Minutes of X3J11.1 Meeting #6
(NCEG Meeting #11)
6-7 December 1993

Kona Hilton
Resolution Room
Kailua-Kona, Hawaii

Attendees

Harry Cheng, Frank Farance, Sam Figueroa, Jamie Frankel, Doug Gwyn, Rex Jaeschke (Chair), Bob Jervis, David Keaton, John Kwan, Tom MacDonald (Vice Chair), Mike Meissner, Tom Plum, Linda Stanberry (Secretary), Jim Thomas, Fred Tydeman, Neil Weidenhofer, Jeff Zeeb, Jonathon Ziebell.

Legend

The following symbols in the left margin of these minutes have the indicated meaning:

A	General approval
SV	Straw vote
MSP	Moved, seconded, passed (formal vote)
*	Action item

The activities reported here are grouped by subject and do not necessarily follow the exact chronological order of presentation during the meeting.

1. Opening Activities

1.1 Opening Comments, Goals and Purpose of the Meeting

The meeting was convened at 9:00 a.m. on Monday, 6 December 1993, by Chairman Jaeschke. MacDonald served as Vice Chairman. Stanberry served as meeting secretary.

Jaeschke announced that the goals and purpose of the meeting were essentially "business as usual," to continue work on each subgroup's proposals. Three have already been "forwarded" to X3J11, and four others are still pending. Another goal is to discuss how to continue after the merger of NCEG and X3J11, which will be effective in January.

1.2 Host Facilities

Plum Hall, Incorporated, was the host for this meeting. Plum gave information on local facilities and provisions for photocopying.

1.3 Approval of Previous Minutes [93-025]

Tydeman corrected the dates on page 19 of the "next" X3J11 meeting. They met on 1-3 of December.

229

A The minutes were approved without further correction.

1.4 Status of Action Items from Previous Meeting

Jaeschke reported that, although not indicated as an action item in the minutes, he had investigated making the rationale for the C standard available electronically, and it is now included with the NCEG documents available via anonymous ftp in directory `DMK/nceg` at

```
ftp.dmk.com
```

Jaeschke will notify Hal Computers, David Hough, and Microsoft of loss of voting status for non-attendance. Done.

MacDonald will forward 93-019 to X3J11, and to Keaton for ftp update. Done.

Jaeschke will talk to Prosser on status of the [compound literals and designated initializers] proposal, and ask him to send [it] to the NCEG reflector with the changes when it is done. Done, 93-024.

Meyers, Tydeman, and MacDonald should send their comments on FPCE proposal to Thomas electronically. Done.

Thomas will produce a cover letter with these comments and FPCE proposal with the approved amendments, and forward to X3J11 and to Keaton for ftp update. Done, although not yet sent to Keaton for ftp update.

Meyers and Kwan will develop a proposal for an <inttypes.h> mapping. Done, X3J11/93-056 (WG14/N309).

Meyers will revise and recirculate his paper on extended integer type issues. Not done, but subsumed by work with Kwan on <inttypes.h> proposed mapping.

Kwan will report the NCEG votes and positions back to the SPARC committee. Done.

Jaeschke will talk to Prosser and Ritchie to determine the status of the [fat pointers VLA] proposal. No response received, and Jaeschke didn't follow up on this.

Meissner will work with Prosser to produce a complete VLA proposal. Meissner has brought an updated proposal, but it is independent of the earlier Prosser/ Ritchie proposal.

MacDonald will include Plum's notice for the December meeting in the next mailing. Done.

Tydeman believed he had two additional actions items to answer: (1) Does a 48-bit pointer imply that an implementation must have a 48-bit integral type? and (2) How many digits must strtod() process? Both of these were submitted as defect reports to X3J11/WG14 and answered there.

230

1.5 Approval of the Agenda

The agenda was rearranged in order to accommodate new requests for agenda time for presentations. Papers for agenda items were identified. The DPCE subgroup announced an evening meeting for Monday night at 8 PM.

1.6 Introduction of Participants

Attendees introduced themselves, and identified their roles within subgroups. An attendatnce list was circulated and is attached to these minutes.

1.7 Distribution of New Documents

X3J11.1/93-034, "Data Parallel C Extensions," which was not available for the second mailing was distributed at the meeting.

New documents were assigned X3J11(WG14) numbers, and will appear in the next X3J11 mailing. There will not be any more X3J11.1 mailings. Available documents were distributed.

93-054 (N307)	"Why Infix Relational Operators," Thomas, distributed.
93-055 (N308)	"Hexadecimal FP Constants," Thomas, distributed.
93-056 (N309)	"Extended Integers," John Kwan and Randy Meyers, distributed.
93-057 (N310)	"Why Imaginary Types," Thomas, distributed.
93-058 (N311)	"Numerical Extensions to ANSI C in the C^H Programming Language," Cheng, distributed.
93-059 (N312)	"Fat Pointers," Meissner, distributed.
93-060 (N313)	"A Proposal for Standard C++ Complex Number Classes," Vermeulen, distributed.
93-061 (N314)	"Minutes of DPCE Meeting, 6 December 1993," Stanberry.
93-062 (N315)	"Minutes of X3J11.1 Meeting #6, 6-7 December 1993," Stanberry.

1.8 Identification of voting members

After the attendance list was consulted, MacDonald announced that there were 17 eligible voting members, and 10 were attending. 8 non-voting members were in attendance.

2. Status of proposal to merge J11.1 into J11

Jaeschke reviewed the status of the merger. The results of the unanimous letter ballot in favor of the merger from X3J11.1 members was forwarded to the OMC (formerly SPARC), who essentially gave their unanimous approval to the request. It is currently the subject of an X3 letter ballot, closing in January, for final approval. It was assumed for the rest of the discussion that there will be no objection to the merger.

Jaeschke clarified that NCEG/X3J11.1 was not "disbanding" as indicated, unfortunately, in the letter ballot; rather, the same group of heretics would be reconvening under a different banner.

2.3/

Jaeschke then discussed the organizational changes for handling NCEG issues under X3J11. As convenor of the J11 meetings, he will attempt to cluster the NCEG agenda items together for those who attend only those discussions. Only a few persons are affected by this, as most of the NCEG representatives are also the J11 representatives for their organizations and have been attending both meetings.

From now on NCEG documents will have dual X3J11 and WG14 numbers, to be obtained from P. J. Plauger (see information below on mailings). MacDonald asked about what should be done with X3J11.1 documents that were allocated, but have not yet been produced. These documents will need to be reassigned X3J11/WG14 numbers if they are going to be distributed in future J11 mailings.

As for voting rights after the merger, Jaeschke intends to propose that voting rights be formally extended to those few NCEG members who have not been J11 members, essentially counting their attendance at X3J11.1 meetings as attendance for J11 voting determination. Those members who have belonged only to X3J11.1 will have to become members of X3J11.

Plum confirmed that the committee has the right to make such exceptions to the usual attendance and voting rules to accommodate the transition from J11.1 to J11. He proposed that a specific list of persons to whom to grant voting rights be prepared for the next J11 meeting.

* Jaeschke will email both the NCEG and J11 reflectors to notify anyone who may be affected on the rules for attendance and voting after the merger.

3. Liaison Reports

<u>X3J11</u>

Jaeschke reported on the recent activities of the J11 committee which includes processing defect reports, publishing their first Record of Response and Technical Corrigenda, reaffirming their support for the Normative Addenda, considering NCEG documents, and voting to undertake a revision of the C standard. The international vote on the Normative Addenda at the WG14 meeting the previous week was 3 for, 0 against, and 1 abstention. The work for revising the C standard will begin with producing a charter proposal to set the ground rules for what will be considered, focusing on (1) C/C++ compatibility, and (2) prior art (including some or all of the NCEG technical report). Jaeschke has taken an action item to collect suggestions for what to include in the charter, and to produce the proposal for the next J11 meeting.

<u>X3J16</u>

Plum reported on the recent activity in the C++ committee. There has been a push to resolve many C/C++ incompatibilities, which included defining "plain ol' data structures" (PODS) that maintain C semantics under C++. There was nothing very specific dealing with NCEG issues to give as feedback, due mostly to the fact that the J16 subcommittees have too much already on their agenda to spend much committee time on the NCEG proposals. The following points were briefly mentioned:

The restricted pointers paper had been presented, but currently rejected by the extensions subcommittee.

232

A Complex proposal from Vermeulen of Rogue Wave was presented at the meeting in November. The proposal is consistent with the Knaak NCEG paper, and presents 'complex' as implementable as classes or as a builtin data type.

There was some discussion about the FPCE proposal with respect to C++ (Jervis, Thomas, Gwyn). Plum noted there is an incompatibility with the FPCE widest-need and C++ overloading rules–one or the other could be specified, but not both. Gwyn requested feedback from J16 on the FPCE overloading issues, and MacDonald asked for feedback on the controversial FPCE issues such as pragmas and relational operators.

