HPFF Meeting Notes for the June 8-10, 1992 Meeting

High Performance Fortran Forum

CRPC-TR93313
May 1993

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

HPFF Meeting Notes

June 8-10,1992
Dallas, TX - Bristol Suites Hotel
Notes taken by Chuck Koelbel, with additional material by Henk Sips

Executive Summary

This was the third meeting of the High Performance Fortran Forum. Most of the meeting was devoted to discussion of the Fortran 90 subset, data distribution model, and storage and sequence association proposals. The key decisions in these areas were:

- A partial Fortran 90 subset was approved consisting of the Fortran 77 standard (except possibly for storage and sequence association in some situations),the MIL-STD-1753 additions, the Fortran 90 array sublanguage, INTERFACE blocks, the INTENT attribute for dummy arguments, and certain syntax improvements. Some Fortran 90 features that were not included, but which might be added at a later time, included the object-oriented declarations syntax, KIND parameters, Fortran 90 pointers, and modules. Future meetings may add features to the HPF subset, but will not remove them.
- A two-level data distribution model was accepted for use in High Performance Fortran. One level maps array elements to an abstract TEMPLATE, while the other partitions the TEMPLATE among coarse-grain virtual processors. The virtual processors are mapped to physical processors in a machine-dependent manner. Certain data distribution features were not included in the proposal that was accepted; these included dynamic redistribution, subroutine interface issues, and the handling of global data such as COMMON arrays. This vote was a first reading of the proposal; therefore, it may be amended at the next meeting. (Issues not included in the proposal must still survive two readings.)
- A proposal to deal with distributing COMMON blocks was advanced against considerable resistance. Briefly, the proposal said that if any variable in a COMMON block explicitly used storage association, then the entire COMMON was "contaminated." Other proposals were advanced to liberalize this treatment. A straw poll showed support for liberalization of the rules, and work on an amended proposal is continuing.

In addition, the FORALL and intrinsics subgroups gave short presentations. Some results of these were

- The FORALL group was encouraged to develop proposals in three areas: a new statement (the FORALL) with parallel semantics, directives

(currently called INDEPENDENT) to assert that sections of code could be executed in parallel, and local subroutines which would allow programming individual nodes in a parallel machine. Their proposal will have its first reading at the next meeting.

- The intrinsics group also presented a preliminary proposal, including intrinsics for extending some existing intrinsic capabilities, various combining intrinsics (including prefix computations), new elemental operations for bit manipulation, sorting, and system inquiry functions. The major point of controversy was whether these should be required as intrinsic functions or whether library implementations would suffice. A formal proposal will have its first reading at the next meeting.
- No action was reported from the parallel I/O subgroup. In view of the lack of positive proposals, the group's convener suggested they might disband. Constructive proposals are urgently sought.

Dates and agendas for future HPFF meetings are as follows:

July 23-24	Storage Association, Dynamic Data Redistribution
September 9-11	FORALL (and Similar) Constructs, Intrinsic Functions Remaining Distribution Issues
October 21-23	Official Fortran 90 Subset, Miscellaneous Features
December 2-4	Approval of Final Draft

Some notes about the July meeting that may be particularly interesting to those not on the committee: As it is being held immediately after the International Conference on Supercomputing, there will be an opportunity for significant public participation. This meeting will be open to the public, subject only to the condition that any questions or comments must be made through the HPFF committee members themselves (i.e. not from the floor of the audience). There will also be a Birds of a Feather session at the conference itself.

HPFF Documents

The following documents related to HPFF are available by anonymous FTP from titan.cs.rice.edu in the public/HPFF directory. Files added or changed since the April meeting are marked with a plus sign (+).

announce.* HPFF Announcement and Call For Participation

+ archives Directory containing the archives of the HPFF mailing groups.

