SUMMARY OF VOTING ON

Letter Ballot Reference No: SC22 N3574
Circulated by: JTC 1/SC22
Circulation Date: 2003-05-01
Closing Date: 2002-08-01

SUBJECT: Summary of Voting on SC 22 N 3574 - Second Letter Ballot for ISO/IEC PDTR 18037 - C Extensions to Support Embedded Processors

---

The following responses have been received on the subject of approval:

"P" Members supporting approval without comment

8 (Canada, China, Czech Republic, Denmark, Italy, Republic of Korea, Norway, Russian Federation)

"P" Members supporting approval with comments

3 (Japan, Netherlands, USA)

"P" Members not supporting approval

2 (Switzerland, UK)

"P" Members abstaining

2 (Austria, Germany)

"P" Members not voting

10 (Belgium, Brazil, Egypt, Finland, France, Ireland, DPR of Korea, Romania, Slovenia, Ukraine)

_____ end of summary, beginning on NB comments _____

Japan

This PDTR does not have the formal cover letter on which the information of the ballot (e.g. the type of the TR) must be described. It should be accompanied with the letter ballot.

Netherlands

Section 7.18a.3 (page 27), 3rd para: replace "The values given below" by "The integer values given below". Rationale: the current text requires the pre-processor to be able to do fixed-point arithmetic; this was never the intention.

1 **MB**[1]	2 **Clause No./ Subclause No./ Annex** (e.g. 3.1)	(3) **Paragraph/ Figure/Table/Note** (e.g. Table 1)	4 **Type of com-ment**[2]	5 **Comment (justification for change) by the MB**	(6) **Proposed change by the MB**	(7) **Secretariat observations** on each comment submitted
CH	General		ge	These comments are the result of an attempt to implement the different versions of the chapter 4 "Basic I/O Hardware Addressing" of the TR since the Copenhagen meeting in 2001. This chapter has changed considerably in the course of its development, and major modifications were introduced between the first and the second SC22 ballot. Generally, these modifications are very positive and make the background and interface usage of I/O HW addressing much clearer. But unfortunately, some specific changes in interface definitions make the usefulness as well as the efficient implementability of the interface very problematic. Actually, while an efficient implementation of interface from the TR of the first ballot was successfully finished, it turned out as being essentially impossible to create an efficient implementation of the interface as specified in the TR for the second ballot. This does not say that we object the whole restructuring of chapter 4. But some specific modifications changed the actual substance of the hardware addressing interface without any given plausible technical rationale.		
CH	4.5		te	The proposed 7.8a.1p1 (p.64) states: "An I/O register is accessed (read or written) as an unsigned integer." This can be misleading. There might be no unsigned integer type that can accomodate the value of an I/O register. I.e. though the underlying register is generally indeed treated as an unsigned interger value, the actual C data type to hold it might be something different. E.g. if the register is 128 bit wide, but the largest unsigned integer type available holds only 64 bit, one needs a struct to hold the value. (Arrays are generally also possible but	Write the problematic sentence as: "An I/O register is accessed (read or written) as an unsigned integer value[1]." "Note 1: This does not necessarily imply that the type used is actually one of the set of unsigned integer types provided by the compiler."	

1 **MB** = Member body (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial
NOTE Columns 1, 2, 4, 5 are compulsory.

1 **MB**[1]	2 **Clause No./ Subclause No./ Annex** (e.g. 3.1)	(3) **Paragraph/ Figure/Table/Note** (e.g. Table 1)	4 **Type of com-ment**[2]	5 **Comment (justification for change) by the MB**	(6) **Proposed change by the MB**	(7) **Secretariat observations** on each comment submitted
				problematic as they can not be passed by value.) Or if the register is 24 bit wide, but there is no unsigned integer type of that width, one might also want to hold the value in a struct to avoid unnecessary conversions or paddings.		

CH	4.5		te	The proposed 7.8a.4 (p.67) defines the register access interface as functions. Though the proposed introduction to 7.8a (p.64) and also 4.4.2 (p.59) allow the implementation as macros, 4.4.2 explicitely requires that a macro-based implementation provides exactly the same effects as the function implementation.
				
				This is a major difference to the previous version that generally defined the interface as macros and allowed the implementation to use (possibly a special kind of) functions. The problematic difference is the definition and handling of parameter types. (Of course, this discussion applies to return types as well.)
				