Plum briefly discussed the time schedule for J16. The November meeting was supposed to be the last time to make new substantive proposals. In order to finish a committee draft on schedule, they can only consider organizational issues form now on. Hence any NCEG proposals would be considered just as "compatible extensions" from now on, but would not be part of the proposed C++ standard.

Plum expressed a concern with the work on a new revision of the C standard, as this will greatly increase the need for liaison activity between J11 and J16. Jaeschke suggested that C++ could freeze C as of now, and even consider the Normative Addenda after issuing their standard.

<u>X3H5</u>

Farance reported that he had attended the X3H5 meeting held in November, at which time they agreed to continue informal liaison with the DPCE subgroup. Jaeschke noted that the X3H5 chair is in danger of losing his officer status for failure to follow X3 rules (e.g., not submitting an annual report).

<u>X3T2</u>

Tydeman reported on the current status of the LIA (Language Independent Arithmetic) proposal, which is currently out for public comment. J11 agreed to forward Tydeman's remarks on the document as the J11 position, along with the question,"Is this needed?"

Tydeman will circulate his edited comments to the J11 reflector before forwarding them to T2. He plans to report on what else is happening in other SC22 groups with respect to the LIA and similar proposals.

There was some discussion of whether or not LIA requires IEEE754.

<u>POSIX</u> and <u>DSP</u>

There was no one present to report on any activity in these areas.

4. Designated Initializers and Compound Literals [X3J11.1/93-024]

Jervis led the discussion of the status of this proposal, presenting the updated proposal in 93-024. He reviewed the J11 meeting decisions, in which Plum had raised the issue of C++ compatibility: the functionality provided by compound literals should logically be provided by constructors in C++. Plum has a J11

2-33

action item to produce a paper addressing these issues, comparing the two approaches.

Jaeschke noted that the proposal could be split into two separate proposals; they had just been presented as one proposal up to now for organizational convenience, but are orthogonal proposals.

There was discussion (Gwyn, Jervis, Keaton, Meissner, Farance) questioning the intent that compound literals in a loop block would be allocated but not freed. The paper is explicit that this is the intent. It was noted that the storage could be reused if in a block: some confusion exists on the proposal's use of "function body" rather than block, and there was consensus that this needs to be clarified in the next revision of the proposal. The functionality of malloc() versus alloca() was compared. An explicit control mechanism for storage management is desirable. The motivation for compound literals is to provide a portable, convenient method for expressing structured values.

There was some discussion of the utility of the two proposals. Jervis believes that designated initializers are more important than compound literals. Farance believes that compound literals will be very useful for DPCE.

Plum opposes both, believing that neither are strongly motivated numerical extensions, and constructors/destructors buys you more, such as dynamic initialization of statics.

Farance noted that adding constructor/destructor functionality won't eliminate the need for current C initializers as in "int i = 10;". The proposal is a natural extension of what we currently have.

Jervis reported that in the ad hoc meeting held last week on what extensions J11 members would like to see in the next revision of the C standard, the vote was "14 Yes, 1 No" for compound literals and designated initializers, and "7 Yes, 4 No" for constructors/destructors. He further indicated he would support the proposal with changes to eliminate the surprise of infinite allocation and clean up implied with the current proposal.

Gwyn indicated he would support the proposal with changes of "function body" to "block."

MacDonald reminded us that Fortran has the designated initializer functionality, and we should provide the equivalent in C.

Jaeschke closed the discussion by suggesting that proposals for handling the scoping/allocation issues be emailed to Prosser. When Plum's paper is available, we should revisit whether this belongs in the TR for NCEG, or if it should be deferred to the J11 standard revision.

Keaton has a J11 action item to take over the work on this proposal.

* Keaton will take over work on the compound literal/designated initializers proposal, and pursue answers to the questions raised at this meeting.

5. Aliasing [X3J11.1/93-026, 93-031, 93-040, 93-041; X3J11/93-058 (WG14/N311)]

MacDonald gave an overview of the presentation given to J11, in which he had described the proposed edit in 93-040 to the restricted pointer proposal in 93-026:

Restricted pointer edit

• Delete section 1.6: aliasing of unmodified objects

• In the Semantics subsection of section 2, delete the phrase shown on the first line below:

~~If O is modified, then~~
all references to values of
O shall be through pointer
expressions based on P.

The intent

Consider types with two successive restricted pointer derivations, as in:

```
int * restrict * restrict p;
int * restrict * restrict q;
```

The intent was that p and q can be analyzed as if they were arrays:

```
int a[][n];
int b[][n];
```

The problem

Without the edit, there is an unintended distinction between references which modify the pointers referenced thru p and q:

```
*++p[i] = ...
*q[k]++ = ...
```

and those that do not:

```
p[i][j] = ...
q[m][k] = ...
```

Example 1

```
void f1(int n, int * restrict * restrict p,
               int * restrict * restrict q)
{
        int i, j;                   /* Without edit, */
        for (i=0; i<n; i++)         /* optimization  */
          for (j=1; j<n; j++)       /* is inhibited. */
             p[i][j] += q[i][j-1];
}
```

Without the edit, p and q may point to the same array of restricted pointers, since that array is not modified.

• The most natural usage does not promote optimization

Example 2

```
void f2(int n, int * restrict * restrict p,
               int * restrict * restrict q)
{
     int i, j;                      /* Without edit,   */
     for (i=0; i<n; i++)            /* optimization    */
       for (j=1; j<n; j++)          /* requires extra  */
          *++p[i] += *q[i]++;       /* analysis.       */
}
```

Without the edit, p and q may not point to the same array of restricted pointers, since that array is modified.

• The compiler must track which pointers are modified to establish the absence of aliasing that is in this example but not in the first example.

Example 3

```
void f3(int n, int (* restrict p)[n],
               int (* restrict q)[n])
{
     int i, j;
     for (i=0; i<n; i++)            /* Optimization */
       for (j=1; j<n; j++)          /* is easy.     */
          p[i][j] += q[i][j-1];
}
```

Here, since p and q point directly to the modified objects, those objects must be different (with or without the edit).

• The original intent was that f1 and f2 should be no more difficult to optimize than f3, and this intent is realized by the edit.

Minimal disadvantages

The disadvantage is only that function calls that result in aliasing of unmodified objects will now have undefined behavior, even in those cases in which no optimizations are at stake.

```
void f(int * restrict p, int * restrict q,
       int * restrict r, int n)
{
     int i;
     for (i=0; i<n; i++) p[i] = q[i] + r[i];
}
```

236

A call `f(a,b,b)` will be undefined (even though it will almost certainly give the expected results). The programmer can either un-restrict q and r, or define another version with one less parameter.

Jervis asked if Fortran has the same problem. MacDonald clarified that Fortran doesn't have pointers, only arrays.

Plum asked if the need for the edit reflects that the specification of restrict is still imprecise; that it hasn't been characterized carefully enough yet? Is the problem that we haven't located the motivating example yet? Or is there no motivating example? Could the problem be solved with const versus non-const distinguishing of pointers--e.g., in example 1, use

```
int const * restrict const * restrict q
```

Jervis and Keaton both argued that optimization in example 1 is not inhibited without the edit, and that the edit is not needed. After off-line discussions, however, they agreed that the edit was needed.

Gwyn suggested that the proposal needed to better distinguish what is meant by modifiability. MacDonald related that it is really hard to get the words right about walking through references with modifiability; he had tried before and abandoned it as an intellectual exercise!

Consensus at last was that CRI should proceed with the edit and present the revised proposal at the next meeting.

* MacDonald/Homer will produce revised proposal for aliasing.

Cheng believes that restricted pointers add too much to the language, and he presented alternatives to this approach from his papers X3J11.1/93-031 and 93-041, as summarized in X3J11/93-058.

<u>Alternative Solutions to Restricted Pointers for Anti-Aliasing</u>

1. Add do-loop. This has "fixed size" advantages.
2. Add assumed-shaped arrays, only arrays can be passed to assumed-shape arrays.
3. Add nested-functions that modularize the program and ease the detection of aliasing. The functions must be in the same file, which helps analysis.
4. Add array syntax without aliasing.

Jervis asked how his do-loop proposal handled aliasing–does it implicitly "restrict" the pointers? MacDonald suggested that you can capture aliasing and bound information before the loop, but this doesn't necessarily help alias analysis. Jervis indicated that the C semantics with respect to pointers shouldn't be different from Fortran because that would be confusing; the alias analysis should be the same.

Meissner itemized two objections to the alternative proposal: (1) nested functions introduce all sorts of problems (e.g., &nested_func implies getting context of the function), and (2) the proposal is too limiting, not general and separable from other proposals.

Jervis noted that assumed-shape arrays can't be dynamically allocated, and this is losing a C advantage over Fortran. Cheng pointed out that you can use deferred-shape arrays for this; assumed-shape arrays are not intended to be the only mechanism for passing arrays.

Gwyn and Thomas also voiced concern over the scope of the proposal and the fact that it addresses more than one problem, without separating them out into distinct proposals.