- + hpff.* All messages sent to hpff@rice.edu
- + hpff-core.* All messages sent to hpff-core@rice.edu
- + hpff-distribute.* All messages sent to distribution subgroup
- + hpff-f90.* All messages sent to Fortran 90 subgroup
- + hpff-forall.* All messages sent to FORALL subgroup
- + hpff-intrinsics.* All messages sent to intrinsics subgroup
- + hpff-io.* All messages sent to I/O subgroup

+ database The HPFF mailing list

handouts Directory containing handouts from the HPFF meetings

apr Directory containing April meeting handouts

ALL Concatenation of the other files in this subdirectory

Distribute Proposal for data distribution model
FORALL Proposals and discussion concerning FORALL
Fortran90 Storage association proposal
Pointer Discussion of Fortran 90 pointers
Subroutine Discussion by the Subroutine interfaces subgroup
+ june Directory containing June meeting handouts
+ ALL Concatenation of the other files in this directory
+ Association Storage and sequence-association proposal
+ COMMON-Kennedy Distributing COMMON proposal (Ken Kennedy)
+ COMMON-Meltzer Distributing COMMON proposal (Andy Meltzer)
+ Distribute.tex Data distribution proposal
+ FORALL-Loveman.tex FORALL construct proposal (David Loveman)
+ FORALL-Wu.tex FORALL construct proposal (Min-You Wu)
+ Intrinsics Intrinsics proposal
+ Local.tex Local subroutines proposal
+ Subset Fortran 90 subset proposal
minutes Directory containing minutes of past HPFF meetings
jan-minutes Summary of the first HPFF meeting
jan-proposals Point-by-point comparison of language proposals at first HPFF meeting
mar-minutes Summary of HPFF meeting, March 9-10
apr-minutes Summary of HPFF meeting, April 23-24
+ europe-minutes Summary of European Workshop on High Level Programming Models for Parallel Architectures
+ june-minutes Summary of HPFF meeting, June 9-10
papers Directory containing papers related to HPFF, primarily from presentations at the first meeting.
convex.* Slides for Convex presentation
fd.* Fortran D language specification
fd-over.* Fortran D project overview
fd-sc91.* Fortran D compilation paper (presented at Supercomputing '91)
hpf.ps DEC HPF language specification
mpp.ps Cray MPP Fortran language specification
tmc Position paper by Thinking Machines on HPFF
+ vf.* Vienna Fortran language specification
+ vf-over.* Vienna Fortran language overview
welcome "Welcome to HPFF" message, including instructions for using anonymous FTP and joining mail lists.

See the README file in the main directory for information on file extensions and compressed files.

Public comment on any of the proposals is welcome. Please address your remarks to the appropriate email list (see the "welcome" file for a list of these groups).

Detailed Meeting Notes

Attendees

Alan Adamson (IBM Canada), Robert Babb (Oregon Graduate Institute), Jim Bailey (Thinking Machines), Ralph Brickner (Los Alamos National Lab), Alok Choudhary (Syracuse University), Marina Chen (Yale University), James Cownie (Meiko), Rich Fuhler (Lahey), Uwe Geuder (Universitat Stuttgart), Peter Highnam (Schlumberger), Maureen Hoffert (Hewlett Packard), Ken Kennedy (Rice University), Bob Knighten (Intel), Ross Knippe (Cray Computer), Chuck Koelbel (Rice University), John Levesque (Applied Parallel Research), David Loveman (Digital), Piyush Mehrotra (ICASE), Andy Meltzer (Cray Research), Tin-Fook Ngai (HP Labs), Rex Page (Amoco), Jean-Laurent Philippe (Archipel), David Presberg (Cornell), J. Ramanujam (Louisiana State University), P. Sadayappan (Ohio State University), Rony Sawdayi (APR), Randy Scarborough (IBM), Rob Schreiber (RIACS), Vince Schuster (Portland Group), Rich Shapiro (United Technologies), Henk Sips (Technical University Delft), Marc Snir (IBM), Guy Steele (TMC), Richard Swift (MasPar), Joel Williamson (Convex), Min-You Wu (SUNY Buffalo), Mary Zosel (Lawrence Livermore National Lab)

Also in attendance were 5 Apple PowerBooks, 1 laptop PC clone, and 1 Sparc notebook.

Memorable self-introductions

Ross Knippe - " From some other Cray."
James Cownie - "Also known as Dennis the Menace."
Joel Williamson - "From the still hiring Convex Computers." (to which Ken Kennedy responded, "Note that you won't get extra votes by hiring from the committee here.")

Logistics

Chuck Koelbel started the meeting with a short discussion of logistics for future meetings. His main point was that there were many more people interested in HPFF than could comfortably fit in the meetings. As he put it, "HPFF has gotten too popular for its own good." He asked for suggestions on balancing the needs of an open process with then need to keep the working group at a reasonable size. This was a particular concern since many people had asked to attend just one future meeting of the HPFF committee, namely the next meeting in DC. The discussion focused on two problems: handling the meetings themselves, and making the HPFF process as open as possible.