				Macros allow any type as arguments, even different argument types for the same macro, and take the arguments as given. But functions are defined to take exactly one argument type for each parameter, and convert any other argument types to the parameter type. Even worse (in this specific case), functions require integer type arguments at least converted to (unsigned) int, i.e. there is no use in defining functions with parameter types of unsigned char or unsigned short. So, an 8-bit value is always converted to an integer at least once for each read and each write, which causes an overhead that is always annoying and sometimes forbidding. Though in theory a good optimizer could remove any unnecessary conversions, in practice this will rarely be the case, as the implementation of these operations typically implies the usage of specific assembler instructions that are generallyuntouched by optimizers.
					In the proposed 7.8a.4, all the function definitions are replaced by respective macro definitions, e.g. 7.8a.4.1:
					
					replace:
					
					Synopsis
					#include <iohw.h>
					unsigned int iord(ioreg_designator);
					unsigned long iordl(ioreg_designator);
					
					Description
					
					The functions iord and iordl read the individual I/O register referred to by ioreg_designator and return the value read. The I/O register is read as an unsigned integer of its size; the read value is then converted to the result type, and this converted value is returned.
					
					by:
					
					Synopsis
					#include <iohw.h>
					iord(ioreg_designator)
					
					Description
					The function like makro iord reads the individual I/O register referred to by ioreg_designator and returns the value read. The I/O register is read as an unsigned integer of its size; this value is returned	

1 **MB** = Member body (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial
NOTE Columns 1, 2, 4, 5 are compulsory.

1 **MB**[1]	2 **Clause No./ Subclause No./ Annex** (e.g. 3.1)	(3) **Paragraph/ Figure/Table/Note** (e.g. Table 1)	4 **Type of com-ment**[2]	5 **Comment (justification for change) by the MB**	(6) **Proposed change by the MB**	(7) **Secretariat observations** on each comment submitted

				Furthermore, the proposed interface functions are provided as int and long versions, so it's not possible to use these functions for I/O register values larger than an unsigned long.	without any conversion.
					or 7.8a.4.3:
					replace:
					Synopsis
					#include <iohw.h>
					void iowr(ioreg_designator, unsigned int a);
					void iowrl(ioreg_designator, unsigned long a);
					Description
					The functions iowr and iowrl write the individual I/O register referred to by ioreg_designator. The unsigned integer a is converted to an unsigned integer of the size of the I/O register, and this converted value is written to the I/O register.
					by:
					Synopsis
					#include <iohw.h>
					iowr(ioreg_designator, a)
					Description
					The function like macro iowr writes the individual I/O register referred to by ioreg_designator. If the unsigned integer value 'a' is of the same size as the size of the register, it is written to the I/O register without any conversion.
					If 'a' is of a different size than the size of the register, it is converted to an unsigned integer value of the size of the I/O register, and this converted value is written to the I/O register.
					iowr does not return anything.	

1 **MB** = Member body (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial
NOTE Columns 1, 2, 4, 5 are compulsory.

1 **MB**[1]	2 **Clause No./ Subclause No./ Annex** (e.g. 3.1)	(3) **Paragraph/ Figure/Table/Note** (e.g. Table 1)	4 **Type of com-ment**[2]	5 **Comment (justification for change) by the MB**	(6) **Proposed change by the MB**	(7) **Secretariat observations** on each comment submitted

CH	4.5		te	All functions in the proposed section 7.8a define a parameter "iogroup_designator" or "ioreg_designator". The types for these parameters are correctly left undefined.
				On a specific platform, these designators might have completely different types for different kinds of registers.
				But it is not possible (in C) to have the same function name for different types of the same positional parameter.
				Therefore, the designators must be unified to a common type which typically costs additional overhead. As this should be avoided, the function approach turns out to be wrong here as well.
					In the proposed 7.8a.3, replace the function definitions by respective macro definitions, e.g. 7.8a.3.1:
					replace:
					Synopsis
					#include <iohw.h>
					void iogroup_acquire(iogroup_designator);
					void iogroup_release(iogroup_designator);
					by:
					Synopsis
					#include <iohw.h>
					iogroup_acquire(iogroup_designator)
					iogroup_release(iogroup_designator)	

CH	general		ge	An efficient (zero-overhead) implementation of the interface according to the proposed changes was successfully accomplished.		
CH	general		ge	The disapproval will be changed to approval if the proposed changes are accepted.		