6. Complex [X3J11.1/93-020, 93-030, 93-035, 93-036, 93-048, 93-049, 93-058; X3J11/93-057 (WG14/N310)]

MacDonald presented revision 10 of Knaak's proposal for complex, X3J11.1/93-049.

• There are no diff marks in revision 10 to indicate the changes from revision 9, due to the extent of the changes in reorganizing the document to include the rationale in each section.

• This is intended to be CRI's final proposal.

The following additional changes to the proposal were described:

<u>Promotion Rules</u>

• Changed promotion rules back to:

When `real` *op* `complex, real` gets promoted to`complex`

• Rev 9 promotion rule only solved half of the *gratuituous NaN* problem–e.g., doesn't address:

`real` *op* `complex`

if one operand has a zero real part and the other contains an infinity (a grautitous NaN might be generated).

<u>Grammar</u>

Added grammar rules and terminology to clarify the distinctions between floating types, real floating types, and complex floating types. Last time had tried to fold types and the sense of the committee at that time was that these needed to be distinct.

• floating types:

real floating types
complex floating types

floating-constant:
real-floating-constant
complex-floating-constant

2-38

Octal and Hex constants cannot be suffixed with `i`.

```
        1i            /* ok still */
        0x1i          /* not ok   */
```

Note that this does not interact with Thomas' FPCE proposal for hex floating point constants as it uses different syntax.

Other changes

- changed section numbers from ANSI to ISO (ANSI)
- eliminated octal and hexadecimal complex constants
- changed the Rationale sections to reflect the current state of affairs and to prepare for possible publication in the Technical Report.

With respect to the revised promotion rules, Thomas noted that the sum of two imaginary numbers is always imaginary, independent of value. MacDonald will talk to Knaak about this.

Plum, acting as C++ liaison, requested that among the names defined in <complex.h> there should be synonyms added, or the names should be changed, to use underscores for float_complex, etc. After discussion, the consensus was to provide names without underscores because we have CRI prior art, but MacDonald will discuss it with Knaak.

* MacDonald will recommend changing the names defined in <complex.h> to use underscores to Knaak.

Thomas presented his latest revision of his complex proposal, X3J11.1/93-048, along with a Monday paper on imaginary types, X3J11/93-057 (WG14/N310):

- Complex support for Special Values
  - Infinities, NaNs, signed zeros
  - CCE (93-048) versus CEC (93-049)
    CCE extends treatment of special values from real to complex
    CEC does not handle special values
  - Key difference: imaginary type
- Semantic/efficiency problem without imaginary types
  Good:

```
    2.0i * (∞ + 3.0i) => - 6.0 + ∞i
```

2 multiplies

Bad:

```
    2.0i * (∞ + 3.0i) => (0.0 + 2.0i) * (∞ + 3.0i)
=> (0.0 * ∞ - 2.0 * 3.0) + (0.0 * 3.0 + 2.0 * ∞)i
=> NaN + ∞i
```

4 multiplies, 2 adds, and undesirable result

2-39

MacDonald disagrees with the conclusion, and believes that you can still achieve the optimization of number of operations. Gwyn noted that sufficiency requirement must be considered, too. Kwan noted that the implication is that no promotion rule applies for `complex` *`op`* `complex`.

Thomas next identified the following issues for comparing the proposals:

- Perfection or nothing. Belives the CRI position is that, since we can't handle all special values, don't handle any. Thomas believes that we can handle a significant number of the special value cases, however. MacDonald stated that this is not worth adding three new types. Thomas noted that this is not argued in the Knaak paper.

- Special cases are rare, which makes their careful treatment more important. Noted that they are rare from a data perspective, but not necessarily rare in code treatment. Farance believes that you would still need to have code that deals with special value cases with this proposal. Gwyn noted that since there are no universal formulas for handling NaN's and ∞'s, he questioned the utility of automatic computation of results for special cases as this may be a disservice to users who then don't (have to) think about these cases. But Thomas believes this disallows making results predictable for the specialist, who otherwise must be paranoid in writing code.

- Simplicity–the Thomas proposal simplifies the specification, implementation, and the programs for complex.

- Complex special cases are more, well, *complex*.

- Preclusion of support for special cases. Special cases are not precluded in the CRI proposal. Thomas is still skeptical of this.

- Other languages and imaginary types. Although there is no imaginary type in Fortran, there is a class-like package available in Ada with imaginary types (see X3J11.1/93-035, 93-036).

- IEEE and complex. IEEE doesn't specify complex, but it is derived from real arithmetic model–i.e., the Cartesian product model.

Finally, Thomas outlined what the alternatives would be if there was no imaginary type included with the complex proposal, and claimed that each of the alternatives would be at least as complicated as the imaginary type proposal.

- under the table implementations
- exclude infinities, NaN's and signed zeros from simple specification
- auxilliary specification
- imaginary unit I which has complex type and imaginary property
- propogation rules for imaginary and real properties–i.e., promotion rules by property rather than type
- ci + ci = ci, ci * ci = cr, ...
- conversion rules of ci, cr to complex yields ...
- adaptation for C++ problematic

2.40

• simpler for whom? Without imaginary type, complex is only useful for a very few persons who are really interested in managing all the special cases themselves.

Cheng presented the complex proposal components of C^H as described in X3J11.1/93-020 and X3J11/93-058. This proposal includes specification of:

- Declaration of Complex Variables
- A Complex Constructor
- I/O for Complex Numbers
- Semantics of operations on complex values, both as operators and functions
- Valid lvalues related to complex numbers

Cheng presented an overview of different models for representing complex. These models differ in their specification of infinities in the complex domain. Cheng's proposal is consistent with a complex model in which there is only one infinity. There was some discussion (Farance, Gwyn) on the regularity of infinities under these models.

Cheng noted the following properties of his proposal:

- Complex numbers follow mathematical conventions
- Program Complex Numbers in the Extended Finite Complex Plane
- Deliver a Consistent Correct Numerical Value or Complex NaN
- Distinguish -0.0 from 0.0 in Real Numbers, not in Complex Numbers
- The Principal Value Θ lies in the range of $-\pi < \Theta \le \pi$
- $F(x + i0) = F(x) + i0$, if x is Within the Valid Domain of $F(x)$
- When a complex is converted to real number, only its real part is used and the imaginary part will be discarded if the imaginary part is identically zero. If the imaginary part is not identically zero, the converted real number becomes NaN.
- Optional Arguments for Different Branches of Mulitple-Valued Complex Functions are used

7. FP/IEEE [X3J11.1/93-028, 93-029, 93-033, 93-058; X3J11/93-054 (WG14/N307), 93-055 (N308)]

Thomas presented a few last proposed tweeks to the FPCE part of the technical report.

The first addresses the comments received on the last ballot from Meyers, in which he noted that returning HUGE_VAL from strtof() and strtold() would be problematic. The proposal is to introduce new macros for the float and long double versions of HUGE_VAL, HUGE_VALF and HUGE_VALL. The changes are then made to the appropriate sections throughout the proposal.

Another tweek resulted from a problem noted by Clive Feather with Standard C name spaces and the new (unreserved) names introduced by <fp.h> and <fenv.h>. The concern is that, since the names are not reserved names, it would be possible to link with an incorrect library. The proposed fix is to add words in the rationale to suggest how an implementation can preserve the program name space by providing a mapping of the new names to names reserved for the C library:

241

#define acosh __acosh

There was some discussion on whether this was adequate to solve the problem. It was noted that this is the same solution suggested in the Normative Addenda for the multibyte names.

There was one other substantive change proposed, in response to a bug found by Yinsun Feng, correcting the wording of differences between roundtol() and rinttol() in F.6.7 to be with respect to default rounding direction.

There were some additional editorial changes to other parts of the document. There were no objections to any of the proposed changes.

MSP Shall we accept the proposed changes to the FPCE document? (Farance, Zeeb)
9 Yes. 0 No. 1 Not voting.

* Thomas will prepare draft #2 of the FPCE document with these changes.

Cheng presented an overview of the FP/IEEE features included in C^H as summarized in X3J11/93-058.

- Program Real Numbers in the Entire Real Domain with NaN, ±0.0, and ±Inf
- Deliver a Consistent Correct Numerical Value or Nan
- Polymorphic Functions
- Add Binary Integral Constants–e.g., 0b11110, 0B11110
- Add Binary Format Specifier, %b
- Add Double Constants–e.g., 3.4D, 3.4e9D, 3.4E9d
- Add Exclusive-or Operator, ^^
- Increment and Decrement Operation–allow multiple ++ or -- operations
- Bitwise Shifting Operation–reverse shift direction if shift amount is negative
- Functional Type Conversion Operation–int, float, complex conversions
- Relational Operations–detail results for exceptional values in relational expressions
- Single Delimiter Comment–uses /# to delimit comment to end of line

Gwyn suggested that Cheng should propose these changes to X3J11. MacDonald noted that the mechanisms for getting agenda time to consider such proposals requires that a champion attend the meetings to present the proposal.

