issue for the next revision.”

Fred Tydeman reminded us that N522 (1996) spoke to this issue.

This DR will never affect TC3 anyway. Leave OPEN. Do not change proposed response. ACTION Joseph Myers to submit his words for DR314 as a proposal for C1x in the post-London mailing.

DR 315

Implementation-defined bit-field types

The “Fall 2007” discussion should be “Fall 2006”. The proposed TC effectively allows an implementation to choose to follow either the C90 defect reports or the C++ behavior.

This should be discussed in more detail for the potential revision.

Agreed that the current words are a masterful way of hiding the problem, and should move this to REVIEW.

DR 328

String literals in compound literal initialization

In Portland we appear to have decided to leave 6.5.2.5 para 1 alone, but cannot currently understand why we did this. Rich thinks it can't just be deleted, since this is a constraint on the list, while the 6.7.8 constraint is on the type of the entity to be initialized. This is an argument for leaving the proposed TC as it is.

Agreed that this is good enough. Move to REVIEW.

DR 329

Math functions and directed rounding

Note that the line is in the wrong place ... should be above “Committee Discussion”. Also the <\p> should become </p>.

On F.9.8.3 The nextafter functions should be updated along the lines of:

> Even though underflow or overflow may happen, the returned value is independent of the current rounding direction mode.

Note that the use of “may” is problematic with the ISO style rules. Use “can occur” instead of “may happen” here. Better is to remove the “Even though underflow or overflow may happen,”. Same for nexttoward.

Committee Discussion says “They might break an existing implementation (possible). Therefore, only add these to annex F.” These changes only apply to annex F in the first place.

There are still a lot of actual edit changes to make the words something that can be voted on. Globally change “Should be updated along the lines of” or “Could be updated along the lines of” to “Add the following sentence”.

DR 330

Externally visible exceptional conditions.

In the proposed TC, 7.21.1 should be 7.12.1, and “exceptions” should be “exceptional”. It is sometimes extremely difficult to catch intermediate inexact exceptional conditions, and we shouldn't try too hard to force this behavior. Even underflow is difficult to avoid. Should exclude inexact and underflow.

Footnote “spurious”, or move existing footnote closer.

State the requirements positively: “Each function shall execute as if it were a single operation without generating any of the exceptions invalid, divide-by-zero, or overflow except to reflect the result of the function.”

Fred sent email to the reflector reminding us about discussion in Portland:

> In the Rationale, please add to section 6.5 Expressions, as a new paragraph, words along the lines of:
>
> The "inexact" floating-point exception is NOT an exceptional condition because "inexact" arises from computing a mathematically defined value in the range of representable values, therefore, from the definition, "inexact" is not exceptional. This matters for spurious exceptional conditions in the math library (7.12.1).

These words should be in the committee discussion, and added to the rationale.

Move to REVIEW, with proposed TC:

Each function shall execute as if it were a single operation without generating any of the exceptions invalid, divide-by-zero, or overflow except to reflect the result of the function.

DR 331

permit FE_DIVBYZERO when errno says EDOM

Some belief that previous consensus was “no change required”.

Source of this DR was actually PJ, though it is noted as Plum/Plm.

Consensus at this meeting is no change required, move to REVIEW.

DR 334

This is a new defect from this meeting. See N1203.

Missing semantics of comparison macros.

The paper suggests adding a constraint violation. General feeling is that this is inappropriate. Current compilers (using extensions) accept this, and a constraint violation would render their extensions illegal.

Some people believe that there is no question that these comparison functions are underspecified.

Current wording says "In the synopses in this subclause, real-floating indicates that the argument shall be an expression of real floating type." This is well defined. (see 6.2.5 para 10). However, several people believe there should be explicit language explaining whether or not the two arguments to a comparison macro must be the same real-floating type or allowed to be different real-floating type.

There are no suggested words for a proposed TC at this time.

Fred asked for the following to be added to the Committee Discussion: IEEE-754 (and IEEE-754R) require that comparison operations work in all supported formats, even if the operands' format differ.C99+TC1+TC2, Annex F.3, printed page 445, first bullet says that the comparison macros (along with the relational and equality operators) are the IEC 60559 comparisons.

Leave OPEN.

DR 335

This is a new defect from this meeting. See N1204.