1 **MB** = Member body (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial
NOTE Columns 1, 2, 4, 5 are compulsory.

UK

Generally that we should not build on the maths of C99 until the current problems in the standard are fixed.

Note that the majority who were against this PDTR work in the embedded field.

USA

The US National Body votes to Approve with comments ISO/IEC PDTR 18037.2 - C Extensions to Sup port Embedded Processors, SC22 N3574. Comments are listed below.
Comments:
_X___ technical:
p.27, 2.2/7.18a.2: "If there is no ..." points out a glaring need for test macros so that applications can determine when there will be a problem using such a typedef. HAS_INT_R_T for example, with no requirement to provide a typedef when the corresponding macro is not defined.

7.1.3 (overflow and rounding): Is there any relationship between the rounding done with floating-point numbers and the rounding done with fixed-point numbers? If they are independent, is there a way to determine and/or alter the fixed-point rounding method (similar to FLT_ROUNDS or fesetround())? Is the rounding method for fixed-point static or dynamic?

7.18a.6.8 (strto*): What is the order between rounding and negating? To match the spirit of IEEE-754, negating should come before rounding. What is "correctly rounded" for conversions from decimal to fixed-point numbers? Currently, that term is only defined for floating-point numbers. Does it have the same meaning?

_X___ editorial:
p.5, Introduction para.4: Sentence beginning "In order to allow" should be adjoined, with a comma, to the following sentence.

p.6, 1.1 Scope para.2: "standard, necessary" should be "standard necessary".

p.7, 1.3 Conformance: "free standing" should be "freestanding".

pp.9-10, 2.1.2 Spelling ..., the requirement or lack thereof for aliases vs. the keywords is not clear; in particular, "redefined" raises questions and "or to another spelling" suggests that the natural spelling might not be defined. Suggest "... <stdfix.h>, these formal names are used to define the natural spellings as aliases, and may be used to define other spellings, for instance ..." (While I disagree with the notion that a standard header should define unspecified names not in

an implementation-reserved namespace, apparently this was already deliberately decided upon. It is not reflected in the normative wording, however!)

p.10, 2.1.3 Overflow ..., last line: insert "the" before "correct".

p.13, 2.1.6.2.1 para.2: There is an error in line filling (the second sentence should start right after the first).

p.14, 2.1.6.2.1 para.3: Another line filling error; also, "deprecate" is misspelled.

p.14, 2.1.6.2.1 para.5: "perform a multiply of" should be "multiply" and the following "and" would be better as "by".

p.14, 2.1.6.2.2: Change "type int" to "integer type" to match the actual normative text.

p.15, 2.1.6.4 Example: The braces { } are unnecessary.
p.18, 2.2 Detailed changes ...: "will get in the new document" should be "may get in the new document". It is possible that additional parent-section insertions change "mm", for example. (Also p.47, 3.3.)

p.18, 2.2/5.2.4.2.3 para.3: The radix "dot" is assumed to be between... what? Between implies two things, not just the one most-significant-digit. Perhaps "between or" should be deleted.

p.19, 2.2/5.2.4.2.3 para.4: Another line filling error, or perhaps this should be two paragraphs.

p.19, 2.2/5.2.4.2.3 para.5: "exact" should be "exactly".
p.19, 2.2/5.2.4.2.3 para.6: "maximal" should be "most" (three occurrences). "maximal negative" is simply wrong.

p.20, 2.2/6.2.5 paras.3-4: Suggest moving "_Sat" just before "_Fract" in each type, for consistency with other uses.

p.21, 2.2/6.2.5 para.4: The last use of "accum" should be italicized.

p.21, 2.2/6.2.6.3 paras.1-2: "values of any padding bits" should be "contents of any padding bits". (The term "value" as used in the C standard must carefully exclude padding.)

p.22, 2.2/6.2.6.3 para. 4 first item: "bits or" should be "bits as or" (optional commas around alternative).

p.22, 2.2/6.2.6.3 para. 5: Delete "where practical", which adds nothing and raises an unanswered question.

p.24, 2.2/6.4.4.2a Syntax: second form for decimal-fixed- constant should have "opt" suffix on exponent-part; otherwise 1k (for example) is not a valid fixed-constant. This also aligns with the strto* functions.

p.28, 2.2/7.18a.3 list: Expressions such as (-0.5HR-0.5HR) overflow and thus have undefined behavior. The true required minimum is something like (-0.9921875HR).