Those interested in further discussions with Cheng on these issues may contact him at:

hhcheng@ucdavis.edu

8. Array Syntax [X3J11.1/93-034, 93-046, 93-050; X3J11/93-058 (WG14/N311)]

Stanberry presented the status of the DPCE subgroup. (Thanks to Keaton for providing portions of this section of the minutes from his notes on the discussion!)

- DPCE met in September in Toronto to work on editing and preparing X3J11.1/93-034 for distribution at this meeting. This document is considered a "draft" as it is still partially unspecified. However, it is the basis for future changes that the subgroup will consider. The subgroup has learned that it is not easy to develop complete and consistent standardese to capture the intent of their technical proposals!

• They will hold a technical/edit session in the evening during this meeting (see below).

• They will hold another technical/edit meeting, March 23-35, in Pleasanton, CA. They plan to complete a revised draft of the proposal for the second mailing of J11 before the June meeting.

• Discussion of the proposal takes place between meetings on email at:

```
dpce@farance.com
dpce-request@farance.com /* to subscribe*/
```

• The document will soon be available in Keaton's ftp archive as well.

Stanberry then gave an overview of the current proposal as detailed in 93-034:
• Concepts - the proposal introduces new concepts including/involving:
  • shapes – encapsulation of
    • rank
    • dimension
    • layout
    • context
  • types – objects, values, operands
  • operations
  • contextualization
  • intrinsics and builtin functions

• A shape is a descriptor of parallel data.
  • not itself parallel
  • used to define parallel types
  • is a new object type, which implies that it can be assigned, copied, passed as a parameter

```
shape [10][20]S;
shape [200]T;
shape U;

U = S;
U = T;
f(..., U, ...);
{ int:U localvar; /* block (function body) scope */
    ...
}
```

• Shapes when declared can be partially or fully unspecified, or fully specified:

```
shape S;        /* fully unspecified */
shape [10]T;    /* fully specified */
shape [][]U;    /* partially unspecified */
```

243

• Examples to illustrate how rank, dimension, and layout are specified:

```
        shape [100][200]S;
            /* rank 2, dimensions 100 and 200, default
               layout (implementation defined) */

        shape [1000 block(10)]T;
            /* rank 1, dimensions 1000, layout-blocks of
               10 consecutive elements per memory
               partition */

        shape [500 scale(10)]U;
            /* rank 1, dimensions 500, layout-evenly
               distributed across memory partitions in
               blocks of 10 elements */
```

MacDonald asked if shapes can be multidimensional, unlike arrays. Answer was that that wasn't quite the right question, but shaped objects (not shapes themselves) are multidimensional in the way that arrays aren't. Gwyn made an analogy between shapes (descriptors) and Algol-60 dope vectors.

It was noted that a goal of the proposal is not to cater to either SIMD or MIMD machines in particular.

There was some discussion of the syntax of shape declarations.

- A *parallel type* is a new aggregate data type derived from an element type and a shape.

- A *parallel object* is an object of parallel type: "A structured collection of one or more identically-sized objects where the structure is defined by a shape."
    - A collection of C objects
    - Object-like, but not strictly a C object

It was noted that parallel objects are not necessarily contiguous.

There was discussion (MacDonald, Frankel, Jervis) of memory layout: memory layout distinguishes an array of parallel objects from a parallel array of objects. Various people attempted to explain implications of this but much confusion remained. During the discussion, it was pointed out that a picture would help.

```
    shape [100]S;
    shape[10][20]T;

    int:S x;            /* parallel int type */
    float:T y;          /* parallel float type */
    double:S z;         /* parallel double type */

    struct {...}:T a; /* parallel struct type */
    union {...}:S b;  /* parallel union type */

    int:S *p;           /* pointer to parallel */
```

244

```
float:T q[100];    /* array of parallel floats */

typedef int A[100];
A:S w;             /* parallel array of ints */

int:S f();         /* function returning parallel int */
```

• Note: element type of a parallel type cannot be parallel, and parallel structs or unions cannot have parallel members or pointer members.

• Parallel operands are parallel-typed: either parallel objects (variables) or parallel values (expressions)

• Parallel operations
  • Promotion rules extended–a non-parallel value is promoted to a parallel value by replicating the non-parallel value
  • Conversion rules extended–both the integral promotions and the usual arithmetic conversions are modified for parallel operands

```
shape [10][20]S;
short:S a;
int:S b;
char c;
float:S d;

a + b;     /* a is promoted to parallel int */
b + c;     /* c is promoted to parallel int */
b + d;     /* b is converted to parallel float */
```

• Operators that are extended to produce parallel values
  • arithmetic: +, -, *, /, %
  • shift: <<, >>
  • unary: ++, --, &, !, *
  • bit: &, |, ~, ^
  • relational/equality: <, >, <=, >=, ==, !=
  • access: ., ->

```
shape [100]S;
int:S a,b;
struct {int a;}:S x;

a + b;     /* parallel int whose element values are
              the element-wise sum of a and b */
a << 2;    /* parallel int whose element values are
              the element-wise shift of a by 2 */
b++;       /* increments every element of b */
b & 0x101;/* parallel int whose element values are
              the logical product of elements of b
              and 0x101 */
a < b;     /* parallel int whose element values are
              the element-wise comparison of a and b
              */
```

245

```
x.a;        /* parallel int whose element values are
               selected from the a field of the
               parallel struct x */
```

- New operators for parallel operands
  - reduction operators–new use of "compound assignment operators" as unary operators, as well as new operators for min and max

```
shape [100]S;
int:S x;

+=x;        /* Sum reduction of elements of x */
&=x;        /* Logical product reduction of x */
<?=x;       /* min reduction */
>?=x;       /* max reduction */
```

  - parallel element accessing–"parallel indexing"

```
shape [100]S;
int:S x;

[10]x;      /* Accesses "10th" element of x */
```

    - reflects potential non-local cost (communication) to access element
    - future specification will detail indexing by parallel int, e.g.,

```
int:S a, b, c;
a = [b]c;
[b]a = c;
```

- Assignment operators

```
shape [100]S;
int:S x;
int sum;

x = 0;      /* Assigns every element of x to be 0 */

sum = 0;
sum += x; /* Adds every element of x to sum */
          /* Same as sum = sum + += x */
```

- Contextualization
  - The context of a shape is the set of active positions–i.e., which elements will participate in parallel operations
  - Default (initial) context is all positions are active
  - Context modification occurs in
    - where and everywhere statements
    - `&&`, `||`, `?:` operators
    - scope of context modification is the operation or statement block
    - context is restored at the end of the operation or statement block
  - Masks are parallel ints specifying context

```
shape [100]S;
int:S a;
float:S b;

b = a;      /* all positions active */
where (a >= 0) /* Sets new context for S */
   b = a;   /* executed for active positions */
else
   b = -a; /* executed for inactive positions */
/* context of S restored */
where (a > 10) /* Sets new context for S */
{ ...
  everywhere (S) a++; /* overrides (resets) context
                         of S */
}
/* context of S restored */
```

- Other features
  - elemental functions
    - operate elementally on elements
    - act as non-parallel functions on scalar arguments

```
void elemental f(int a);
{
   a++;
}
shape [100]S;
int:S x;
int y;

f(x);    /* As if parallel execution for each
            element of x */
f(y);    /* ordinary C function call */
```

  - Not yet specified in the document are the following features that have been proposed and accepted during technical discussions:
    - nodal functions
    - slicing
    - parallel pointer handles
    - intrinsic/builtin functions, including:

dimof()	rankof()
shapeof()	positionsof()
allocate_shape()	deallocate_shape()
pcoord()	

    - library extensions
      - parallel versions of math
      - parallel storage management

Parallel control flow proposals have been debated by the DPCE subgroup during technical sessions, without consensus being reached, so it was left out of the proposal.

Stanberry summarized the discussions and suggestions made at the DPCE evening session on Monday, 6 December:

- Farance proposed adopting APL-style terminology throughout the document. This proposal would require changes to the document in the following areas, which Farance will itemize in a formal proposal to be presented to the subgroup at its next meeting:
    - Definitions (shape, ALO, ...)
    - Separation of the concept of shape from layout and context
    - Revise
        - promotion rules
        - reduction ops
        - declaration syntax
- On contextualization, it was suggested by Jervis that the proposal should limit the scope of contextualization to function boundaries
- MacDonald and Jervis supported separation of layout/distribution from parallel operations; want parallel operations on non-distributed objects
- Limit keywords–make available through an appropriate header file
- Add discussion of errno, exceptions

At last, the time had come "to talk of many things," which are recorded here in the order of speakers, but notice that there are several parallel threads of discussion taking place:

Plum asked if Fortran 90 dope vectors describe parallel data. MacDonald and Farance indicated that it does. Plum suggested that the subgroup develop a design criteria list (functional decomposition) to evaluate both the DPCE and VLA proposals, considering their interaction with Fortran 90 and other proposals such as HPF.

* Farance will investigate other languages' features (Fortran 90 and HPF) that may have an impact on the DPCE and VLA proposals.