_Bool bit-fields.

Also related to DR315 (leave things implementation defined).

To be consistent with DR315, it should be implementation defined how a _Bool bit-field is represented.

Looking at the first question: What is the maximum width of a _Bool bit-field allowed that does not cause a constraint violation? The standard does not specify the size of a _Bool. _Bool has a lower rank than char. However rank is not used in the sizing of an unsigned integer type. DR 262 (published in TC2) says "The expression that specifies the width of a bit-field shall be an integer constant expression that has nonnegative value that shall not exceed the width of an object of the type that is specified if the colon and expression are omitted."

It used to say "number of bits", the intent of the change was to exclude using the padding bits, but only the value bits. 6.2.6.1 para 4 talks about the values stored in bit-field objects.

A _Bool can really only hold the values 0 and 1 (see 6.2.5 para 2 and 6.3.1.2 para 1). Other bits are padding. Perhaps implementation defined if bigger widths could be supported. Certainly no-one could portably rely on anything bigger than width 1.

Things like memcpy could end up putting other non-zero but bigger than width 1 values into an object of type _Bool (or an object containing a _Bool bit-field).

Question is simply how wide can you make a Bool before you get a diagnostic. You can take sizeof(_Bool), and address of, etc. So it is reasonable to expect that it is at least CHAR_BITS wide in reality.

Anything wider than 1 should be implementation defined.

6.7.2.1 para 9 also seems to support the idea that a _Bool bitfield is really just a _Bool.

A _Bool bit-field should have the semantics of a _Bool (and not an unsigned int). It is unclear what conformance category any _Bool bitfield width > 1 falls into.

The width of a _Bool bit-field is at most the implementation defined width of the type _Bool.

Should also consider repudiating DR262. However, see http://open-std.org/jtc1/sc22/wg14/10778.

Proposed TC: None.

Committee Discussion:

The width of a _Bool bit-field is at most the implementation defined width of the type _Bool. A _Bool bit-field has the semantics of a _Bool (and not an unsigned int).

Leave OPEN.

DR 336

This is a new defect from N1206, arising from the Austin Group. TMP_MAX.

Is this an invalidating change? Does it break any existing systems? This is not intended to be so. There is a suggested TC. This also affects TR24731-1. Copy the suggested TC to Proposed TC and move to REVIEW.

DR 337

This is a new defect from N1207, arising from the Austin Group. Macro definition problem in stdio.h.

FOPEN_MAX is required to be at least 8 (7.19.3 para 13). So FOPEN_MAX does not require any additional words.

BUFSIZ likewise must be at least 256 (7.19.2 para 7).

FILENAME_MAX 7.19.1 para 3 requires that FILENAME_MAX must be at least 1.

Proposed Committee Response: All of these constants already have required minimum values that are positive, non-zero. No changes are required.

Move to REVIEW.

DR 338

This is a new defect from N1208. C99 seems to exclude **indeterminate value** from being an uninitialized register.

On some hardware (e.g. Itanium), an 8-bit value may have as many as 257 different values (0-255 and a “Not a Thing” value). However, c99 explicitly forbids such a value for an unsigned char.

Fred believes 5.1.2.3 para 12 shows our intent.

Values are independent of whether they are represented in a register or in memory.

Add to committee discussion:

Page 6, 3.17.2, change the definition of "indeterminate value" to:

either an unspecified value or a trap representation; or in the case of an object of automatic storage duration whose address is never taken, a value that behaves as if it were a trap representation.

5.1.2.3 para 5 second bullet speaks to this also.
Leave OPEN.

DR 339

This is a new defect from N1220. Variably modified compound literals.

NB the paragraph in question (6.5.2.5 para 3) is deleted by DR 328.

Everyone agrees that such code should violate some constraint.

6.7.2.1 para 8 (and footnote 103) says that identify-ability matters.

Expressions with variably modified types do exist, and you can cast to a variably modified type, so a variably modified type is an ordinary identifier.

Committee Discussion should say:

The suggested TC looks appropriate. However, this may be redundant.

Leave OPEN.

DR 340

This is a new defect from N1221. Composite types for variable-length arrays.

Implementations differ in the way they interpret the current words.

See reflector messages 11145-11147 for discussion.