p.28, 2.2/7.18a.3 *FRACT_MAX, *ACCUM_MAX: The second (hex) form contains a spurious "C". Suggest deleting all these (redundant) forms, or correcting them and deleting the first forms.

p.33, 2.2/7.18a.4 Description: "ulp" is not defined in the changes to the standard. Suggest using small caps ("ULP", "ULPs") and defining it here (as in the footnote to 2.1.3).

p.33, 2.2/7.18a.4 Description: Change "multiply and divide" to "multiplication and division". Nouns, not verbs.

p.33, 2.2/7.18a.5 Description: "according to the set state" is unclear. Should this be "saturating"?

p.35, 2.2/7.18a.6.1 Returns: "saturated" seems wrong when there's no overflow. Suggest "saturated if it would overflow"?

p.35, 2.2/7.18a6.3: Suggest these be called "rounding" functions rather than "round" functions (also in 2.2/7.18a.6.7). Under "Returns", change "The functions" to "The round[ing] functions". If "round" is retained, the latter should be in Courier as used in the countls Returns.

p.38, 2.2/7.18a.6.6: Change "The bits functions" to "The above functions". (There are other functions whose names contain "bits".)

p.38, 2.2/7.18a.6.5 Returns: Change "bitpattern" to "bit pattern".

p.41, 2.2/7.19.6.1 'h': Should be insertion before the last semicolon, as specified for 'l'.

p.41, 2.2/7.19.6.1 'l': Change "semi-colon" to "semicolon".

p.41, 2.2/7.19.6.2 'h': Inserted text should start with "or".

p.42, 2.2/7.19.6.2 r,R,k,K: Should be a comma before

last "or".

p.45, 3.1.3 para.4, last sentence: Change "any" to "a null".

p.46, 3.2.2 para.1: Change "pre-defined" to "predefined".

p.46, 3.2.2 Examples: Make first occurrence of "char" Courier.

p.48, 3.3/6.2.4a para.1: Append "Unless otherwise specified, objects are allocated in the generic address space." (Even though this is covered in changes to 6.2.5, it needs to be emphasized here.)

p.49, 3.3/6.2.5 para 26, para.1: Last sentence should have appended "and address space qualifiers".

p.52, 3.3/6.5.16.1: Change "referenced type of" to "type pointed to by" (four occurrences altogether). "Referenced address space" can remain, but only because there is no better way to say that.

p.52, 3.3/6.7.1 syntax: Also add:
register-name:
identifier
p.53, 3.3/6.7.1.1 para.1: Change "Modifying" to "Accessing". (Read access might also have side effects.)

p.53, 3.3/6.7.2.1: Seems wrong; the members get qualified by the address-space qualifier of the aggregate. Suggest deleting this. (Unnecessary due to other requirements.)

p.53, 3.3/6.7.3 syntax: Also add:
address-space-name:
identifier
p.54, 4.1.1 first bullet: Change "market place" to "marketplace".

p.55, 4.2: Too many italics. I suggest changing all italics in this section to normal font, then change all bold to italic, which is consistent with usage in the C standard.

p.55, 4.2: The bullet "Multiple I/O registers may form an I/O group." is redundant with the next bullet and should be removed.

p.56, 4.3.1, last line: Change "interleave" to "interleaving".

p.57, 4.3.1: Change "<->" to a double-headed arrow glyph.

p.57, 4.3.2 figure: Change "users" to "user's" and

"vendors" to "vendor's".

p.59, 4.4.2: Should use so-called "smart quotes" around "function"; save the "straight double quote" for C code. (Also around "dense" on p.61, 4.4.3, "kind" on p.62, 4.4.5, "acquiring" on p.62, 4.4.6.1, "<iohw.h>" and "function on p.64, 4.5/7.8a, and possibly elsewhere.) This might be present in the PDF document, but was not evident in the selected font.

Annex A? Rationale for 6.2.6.3 should explain why only sign-magnitude representation is allowed. One would think that ones- or twos-complement adders would be faster.

Annex A? Rationale for 6.2.6.3 should explain why a signed accum type has to have at least as many integral bits as the corresponding unsigned accum type. One would think that it could reasonably be one fewer.

Annex C, general: "Interleave" is a verb, not a noun or adjective. Use "interleaving" for the latter. (Several occurrences.)

p.96, D.2.5 title: Change "users" to "user's".
p.99, E.4: Change "2's complement" to "twos-complement".