Jervis and MacDonald summarized Frankel's concern that Fortran 90 lacks the ability to adequately express communication cost.

MacDonald added that HPF recognizes that it can't provide a model that is optimal for all communication architectures.

Cheng noted that Fortran 90 is architecture independent, high level specification, and that this is not good for system programmers. He believes we need some middle ground for system programmers and beginners.

Gwyn stated that we must not lose sight of C clients (signal processing, toasters, etc.) who need access to architecture dependent features.

MacDonald added some cautions from Fortran 90 experience: (1) must be careful using operators and intrinsics–if any operands overlap, user must make own temps/copies; and (2) equivalence supports compound implementations.

Jervis reiterated problems of non-separability of layout and context from shape for his customers. The constraint of "same shape" as currently in the DPCE proposal is more than most data parallel programs are interested in. It imposes a burden on non-distributed memory architectures to "plan ahead" when it is irrelevant to their architectures.

248

Plum noted that the critical functional feature is the non-contiguity of an ALO. The question is how to specify distribution of objects. For instance, C++ dynamic arrays allow dynamic extensibility without moving the original object as a consequence of non-contiguity relaxation.

Gwyn reminded the committee that there are already many instances in the C standard (e.g., pointer sizes) in which we have adopted more neutral specifications to accommodate architectural variations.

Farance stated that his proposal will separate the functional aspects of layout and context from shape.

Frankel disagrees that DPCE is oriented toward distributed memory architectures, although he believes the distributed memory model is the most general and can work well on any architecture. Thinking Machines experience of customers writing programs without layout in the design resulted in very poor performance, and adding layout directives after the fact does not yield optimal performance.

Plum noted that the left/parallel indexing will be a major liaison problem.

Farance added that he believes that left/parallel indexing is not portable across architectures. Separation of layout from shape will still allow implementations to combine the concepts for specific architectures.

Frankel explained that left versus right indexing separates non-contiguous from contiguous address spaces. He noted that you can still have all indices on the right, however. The main point is to distinguish parallel objects from arrays since they aren't accessible in the same way as C arrays. He noted that if fat pointers are used for distributed objects, it would be possible to support ordinary array indexing through pointer operations.

Plum noted that parallel objects under DPCE need layout information for efficient operations. He clarified that resizing is not part of the proposal; this would require changing a shape as part of resizing.

Jervis reiterated the two fundamental differences in approach: (1) if the concepts of layout and context are separable from shape, you can maintain contiguous access operations, but (2) if they are combined, you give up non-contiguous access. The proposals haven't captured what we want: elegance of expressibility with C efficiency.

Plum suggested that C needs to grow beyond efficiency to handle new data concepts.

Gwyn suggested that we need to allow each programmer to design for own needs.

Thomas asked for clarification on what happens with subgroup proposals now that J11 and NCEG have merged. Jaeschke responded that it will require J11 approval to include a proposal in the technical report, and a 2/3 vote to change a proposal thereafter.

Frankel stated that Thinking Machines doesn't believe that any of the current C++ proposals or class libraries provide needed performance.

249

Plum said we shouldn't preclude class library implementations.

Farance led a review of the document, section by section, in the time remaining. There were numerous comments and suggestions from the committee that will be incorporated in the next revision of the document. These included:

- physical - should be defined in terms, §1.6
- clarify that shapes are multidimensional, unlike arrays
- alphabetize term definitions
- clarify "single thread" - programmer's model versus execution model
- clarify that you can't really express in C the same functionality as in DPCE, as suggested in the example in §2.1
- avoid COBOL-style keywords (using underscores)
- use header file to introduce new "keywords"
- define what is meant by "nodes" in §2.1.2
- clarify layout of physical - assumes gang scheduling?

* Meissner will determine what is the official dictionary for terms not defined in the C standard.

There were numerous editorial/standardese corrections suggested that will also be included in the next revision.

Farance asked for specific direction from the committee on proceeding with the DPCE proposal.

SV Should Farance prepare a proposal for separable layout and context concepts with respect to the DPCE document?
9 Yes. 3 No.

MacDonald presented the latest revision of the CRI proposal for iterators, X3J11.1/93-050.

- Matrix multiply example

```
void f(int m, int n, int l,
            double a[m][n],
            double b[m][l],
            double c[l][n])
{
    unord int I = m, J = n;
    iter int K = l;
    /* implied loops */
    a[I][J] += b[I][K] * c[K][J];
}
```

- unordered iterators imply no guaranteed order
- ordered iterators imply C (sequential) order

250

- Guard – atomic update or critical section, without ordering

```
double f1(double *a, int n)
{
     double sum = 0.0;
     guard(unord int I = n) sum += a[I];
     return sum;
}

void f(int m, int n, int l,
          double a[m][n],
          double b[m][l],
          double c[l][n])
{
     unord int I = m, J = n, int K = l;
     for (I, J, K)
          guard(K) /* sequentializes (unordered) */
               a[I][J] += b[I][K] * c[K][J];
}
```

- Key points
  - separates parallel operations from data distribution (layout)
  - guards allow parallel operations on linked lists and trees
  - minimal addition

- Changes
  - iterators are now specified with a storage class, rather than type-specifiers

Meissner expressed concern about the use of a storage class. MacDonald clarified that this is desirable to prevent access of iterators.

- Advantages were itemized

- Compatible with previous implementation by Analog Devices

Frankel commented that the expressability of iterators is very good–can use to denote scans and spreads. It is a very high level specification, but gives no way to estimate performance. It is left up to the compiler to make efficient, which has both advantages and disadvantages.

Meissner noted that the proposal assumes 0 as a lower bound on index range. MacDonald indicated that it could be extended for non-zero lower bound in the future.

Farance commented that the iterator proposal is not necessarily competing with other parallel extension proposals.

9. Variable Length Arrays [X3J11.1/93-038, 93-041, 93-043; X3J11/93-058 (WG14/N311), 93-059 (N312)]

Cheng presented his proposal for assumed-shape arrays, from X3J11.1/93-041 and X3J11/93-058:

- Rules and restrictions
  - the same syntax as for Fortran 90, using a ':' to indicate a variable extent. For example:

    ```
    int funct(complex a[:], int b[:][:]);
    ```

    It can only be used in function definitions or prototypes for specifying array parameters in the current implementation of C^H.
  - the rank of an assumed-shape array is equal to the number of colons.
  - array indices are coerced to adhere to array bounds–if index is less than 0, a 0 index is used; if index is greater than upper bound, the upper bound is used; warning messages are issued.
  - only arrays can be passed as assumed-shape arguments; pointers cannot.
  - polymorphic and intrinsic functions will be applied to assumed-shape arrays, depending on the data type.
- Should this also be a rule?
  - the same array can not be passed to the assumed shape arrays of a function more than once; otherwise, the behavior is undefined.

Gwyn and MacDonald agreed that this last proposed rule should be compatible with the wording and constraints used in Fortran.

There was discussion and questions on pointer operations and sizeof applied to assumed-shape arguments. Cheng pointed out that when pointer and sizeof operations are applied to an assumed-shape array, the results are obtained as if they were applied to the original array passed to the function.

MacDonald suggested that this was similar to the fat pointer proposal, but didn't allow access to the descriptor passed for the array. Also, there is no explicit dereference as required in the fat pointer proposal.

Meissner stated that the fat pointer proposal solves more general VLA needs whereas this VLA proposal only works for function parameters. For example, it doesn't handle dynamic struct applications sucah as a linked list.

Meissner then presented his version of the fat pointer proposal, X3J11/93-059:

- Two parts
  - VLA's–essentially the same wording as in MacDonald's paper for describing scoping details
  - Fat Pointers

    ```
    a[*]     /* Declares fat pointer */
    a[exp]   /* Declares array */
    ```

  - Contain two parts ("dope vector")
    a) pointer to base array
    b) bounds for each array bound until you run out of dimensions

25-2

Kwan asked about the representation of the dope vector.

Plum approved of the improvement in syntax, simplicity of use for beginners from previous versions.

MacDonald noted that this version introduces an implicit dereference; he was smiling when he mentioned this!

- Conversions between pointers and fat pointers
  - ptr <- fat ptr : just converts base pointer
  - fat ptr <- ptr : not allowed
  - fat ptr <- array : base pointer <- array, bound <- array bounds
- Constants/sizeof
  - Neither VLA or fat ptr are constant sized
  - sizeof(fat ptr) is the product of all array bounds and the sizeof(base type)
- Example

```
void foo(int[*]);
void bar(int n);
{ int VLA[n];
  foo(VLA);
}
void foo(int fp[*])
{ int i;
  int max = (sizeof(fp))/(sizeof(int));
  for (i=0; i<max; i++)
    fp[i] = 0;
}
```

Meissner also noted there was a synergy with "shapes" that needs to be explored to see how they interact.

Plum suggested that fp[*] be used to denote the whole array in sizeof expressions–i.e., sizeof(fp[*]) is the size of the array, not of the fat pointer. It would be nice to use the same notation for all arrays. This is a style issue, but it is easier to teach!