VLA should not be treated differently than one of a known constant size. Suggested TC will make this clearer. The change will make the following code unambiguously invalid:

```
typedef void (*T1)(int);
typedef void (*T2)();

int x, y;
void *p, *q;

void
f (void)
```

```
{
  T1 (*a)[] = p;
  T2 (*b)[x] = q;
  (y ? a : b)[0][0]();
}
```

Suggested TC affects the first element of the dash list in 6.7.2 para 3:

— If one type is an array of known constant size, the composite type is an array of that size; otherwise, if one type is a variable length array, the composite type is is an array of that size; in either case, the element type of the composite type is the composite type of the two element types

Nick Stoughton 4/25/07 2:44 PM
Deleted: that type

Douglas feels more word-smithing is needed.

Committee Discussion.

The element types are composed. Suggested TC is close, but not quite right. The example from 11145 should be included here.

Leave OPEN.

ACTION Joseph Myers to provide updated TC for DR 340.

DR 341

This is a new defect from N1222. [*] in abstract declarators.

Parameter declarations are such even without an identifier. This answers the first question affirmatively.

All agreed that first example is/should be valid. Second is/should be invalid. “Attractive” wording still needed.

ACTION: Randy Meyers and Joseph Myers to propose new words for a TC for DR 341.

Leave OPEN.

DR 342

This is a new defect from N1223. VLAs and conditional expressions.

NB DR#XXX-2 refers to DR340.

There are no rules for specifying a composite type when it depends on an expression and the expression is not evaluated. The proposal makes an implicitly undefined behavior explicitly undefined.

General consensus is that implicitly undefined is satisfactory, and we should issue an RR stating this (therefore making it explicit but only in an external document).

Proposed Committee Response: The standard does not speak to this issue, and as such the behavior is undefined.

Leave OPEN.

DR 343

This is a new defect from N1224. Initializing qualified wchar_t arrays.

General agreement that this is non-controversial. Copy Suggested TC to Proposed TC and move to REVIEW.

DR344

On the wiki as dr_344.htm, coming from WG21. Casts in preprocessor conditional expressions.

General agreement that this is non-controversial. Copy Suggested TC to Proposed TC. Late arrival so leave OPEN.

DR 345

This is the new defect submitted during the meeting from Rob Arthan. See arthandr.htm on the wiki.

Q1: The committee believes that 6.9.1p9 is a comment on the storage duration and does not override the lexical scope described 6.2.1p7.

Proposed Committee Response:

As above.

Q2 needs Proposed TC: not ready yet.

Leave OPEN.

Defect Summary

N1191 shows TC/RR status of all closed defects before this meeting. Create TC3 based on these PLUS DRs 298, 311, 322, 323, 324, 332, 333. Some of these were closed at this meeting (298, 332 and 333). Blue lines are recent closes. Yellow lines are TC type reports that have been closed for a long time. Two documents will result: RR1 and TC3. N1124 + TC3 will be used as the base document for C1x. ACTION Convenor to add change to fwprintf to match the change for fprintf %g made in TC2 to TC3.

ACTION Fred, Derek, David to review TC3 draft when available.

11. Separate WG14 administration (Benito) and J11/U.S. TAG meetings (Meyers, Walls)

The registration ballot for TR 24731-2 closed, with 9 approve votes, and 0 no votes. ACTION Nick to prepare draft words for an additional “asprintf” function for this document.

For J11 minutes, see end of document.

12. Defect Reports, continued.

(***Note:*** *NA - See Individual Defect Report Items*.)

13. Potential Extensions For Inclusion In a Revision of ISO/IEC 9899

(N1229)

JB introduced the topic and asked if anyone was prepared to speak against revising. Willem asked for justification for a revision, particularly with respect to backward compatibility. The current language needs cleaning up, not just extending. Embedded systems need less, not more. Should remove floating point and complex arithmetic.

Clark's biggest concern about opening up the standard for revision is the shortage of progress that Microsoft has made toward C99 compatibility. What's the point in extending the language if existing vendors haven't caught up with ten year old stuff? PJ spoke in Microsoft's defense; there is no customer demand for some features. We should consider removing some poorly adopted features. There are three classes of things in C: things that have been there forever, work, well, and fully understand. There are some portions that have been added that tidy up loose ends (e.g. _Bool and VLAs). There are other things that just aren't worth it. There is a burning need for a revision based on concurrency requirements from multi-core processors, to fix up some known weaknesses, and to adopt common existing practice.