MacDonald asked if &(fat pointer) converts to pointer to array. (He was still smiling.) Meissner said that would be ok, and that would be useful for passing the array to memcpy(), etc.

Gwyn suggested that explicit indirection recipe might be better for long term.

MacDonald suggested using ? or : instead of *. Meissner wondered if ? might be better for C++ compatability.

Plum stated that we still don't have any one VLA proposal that answers all the problems raised in other VLA proposals. He suggests that we need a thoughtful synthesis of all proposals.

SV Is implicit dereferencing preferred to explicit indirection?
10 Yes, 3 No.

253

* Meissner will expand and revise the fat pointer proposal for the next meeting.

MacDonald presented the CRI proposals in X3J11.1/93-038 and 93-043. The proposals are the same except 93-043 maintains strict lexical ordering: the relaxed lexical ordering prototype section has been moved to an appendix. The changes otherwise are:

- 93-038 is diff-marked against 93-007

- Variably Qualified vs. Variably Modified
  All occurrences of the term *variably qualified* were changed to *variably modified* throughout the proposal to avoid confusion with other uses of the term *qualified.*

- 6.3.6 (3.3.6) Additive Operators
  Rationale was added to this section to clarify that pointer arithmetic involving "pointers to VLAs" is well defined.

```
{
  int n=4, m=3;
  int a[n][m];
  int (*p)[m] = a;    /* p==&a[0]      */
  p+=1;               /* p==&a[1]      */
  (*p)[2]=99;         /* a[1][2]==99 */
  n=p-a;              /* n==1          */
}
```

- 6.5.2 (3.3.2) Type Specifiers
  - Add identifiers with function prototype scope as accepted VLAs
  - Add block scope static pointers to VLAs; the only difference is in the storage duration of p.

```
int B[100];

void f(int n) {
  static int (*p)[n]=&B;     /* OK */
}
```

Plum asked whether cast/assignment of fixed size array to VLA can be done without a dope vector. MacDonald noted "proof by implementation" since CRI has done it–works like Fortran77 assumed shape arrays, basically.

Cheng asked for clarification of what p[0] is in last example above. It is a pointer to an array element.

- 6.5.4.2 (3.5.4.2) Array Declarators
  - Changed to "The [and] *may* delimit an expression or *" since [] is accepted.
  - Clarify that:

```
int A[(int)(3.2+5.4)]
```

254

declares A to be a VLA, since the size is not an integral constant expression. (Note that this is an error if at file scope.) Since this results in undefined behavior, it could be accepted by a compiler, however.

Gwyn asked if this VLA proposal becomes part of the language, will it matter whether this is treated as a constant expression?

255

- 6.6.2 (3.6.2) Compound Statement
  - Clarify that VLA expressions are evaluated when a block is entered (just like auto initializers)

```
{
    int a[n][n];
}
```

Jervis asked what if "int i = n++;" occurred before the declaration above. The proposal has the right words to enforce "point of declaration" semantics rather than beginning of block, honoring sequence points.

  - Added the words "and variably qualified declarators" to this section.

- Can create own dope vector if needed.

```
struct tag {
   int n;
   void *p;
} x = { 0 };

{
  x.p = malloc(N);
  x.n = N;
}

{
  int m = x.n / sizeof(int);
  int (*q)[m] = x.p;
}
```

Frankel suggested that words be added in Appendix B that make it not sound like an optional implementation.

SV How many favor the strict lexical ordering in the current proposal? 14
How many favor the relaxed lexical ordering? 1

There was discussion of the C++ compatibility of the VLA proposals. Plum stated that we need to propose on both sides how liaison should proceed between J11 and J16. We should propose to J16 that they set up a committee/subgroup to handle extensions not currently in C. Then we need to follow, pay attention, and provide feedback to them on the utility of these extensions in future C++. It would help them to get their standard out if these extensions are not to be part of that standard, however.

Plum believes that only the dope vector/descriptor-based approach will be C++ compatible. He expects hidden object versions won't map into C++ class objects. J16 is also considering some dynamic array proposals that we should be aware of.

MacDonald stated that we also have to consider Fortran compatibility, not just C++.

SV How many prefer the CRI VLA proposal? 5
How many prefer the fat pointer VLA proposal? 10

There was further discussion of how to proceed with the two VLA proposals. MacDonald believes that the CRI proposal is finished; what do they do with it now? Gwyn suggested that it be forwarded to J11 and let them decide to accept it or table it or whatever. Jervis suggested that the technical report could contain two proposals. Plum suggested that both proposals could be published, and that one or both might become part of a future standard. Jaeschke suggested that a comparison of the proposals would be helpful. Thomas prefers that only one VLA proposal be published.

MSP Shall we forward the CRI VLA proposal (X3J11.1/93-043) to J11 for inclusion in the technical report? (Frankel, MacDonald)
9 Yes*, 0 No, 1 Not voting
*1 "Yes, with comment" (Thomas)

SV In favor of Meissner refining the fat pointer proposal?
11 Yes, 1 No, 5 Don't know/don't care

MSP Move to reconsider CRI VLA vote. (Frankel, Kwan)
8 Yes, 0 No, 1 Abstain, 1 Not voting

MSP Shall we forward the CRI VLA proposal (X3J11.1/93-043) to J11 for inclusion in the technical report, and coincidentally make the proposal available for broader distribution for public comment? (Frankel, MacDonald)
9 Yes*, 0 No, 1 Not voting
*2 "Yes, with comment" (Keaton, Thomas)
(Keaton) "This is with the understanding that it does not exclude the other VLA proposal."

* Thomas will provide his comments for his votes on the CRI VLA proposal.

* Meissner will revise the fat pointer proposal for the next meeting.

* MacDonald will forward the CRI VLA proposal to J11.

10. Extended Integers [X3J11/93-056 (WG14/N309)]

Kwan presented an overview of the proposal for an <inttypes.h> as given in 93-056:

- Purpose
  - not a general solution
  - not tied to 64-bit (or an n-bit) systems
  - provide a way to write portable code by
    - defining new integer types
    - providing consistent properties and behavior
    - easily implemented
- Definitions for architectures supporting int sizes that are powers of 2
  - Integers of exactly n bits

```
typedef ? int8_t
typedef ? int16_t
typedef ? int32_t
typedef ? uint8_t
```

257

```
typedef ? uint16_t
typedef ? uint32_t
typedef ? int64_t
typedef ? uint64_t
```

• Largest integer supported

```
typedef ? intmax_t
typedef ? uintmax_t
```

• Most efficient integer type

```
typedef ? intfast_t
```

Gwyn asked isn't "int" the most efficient integer type. This was true under K&R, but not necessarily in standard C.

Keaton asked if there is any minimum bit count that `intfast_t` has to be? 0..intmax was suggested.

Jaeschke and Frankel suggested that you also need to define a `uintfast_t`.

MacDonald asked if implementers are free not to define any type not supported. The answer was yes. This led to discussion of how to be portable then. Requires use of macros to determine which sizes are supported.

• Pointers
  • Integer datatypes that are large enough to hold a pointer

```
typedef ? intptr_t
typedef ? uintptr_t
```

Discussion: should these be void *? Note that there is no requirement that any int type be the size of a pointer in standard C.

• NOTE:
  • Not all pointers are the same size in some systems
  • Should ptrdiff_t be the same as intptr_t?

Discussion: no, ptrdiff_t and intptr_t should not be the same.

• Limits
  • Fixed length integer type

```
#define INT8_MIN (-128)
#define INT16_MIN (-32768)
#define INT32_MIN (-2147483647 - 1)
#define INT8_MAX (127)
#define INT16_MAX (32767)
#define INT32_MAX (2147483647)
#define UINT8_MAX (255)
#define UINT16_MAX (65535)
#define UINT32_MAX (4294967295)
```

258

There was discussion of how to express limits in form (*n* - 1) when it may not be possible to represent *n*.

Meissner questions the usefulness of min/max macros. Plum indicated that it was a convenience to the programmers, not a standard requirement.

Jervis indicated that there should be a min/max for every type, including `intfast_t` and `intptr_t`.

There was discussion of whether to map to these types if not supported–e.g., 36 bit machines. No, that's what "at-least" versions are for. Need to use "feature" macros to determine what types are supported.

- Constants

```
#define __CONCAT__(A,B) A##B
#define INT16_C(c) (int16_t c)
#define UINT16_C(c) ((uint16_t)__CONCAT__(c,u))

#define INTMAX_C(c) ((int64_t)__CONCAT__(c,ll))
```
  - implementation defined

Jervis questioned the need for these macros as they don't seem to buy an advantage over a cast. MacDonald and Meissner argued that they are needed to generate ll constants.