Joseph talked about subsetting, and noted that there is already language to permit certain subsetting particularly with regard to complex and certain floating point support.

Customer demand should be the driving factor.

Bill said options can end up with hundreds or thousands of different “conforming” systems that are all different.

There are several DRs that effectively ask for new features.

Randy asked for vendor feedback.

Douglas responded with “No invention”. Don't change existing practice to bring in some feature. Must be a widely implemented, widely understood feature. Threading support, dynamic libraries, security features.

JP motivation is to find cross platform portability. C has lost relevance, too many important extensions needed to write sensible, large programs. Wants at least 2 implementations of a feature before it is adopted.

Tom spoke about C++ experience, including being experimental, breaking new ground and inventing features out of whole-cloth. Responsibility of this group is to be more restrictive. C is the more reliable language. The language with less room for argument (!!) - focus on the practical, useful features for most users. Our

language is more reliable. If we adopt any inventions, they should have been implemented. Don't want users to have to jump to C++ to get some small feature of that language we don't support. C99 probably went too far. We probably have enough mathematical support. POSIX was the driver for C99 adoption ... when POSIX mandated it, a lot of vendors felt compelled to get their compilers up to scratch.

Rich: One of the things that help people to write good code is good diagnostics. We have some good constraint conditions. But we also have a lot of undefined behavior. Some of these could move toward diagnostic requirements.

It was suggested that better portability warnings from compilers should be addressed.

C++ often regards the “must be implemented” as a joke.

Need to be careful about use in user namespace ... should we use _ or not?

We also need to ensure that we clean up the corner cases of new features coming in, however well defined they may be.

Another big area where we should be looking to cover is to harmonize MSVC and gcc.

We need to create a process, not everything will be accepted. Mistakes were made last time, learn from them. Last time one mistake was thinking our competitor was Fortran. Security, parallelism, dynamic libraries, vendor specific additions. Also extended characters and embedded systems. Our charter for the C9x revision was basically good, and we should re-adopt it, at least in spirit. See http://www.open-std.org/jtc1/sc22/wg14/www/charter.

The process is extremely important. We are a JTC 1 working group, and we should be careful to follow their rules.

We do need better Unicode support, and should include the extended character TR.

Arjun asked what users expected from the new revision. Would they convert old code to new?

Thread-local storage and statement expressions are two big user pushes.

Must have a large value gain to make changing worthwhile. New code might do things the new way, but little code will be retrofitted.

“Strict” mode is generally used to improve portability. Get better diagnostics. But loses lots of valuable extensions. Controls for data-layout, alignment, packing etc are critical missing features.

There is a lot of legacy code. Must not break this, or at least too badly. There is a huge base of 8-bit systems that use C; embedded community is really there, really big. Very short production cycles.

Is IEEE 754 binary floating point used by Cisco or Oracle? Essentially no. And decimal probably won't help.

Many embedded programmers do not want C99 features, they reduce portability.

POSIX has some subsetting/sub-profiling rules that have worked for them. They might be worth consideration.

Tom brought up declspec v attribute (MSVC v GCC); lots of useful attributes. C++ has gone with declspec, though it is a late inclusion, and might drop out. How do we deal with implementations that have done similar things but used a different syntax? C++ are using [[*attribute*]] – i.e. neither of the existing practice names. Allows for vendor extensions to the set of attributes. We should co-ordinate closely with C++ here.

On this topic, is it the job of the standards group to pick a winner? Why did they end up with different syntaxes? Is it our job to harmonize. Some say yes.

_Pragmas could have some overlap with attributes, but are not popular. But pragmas don't integrate well with the language. Pragmas are regarded as vendor extension space. Attributes are not perfectly designed/documented at this point either.

Should bounds checking functions be included?

Adoption rate of C99; POSIX has been the biggest single driver. Most customers haven't really moved. GCC compatibility is probably biggest driver for most compiler vendors.

Migration is a big issue. Must be able to mix and match C89, C99 and C1x based code easily. It has been easy to creep into C99 from a user perspective. Inline declarations, in-loop declarations, // comments were easy early wins. VLAs popular.