- Formatted I/O - 3 ways to do
  - extend the width specifier
    - use "wnd" where n is the width of object in bits
```
      printf("int 16 is %w16d\n:,s16);
```
  - Use * as the width and sizeof operator
```
    printf("int fast 16 is %w*d\n",
              sizeof(intfast16_t) * bits_per_byte,
              myint);
```
    - can be used with datatypes of "at least" n bits

  - Use macros - requires no extensions

Meissner indicated a fourth way is to cast all to long, which users must do on their own.

- Conversion functions
```
extern int8_t strtoi8();
extern int16_t strtoi16();
extern int32_t strtoi32();
extern int64_t strtoi64();
extern uint8_t strtou8();
extern uint16_t strtou16();
extern uint32_t strtou32();
extern uint64_t strtou64();
```

259

```
extern intmax_t strtoimax();
extern uintmax_t strtoumax();
```

• only the last two are really necessary

• Integers of "at least" n bits
  • Most efficient signed integral types of at least n bits:
```
typedef ? intfast8_t
typedef ? intfast16_t
typedef ? intfast32_t
typedef ? intfast64_t
```
  • Most efficient unsigned integeral types of at least n bits:
```
typedef ? uintfast8_t
typedef ? uintfast16_t
typedef ? uintfast32_t
typedef ? uintfast64_t
```
  • Do we need these? These were difficult to specify and raise issues of space versus execution efficiency.

• level of support
  • two levels of support possible
    • with extensions to library and/or compiler
    • without extensions

Kwan would like feedback to update and complete this proposal for the San Jose meeting.

Thomas asked for clarification that all definitions are available. Yes, they are.

Tydeman noted that a switch is needed to indicate what model was assumed for code. Kwan agreed, and added you also need a switch to indicate what the default is.

MacDonald asked if this part of the technical report would be for information only or expected to become part of a future standard. Is this solving a general problem or just for a set of specific users?

Meissner suggested that the Farance approach (see below) is more general, but the <inttypes.h> approach is more immediate. Plum suggested that we not delay the <inttypes.h> proposal for including the Farance notations, which still need to be worked out with respect to their interaction with C++, etc.

Jervis commented that the <inttypes.h> proposal is ok for the technical report, but would pose problems for some architectures if it became part of the standard. Frankel noted that we are voting for the FPCE proposal even though not all architectures can support it.

SV Should Kwan continue to work on the <inttypes.h> proposal?
14 Yes, 1 No, 2 Don't Know/Don't Care

* Kwan will revise the <inttypes.h> proposal for the next meeting.

260

Farance presented an alternative approach to extended integer range specification:

- Desired features
  - Exact specification of precision
  - Fastest type for precision
  - Promotion of types
  - Byte-ordering/alignment
  - Bit-ordering/alignment

- Exact/Fastest specification
  - Exact type needed for external structure (e.g., shared databases). Performance isn't as important as exactness.
  - Fastest type gives performance
  - Smallest may be bigger than exact

```
exact =>       int:N x;       /* N specifies number of bits */
fastest =>     long:N x;
smallest =>    short:N x;
```

- Promotion of types
  - short, int, long can <u>all</u> be mapped into some int:N. For example:

```
short    int:16
int      int:32
long     int:32
```

  - This naturally merges basic C types with specific ranges
  - User defined types are ordered with C types

- Byte/Bit Ordering/Alignment
  - These features are necessary to write portable code for data structures whose bit/byte ordering/alignment is specified (e.g., Internet packets).

- Bit/Byte Ordering
  - Type qualifiers:

```
big_byte    => Big Endian
little_byte => Little Endian

big_bit     => MSB is bit 0
little_bit  => LSB is bit 0
```

- Bit/Byte Alignment
  - Type qualifiers:

```
byte_align:N      => align to N bytes
bit_align:N       => bit fields align to N bits
```

  - Important: Performance isn't as critical as portability

2(/

• Example

```
typedef big_byte long:32 int32;
typedef big_byte long:16 int 16;

struct
{
  int32 byte_align:4 a;
  int16 byte_align:4 b;
  int16 byte_align:4 c;
};

------------------------------
|    a     | b |    | c |    |
------------------------------
                ^         ^
                |         |
                -----------
                   holes
```

• Prior art
    • In BSD Unix (and any system that supports sockets), there are functions to support network to host ordering conversions. Alignment isn't supported, but XDR is a first cut at support there.

    • Problems with this:
        • too many functions
        • not enough data types
        • no alignment support
        • preference for VAX and 68000
        only 8, 16, 32 data types (no 64)

• Work to be done
    • printf/scanf support
    • other library support?
    • more convenient syntax
    • log2 macro to map range to number of bits
    • min/max values
    • pointers to types - which are valid?

Meissner stated that this sounds a little like the ADA specification.

Gwyn prefers this general size specification to an "exact" size specification.

Thomas noted that this has applicability to other than just int types.

Frankel noted that DPCE also has seen a bit array proposal from Maya Gokhale with similar syntactic notations.

Kwan will try to coordinate with Farance, but not with respect to alignment and endianness parts of his proposal.

Farance will keep his proposal separate until the inter-language details are solved.

11. Administration

11.1 Summary of decisions reached

Stanberry reviewed the votes taken during the meeting. Note: most votes were postponed until this section of the agenda time, but they are recorded with each subgroup discussion in the sections above.

11.2 Action Items Committed To

Stanberry reviewed the action items from the meeting.

11.3 Future Meeting Schedule

Jaeschke reminded us that we start meeting jointly with J11 as of the next meeting. Also, X3J11/WG14 has agreed to begin work on a revised standard, and this will mean meeting jointly and three times per year, instead of twice a year, beginning after the meetings that have already been scheduled (i.e., in 1995).

The meetings scheduled as of this time are:

6/6-10/94	San Jose, CA (J11 only)	Perennial/HP
7/27-29/94	Tokyo, Japan	Japanese Standards
12/5-9/94	Richardson, TX	Convex
6/19-23/95	Copenhagen, Denmark	Danish Standards
10/16-20/95	Boston, MA	Thinking Machines
2/5-9/96	Orange County, CA	Unisys

There will also be a DPCE meeting:

3/23-25/94	Pleasanton, CA	LLNL

11.4 Mailings and Submission Procedures

Now that J11 and NCEG have merged, all document numbers must be obtained from Plauger:

P. J. Plauger
398 Main Street
Concord, MA 01742

pjp@plauger.com
phone: 508-369-8489
fax: 508-371-9014

Items for mailings should have the document number on the cover of the document, and then should be sent to Plauger. He prefers hard copy, then fax, then email.

263

Deadlines for mailings are:

Post-Kona	1/7/94
Pre-San Jose	4/29/94
Pre-Tokyo	6/3/94
Post-San Jose	7/8/94
Post-Tokyo	8/26/94

14.5 Next Meeting Agenda

The needs of the subgroups for the next meeting were discussed. Jaeschke will schedule agenda time for each subgroup, and agenda time for a report on Fortran 90 as it relates to DPCE and VLA proposals.

14.6 Other Business

Jaeschke thanked CRI for providing the NCEG mailings.

There was discussion of what to do with the nceg email reflector now that J11 and NCEG have merged. It was proposed that the J11 members should be removed from the nceg mailing list, and that the j11 reflector address should be added. Note: we won't send to the WG14 reflector, just to WG14 members who have subscribed to nceg reflector. MacDonald and Meissner will coordinate this change.

Jaeschke thanked Plum Hall for hosting this meeting.

Jaeschke thanked MacDonald for serving as vice chair of NCEG.

We all thanked Jaeschke for serving as chair of NCEG.

11.7 Adjournment

MSP Move we adjourn. (Stanberry, Keaton)
Lots Yes.

The meeting was adjourned at 5:42 PM, 7 December. Sunset was at 5:45!

264

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A

X3J11.1 (NCEG) Meeting Attendance/Voting Register

Meeting: X3J11.1 #6 **Date:** 12/6/93

18 attending, 10 voting

Please indicate your presence for each day of the meeting attended by checking the day number. For example, if you attended days one and three only, check only those columns. Also indicate whether you have voting status (V) at this meeting or whether you are an alternate (A)

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A
	Cruse, James	355 Western Drive, Apt. F Santa Cruz, CA 95060							
	Figueroa, Samuel	106 Mercer Street Jersey City, NY 07302			✓	✓			N
Sun Microsystems	Jervis, Bob	17645 Via Sereno Monte Sereno, CA 95030	robert.jervis@sun.com ~~uunet!bjervis!bj~~	408-354-2943	✓	✓			N
	Redelmeier, Hugh	29 Domino Avenue Toronto Ont M4N 2W6 CANADA							
	Schauble, Paul	5316 West Port au Prince Glendale, AZ 85306							
	Tatalias, Kosmo	2516 E. Meredith Dr. Vienna, VA 22181							
	Valerio, Jim	424 NW Skyline Boulevard Portland, OR 97229							
Advanced Computer Research	Bertin, Christian	1, Boulevard Vivier-Merle 69443 Lyon Cedex 03 France							
Analog Devices, Inc.	Hoffman, Marc	Analog Devices, Inc. 1 Technology Way Norwood, MA 02062	marc.hoffman@analog.com						
	Zatsman, Alex								
Argonne Nat'l Laboratory	Cody, W.J.	MCS/221, 9700 South Cass Avenue Argonne, IL 60439							