Namespaces (as in C++) are pretty attractive. We rejected these in C99. Looks like they might be low-hanging fruit, but C++ have many issues. Are they implemented in C? No, just C++.

JB notes that nobody is speaking against the idea of a revision. First pass should be a cleanup one, deprecating or removing unwanted features. Produce a document with the plan ... what should go, what should be added. Existing practice should be in a shipping compiler ... a commercial implementation, not an experimental one.

Each new feature should be covered by a single document, sort of like a TR. Covering all cases, using standardeze. Folding them in is the final step. Proposals could be built on a wiki. Most important is that proposals are complete.

David Keaton said don't gate the revision by the pruning of the previous version. Some removal things may take a long time to agree on. Implicit int removal in C99 was a good example. Also separate proposals can lead to problems with the interaction of two; Compound literals and VLAs were two separate proposals. Both well specified independently. Missed a lot of cases in the overlap of the two.

JB says when we open the door to the revision; we know exactly what the base document is. In particular, TC3 is a big gate.

JP asked when cleaning up the document are we talking about improving the language, or removal/deprecation? Both.

Undefined v unspecified v imp def. Tom believes we should improve on this. For example some implementations all do the same definition of “undefined” for some feature. Other undefined's are really portability issues. Static analysis tools have a hard job with weakly defined things. These (undefined behaviors) end up being the real expense of most static analysis tools.

Should also consider the real hardware profile we are targetting. Should we support one's complement, 9-bit systems?

Most of our “mistakes” weren't really mistakes. C is supposed to be efficient. Must give implementers wiggle-room to do things efficiently. Undefined is used for this a lot. Use undefined, unspecified, and imp def. We need to use undefined more judiciously.

POSIX defines a byte as 8 bits. It also defines a void * as being able to hold a pointer to a function.

There is a small list of “critical undefined” behaviors ... buffer overflow, bad pointers etc. Should work on defining this list as small as possible, and use appropriate language for these. Tom has a draft paper on this. ACTION Tom to submit a paper on critical undefined behavior.

David Keaton said people look to us for guidance. E.g. Looking for guidance from the language in designing new hardware. Hardware vendors want to be compiler friendly. Compatibility with legacy hardware will continue to be an issue, but we should certainly consider dropping support for platform types that simply don't exist any more.

LUNCH

Looking at the charter, Point 1 is about supporting existing code. New keywords should be permissible (e.g. typeof). The idea is that the burden of migration should be very small.

Point 6 is the most intriguing to JB: Keep the spirit of C.

Some of the facets of the spirit of C can be summarized in phrases like

(a) Trust the programmer.
(b) Don't prevent the programmer from doing what needs to be done.
(c) Keep the language small and simple.
(d) Provide only one way to do an operation.
(e) Make it fast, even if it is not guaranteed to be portable.

Points a and b are directly at odds with the security and safety aspects. Perhaps something along the lines of “I don't trust myself, please check me!”.

Point b is felt critical.

Tom suggests the language should be capable of being verified. There are some specific proposals being discussed that may deal with limit enforcement ...

saturated integer arithmetic for example. We don't want overflow exceptions in the language, but saturation can help.

Joseph suggests adding an item to the list to support safety and security rather than remove the existing points. The language should not be accident prone. Derek argues that C is the way it is, and saturated integers don't really solve the problems. Add an additional point to the “Additional Principles” list. Lois asked are we more concerned with Murphy or Machiavelli? Both.

ACTION John Benito to rewrite the C9x charter as the C1x charter for the post London mailing. All to review.

Who are we writing the standard for? Compiler writers or programmers? When we came to revise C in the 90's it was the dominant language, but is it still? Who are the competition, and what should we learn from them?

A lot of C++ is C with classes. Same with C#.

Should a standards committee care about competition at all? In some senses we should be thinking in marketing terms: who are our constituents? How can we win back those who went away because of the overreaches in C99? We do have a responsibility to lead the way as the basic, fundamental programming language.

Where did the original success or C come from? Being close to the hardware while embodying some higher level techniques.

We should avoid adding features most of our community doesn't need. The embedded community is a major customer. They need better static analysis tools. The market is too fragmented for tools vendors to support. How many have implemented the embedded TR as it stands? GCC. EDG. ACE. DinkumWare.

Implementing the Embedded TR as part of the revision is possibly too much detail for this meeting today. Might be a profile. Should consider some sort of sub-setting for the embedded community.

We are run by the people who show up and champion proposals. If you care about a feature that you think should be in the revision, you will need to drive it through.

Walk through N1229 looking for anything we don't want. ACTION Arjun and JB to prepare a similar paper based on MSVC extensions that might be appropriate for standardizing.

ACTION PJ to produce a paper on EDG extensions that might be appropriate for standardizing.

- Statements and Declarations in Expressions. MSVC doesn't have this. Has been discussed in C++. Important for function-like macros. Inline functions have many shortfalls. This fills a need.
- Locally Declared Labels. Needed.

- Labels as Values. Some opposition. At one point in the distant past they were in some other implementations, but not generally thought useful. There are code examples that use these. REMOVE!
- Referring to a Type with typeof. Important.
- Conditionals with Omitted Operands. REMOVE.
- Case Ranges. Has its uses. Not critical. REMOVE.
- Inquiring on Alignment of Types or Variables. Critical.
- Thread Local Storage. Critical.
- Attribute Syntax. Critical.
    - Declaring Attributes of Functions
    - Specifying Attributes of Variables
    - Specifying Attributes of Types
- The try-except Statement. Don't want exception handling in C. Different from C++. Don't throw it out yet. Don has pre-processor support for something like this. Use (or not) of volatile introduces subtle errors; this would help remove some of this. Clark has misgivings; UNIX / Linux users don't seem to need it. No clean migration path. Bad interaction with setjmp/longjmp. How common is usage of this? Too many semantics to nail down to feel comfortable about it. Should have a very high hurdle. Arjun addressed the usage question: in some classes of code, it is very frequently used. What is an exception?If it is done well, it could be very powerful. Interaction with multi-threading also has issues. Can be done without this machinery, but this machinery makes it look better. Lots of reservations.
- The try-finally Statement. Solves the same problem as attribute(cleanup). This and try-except deserve more work, but have potential merit.
- Branch prediction. Useful? There may be utility, but many users may get it wrong. Not intended for novice users, this is a kernel/library/embedded system support. This is like “register”. No. Derek argues that branch prediction is a solved problem, and we don't need support for it. Like restrict – programmers may get it wrong. No general support.

14. *Open for any unfinished business*.

15. Administration

15.1 Future Meetings

15.1.1 Future Meeting Schedule

October 8-11, 2007, Kona, Hawaii
April 14-18, 2008 Delft, Netherlands. NEN as host.
October 2008, Boulder Colorado. Cisco as host.

The April meeting will not be synchronized or co-located with the C++ committee (nor is it likely that later meetings will be for some time). NB C++ are planning to meet in Nov 2008 in Silicon Valley. We might reconsider and move to Silicon Valley to co-locate when the C++ dates are known. During a revision of both C and C++, it is particularly important to keep the two groups closely aligned. Should we move to a three times a year schedule for the revision? Teleconferences have proved very effective for the Austin Group. We can spend several hours discussing the width of a single bit ... will one hour teleconferences work? Use a single wiki rather than one per meeting.

15.1.2 Future Agenda Items

15.1.3 Future Mailings

Post London- May 28.

Pre Kona – September 10

15.2 Resolutions

15.2.1 Review of Decisions Reached

15.2.2 Review of Action Items

ACTION: Ulrich Drepper and a small team to write a liaison report on threads to WG21, in response to Lawrence Crowl's presentation, N1196. JB, Nick, Tom Plum, and Bill Seymour to work on this with Ulrich.
OPEN

ACTION: Convenor to change the C99 Rationale as proposed in N1189.
OPEN

ACTION: Convenor to forward DTR 24732, as revised by the Edison Kwok editorial group, to SC22 for DTR ballot.
OPEN

ACTION Clark Nelson to write up preprocessor issue with casts from WG21 in DR format. This will become DR344.

DONE

ACTION Fred to provide additional (potentially troublesome) examples for the Decimal FP input functions.