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A
AT&T Bell Laboratories	Prosser, David	190 River Road, Rm. F-326 Summit, NJ 07901	dfp@usl.com	908-522-6227					
	Ritchie, Dennis	600 Mountain Avenue,Rm 2C-517 Murray Hill, NJ 07974							
Australian National Univ.	Stewart, David	Program in Advanced Computation School of Mathematical Sciences GPO Box 4, Canberra, ACT 2601 Australia							
Cambridge University	MacLaren, Nick	Computing Laboratory New Museums Site Pembroke Street Cambridge CB2 3QG ENGLAND							
Control Data Corp.	Hashemi, Azar	P.O. Box 70010 Sunnyvale, CA 94086	azar@sul.cdc.com	408-496-4277					
Convex Computer Corp.	Torkelson, Bill	P.O. Box 833851 Richardson, TX 75083-3851	tork@convex.com	214-497-4328					
Cornell University	Sahr, John								
Cray Research, Inc.	Becker, David		becker@cray.com	612-683-5885					
	Knaak, David		knaak@cray.com	612-683-5611					
	Homer, Bill	655F Lone Oak Dr.	homer@cray.com	612-683-5606					
	Tom MacDonald	Eagan, Mn. 55121	tam@cray.com	612-683-5818	✓	✓			✓
CYRIX Corp.	Dunlap, Fred	2703 N. Central Expressway Richardson, TX 75080	fred@cyrix.texsun.sun.com	214-234-8387					
DEC Professional	Jaeschke, Rex	2051 Swans Neck Way Reston, CA 22091	rex@aussie.com	703-860-0091	✓	✓			✓
Depart. of Computer Science	Keppel, David	University of Washington FR-35 Seattle, WA 98195							
Digital Equipment Corp.	Meyers, Randy	VAX C Development 110 Spitbrook Road ZKO2-3/N30 Nashua, NH 03062-2698	rmeyers@tle.enet.dec.com	603-881-2743					
	alternate	Zeeb, Jeffrey	zeeb@tle.enet.dec.com	603-881-2070	✓	✓			✓

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A
Farance, Inc.	Farance, Frank	President 555 Main Street New York, NY 10044-0150	frank@farance.com	212-486-4700 fax: 212-759-1605	✓	✓			voting
Free Software Foundation	Stallman, Richard	545 Technology Square,Room 703 Cambridge, MA 02139							
HaL Computer	Boney, Joel	1315 Dell Ave. Campbell, CA 95008	boney@hal.com	408-379-7000					
Hewlett-Packard Co.	Kwan, John	19447 Pruneridge Avenue Cupertino, CA 95014	jkwan@~~hpda~~ cup.hp.com		✓	✓			✓
IBM	Tydeman, Fred	3711 Del Robles Road Austin, TX 78727-1814	tydeman@ibmpa.awdpa.ibm.com	512-838-3322	✓	✓			✓
IBM Canada Ltd.	Molenda, Pawel	844 Don Mills Road, Station 22, Dept 582 North York, Ont M3C 1V7 CANADA	molenda@torolabg.iinus1.ibm.com	416-448-4279					
IBM Toronto Lab	O'Farrell, Bill	IBM Canada Ltd. B2/894 895 Don Mills Road North York, Ontario M3C1WC CANADA	billo@torolab2.unet.ibm.com	416-448-2732					
IMSL	Smith, Phil	2500 ParkWest TowerOne 2500 CityWest Blvd. Houston, TX 77042-3020							
INMOS Ltd.	O'Neill, Conor	1000 Aztec West Almondsbury, Bristol BS12 4SQ England							
Intel Corp.	Lai, Konrad	HF3-45 5200 NE Elam Young Parkway Hillsboro, OR 97124-6497							
Keaton Consulting	Keaton, David	1630 30th Street, #311 ~~PO Box 1955~~ Boulder, CO 80301 ~~Woodland Park, CO 80866-1955~~	dmk@dmk.com	303-589-9DMK ~~719-687-3577~~	✓	✓			✓
Kendall Sq. Research Inc.	Peters, Tim	170 Tracer Lane Waltham, MA 02154							

please note address correction

267

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A
Lawrence Livermore Labs	Stanberry, Linda	P.O. Box 808, L-300 Livermore, CA 94550	linda@ocfmail.ocf.llnl.gov	510-422-9006	✓	✓			Y
Lucid, Inc.	Schwarz, Jerry	707 Laurel St. Menlo Park, CA 94025							
Maspar Computer Corp.	Alpern, David	749 N. Mary Ave. Sunnyv ale, CA 94086	alpern@maspar.com	408-736-3300					
Microsoft	Papaziannakopoulos, Georgios		georgiop@microsoft.com						
NAG, Ltd.	Datardina, Shah	Wilkinson House Jordan Hill Road Oxford U.K. 0X2 8DR							
NCAR	Adams, Jeanne	PO Box 3000 Boulder, CO 80307							
NEC Corp.	Noda, Mokoto	Daito Tamachi Building14-22, Shibaura 4-Chome Minato-ku, Tokyo 108 JAPAN							
NC State University	Vouk, Mladen	Dept. of Computer Science Box 8206 Raleigh,NC 27695-8206							
Open Software Foundation	Meissner, Michael	11 Cambridge Center Cambridge, MA 02142	meissner@osf.org		✓	✓			M
PACT	Kurver, Rob	President Foulkeslaan 87 2625 RB Delft THE NETHERLANDS							
Plum Hall	Plum, Dr. Thomas	Chairman	plum@plumhall.com	609-927-3770					
Polaroid Corporation	Vetterling, William	Principal Scientist 21 Osborn Street Cambridge, MA 02139							
SAS Institute, Inc.	Bradley, Oliver	SAS Circle, Box 8000 Cary, NC 27512-8000							

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A
Seque Software	Levine, John	PO Box 349 Cambridge, MA 02238							
Sun Pro, A Business Of: Sun Microsystems, Inc.	Hough, David	P.O. Box 20370 San Jose, CA 95160	dgh@validgh.com	415-336-7702					
Supercomputer Systems, Inc.	Rasbold, Chuck	2021 Las Positas Court, Suite 101 Livermore, CA 94550							
Taligent	Thomas, Jim	~~10725~~ 201 North DeAnza Blvd Cupertino, CA 95014-2~~000~~333	jim_thomas@taligent.com	408-777-5466 ~~[illegible]~~	✓	✓			✓
TauMetric Corporation	Ball, Mike	8765 Fletcher Parkway, Suite 301 La Mesa, CA 91942							
Texas Instruments	Simar, Ray	PO Box 1443, MS 701 Houston, TX 77251-1443							
	Tatge, Reid	P.O. Box 1443, MS 6431 Houston, TX 77251-1443							
Thinking Machines Corp.	Frankel, James	245 First Street Cambridge, MA 02142-1264	frankel@think.com	617-234-2754	✓	✓			✓
	Sabot, Gary	245 First Street Cambridge, MA 02142-1264	gary@think.com	617-234-2836					
Universitat Karlsruhe	Kulisch, Prof. Ulrich	Institut for Angewandte Math. Kaiserstrasse 12,Postfach 6980 D-7500 Karlsruhe 1 Germany							
Univ. of Amsterdam	Dekker, Dr. Theodorus	Dept. of Computer Systems Kruislaan 409 Amsterdam NL 1098 SJ The Netherlands							
Univ. of California	Kahan, Prof. W.	EE and CS Dept.-Evans Hall Berkeley, CA 94720							
Univ. of New Hampshire	Hatcher, Phil	Dept. of Computer Science Kingsbury Hall Durham, NH 03824	pjh@cs.unh.edu	603-862-2678					

2-19

COMPANY	EMPLOYEE	ADDRESS	E-MAIL	PHONE	1	2	3	4	V/A	
Watcom Systems, Inc.	Crigger, Fred	Director of R&D 415 Phillip Street Waterloo, Ont N2L 3X2 CANADA								
Plum Hall	Plum, Thomas	Po Box 44610 Kamuela HI 96743	plum@ plumhall.com	808- 882-1255	✓	✓	l	|	A	N
Univ. of California	Harry H. Cheng	Dept. of Mechanical and Aeronautical Engineering Univ. of California Davis, CA 95616	hhcheng@ucdavis.edu	916-752-5020	✓ 1	✓ 2	3	4	V/A	N
United States Army	Gwyn, Douglas A.	801-L Cashew Ct. Bel Air, MD 21014	gwyn@arl.army.mil	(410)838- 2765	✓	✓			[illegible]	N
Unisys Corp.	Jonathan Ziebell	19 Morgan St Irvine, CA	Ziebellz Hoiz@ecsse.tredydev.unisys.com		✓	✓				N
Amdahl Corp	Neal Weidenhofer		nw@amdahl.com	(408)737-5007	✓	✓				A

228