DONE

ACTION Bill Plauger and Rich Peterson to propose new words for N1216, unsuffixed floating point numbers and TTDT. Editorial Group is PJ, Fred, Jim Thomas, Janis Johnson, and Rich Peterson.

OPEN

ACTION: Derek Jones to make a pass through the C standard itself to ensure that the terms “format” and “encoding” are not used in any inconsistent manner, especially with respect to IEEE 754r.

OPEN

ACTION PJ to produce a rationale for N1182 before the FPDTR ballot.

ACTION Chris Walker to investigate sensible constraints for the beta function, and make recommendation for disposal of comment 9 in N1185.

OPEN

ACTION: Bill, Fred, & JB to review changes made to N1182 and the rationale when the updated document is available.

OPEN

ACTION: Convener to forward the updated Special Math TR for FPDTR ballot when ready.

OPEN

ACTION Randy Meyers, Robert Seacord and Nick Stoughton to write additional rationale for the issues raised in N1210.

OPEN

ACTION Tom, Nick, Mark, David to review N1209 (CERT Secure Programming Guidelines).

OPEN

ACTION Joseph Myers to submit his words for DR314 as a proposal for C1x in the post-London mailing.

OPEN.

ACTION Joseph Myers to provide updated TC for DR 340.

OPEN

ACTION: Randy Meyers and Joseph Myers to propose new words for a TC for DR 341.

OPEN

ACTION Convenor to add change to fwprintf to match the change for fprintf %g made in TC2 to TC3.

OPEN

ACTION Fred, Derek, David to review TC3 draft when available.

ACTION Convenor to forward TC3 to SC22 secretariat for ballot when ready.

ACTION Convenor to write rationale for changing the SC 22 programme of work to make TR 24747 special math to IS. Bill to review.

ACTION Barry Hedquist to take the minutes at the next meeting.

ACTION Nick to prepare draft words for an additional “asprintf” function for TR24731-2.

ACTION Tom to submit a paper on critical undefined behavior.

DONE

ACTION John Benito to rewrite the C9x charter as the C1x charter for the post London mailing. All to review.

ACTION Arjun and JB to prepare a paper based on MSVC extensions that might be appropriate for standardizing.

ACTION PJ to produce a paper on EDG extensions that might be appropriate for standardizing.

15.2.3 Thanks to Host

Acclamation for Neil Martin. Another successful meeting!

15.3 Other Business

6. Adjournment

Adjourned at 15.40.

Agenda for the J11/U.S. TAG Meeting, Tuesday April 24th 4:53pm

Attendees

John Benito	Blue Pilot	
Randy Meyers	Silverhill Systems	J11 Chair
Douglas Walls	Sun Microsystems	J11 IR
Fred Tydeman	Tydeman Consulting	J11 Vice Chair
David Schwab	Oracle	
P. J. Plauger	Dinkumware, Ltd	
Tana L. Plauger	Dinkumware, Ltd	

Nick Stoughton John Parks	Usenix Intel	Secretary
Clark Nelson	Intel	
Bill Seymour Arjun Bijanki	self Microsoft	
Tom Plum Christopher Walker	Plum Hall Dinkumware, Ltd	
Keith Derrick	Plantronics	
Rich Peterson David Keaton	Hewlett Packard self	
Edison Kwok	IBM	
Robert Seacord	CERT	

1. 15 voting members present, the total voting members are 18.

2. Select US delegation for the next two meetings.
Not required at this meeting.

3. INCITS official designated member/alternate information.

Be sure to let INCITS know if your designated member or alternate changes, or if their email address changes. Send contact info to Lynn Barra at ITI, lbarra@itic.org.

4. Anti Trust

INCITS J11 members are reminded of the requirement to follow the INCITS Anti-Trust Guidelines which can be viewed at http://www.incits.org/inatrust.htm.

5. Five year maintenance of the Numerical C Extensions document.

Time to renew/revise/withdraw this document. Deborah Spittle said it didn't matter to INCITS if we keep it or kill it. This is INCITS TR17. Originally published in 1997, and hasn't changed since then.

Motion to recommend that INCITS/TR-17 “Numerical C Extension” be reaffirmed for another 5 years (Keaton). Second (Peterson). Motion requires 2/3rds vote. Vote is 15-0-0-3-18.

6. Adjourn at 17:03.