Ken Kennedy stated that it was very important to avoid "freezing out" interested parties from HPFF, both for technical and legal reasons. This principle was agreed to by everybody present. Ken further stated that he was satisfied with letting the plenary sessions of the working group grow as large as necessary (at least until the group was too large to fit in the room or make decisions). It is worth noting that extra chairs had to be brought into the meeting room to seat everyone this time. Discussions in the subgroups were somewhat more problematic as they must actually produce language drafts. The subgroup conveners all said that they were satisfied with the current situation (essentially, extensive email discussions and half-day physical meetings before the HPFF plenary sessions). The basic decision was, therefore, to continue allowing new working group members as long as physical constraints permit. As in the past, at most two representatives from a given organization are allowed (only one of whom can vote), and attendees are asked to commit to being at multiple meetings. Contact Ann Redelfs (redelfs@cs.rice.edu) for more information on joining the working group.

As far as allowing visitors at the HPFF goes, David Presberg suggested a compromise used by ANSI committees: allow the public to attend meetings, but all questions and comments must be submitted through a committee member. As he put it, "This allows the group to get its work done in a reasonable fashion, but still have public comment and a cheering section in the back." This policy will be implemented at the Washington meeting in July, in hopes that ICS attendees can participate. See below for information on attending this meeting -

The question of making HPFF as open as possible was discussed. In general, the entire group reaffirmed its commitment to open public commentary. There was some concern that people outside the working group did not feel their concerns were being addressed adequately, in particular a few voices who were only concerned with a single feature of the language. Although there was some feeling that this was more a problem of perception than reality, several suggestions were made to improve the situation. Marc Snir suggested a more frequent posting of the HPFF contact information, including information on the subgroup mailing lists. These discussions are already public and widely read. Ken Kennedy expanded this by saying that any question to HPFF should receive an "official" response. In the future, this policy will be adopted for any comments sent to the working group mailing list (hpff-core@cs.rice.edu). As far as possible, official responses will also be given to questions sent to the subgroup mailing lists. This was given a slightly lower priority, however, because the active discussions on most of those lists tends to answer questions very effectively and incorporate outside suggestions into new drafts.

European Developments

Henk Sips gave a short report on HPFF-related developments in Europe. The remainder of this section is his words.

There was a meeting in Brussels on May, 19, at which approximately 25 people were present. The subject of the meeting was to discuss the developments in HPFF and to what extent Europe's role could be defined.

It showed at the meeting that there was definitely great interest in HPF developments. Some people in Europe wanted to set up something similar, but I opposed this firmly with the backup of most people, after informing the audience in what speed HPF is progressing.

Having this settled, the discussion went on about how Europe should be involved in HPFF. Some people opted for sending representatives from a European forum to HPFF, but this soon turned out impractical for a lot of operational reasons. For instance, this requires for a person (delegate) a lot of extra time for gathering information and reporting back. Nobody could get this time from his/her employer.

Since two ESPRIT III projects have direct involvement in making HPF products (PREPARE and PPPE) it is likely that at least representatives from one or both projects will take part at the HPFF meetings. These people (e.g. me, Clemens Thole) were willing to report on the progress of HPFF.

It was decided to plan a second meeting on June 22 in Brussels, to report on the Dallas meeting in June (will be done by me). In that meeting it will be decided on next meetings and perhaps organize a larger meeting to inform and discuss the proposal with the European scientific community.

Clemens Thole will serve as a convener for the European meetings and has installed a mailing system at GMD for distributing announcements and other material. Interested parties can contact gmap11@gmdzi.gmd.de for more information.

Data Distribution Proposal

Guy Steele presented of the draft HPFF data distribution model. The entire model is contained in the file Distribution.tex , available by anonymous FTP (see above). It should be noted that the proposal was dated 1:00am, June 9, 1992; even the working group members were impressed by this kind of turnaround. Guy had stayed up late making modifications suggested at the subgroup meeting the day before, and deserves a lot of credit for generating an excellent document under a lot of time pressure. Because the proposal is readily available, these notes will concentrate on the discussions themselves and the resulting modifications to the document.

The first group of changes dealt with redistribution in subroutines. The sense of the subgroup was that there was real trouble with redistributions in subroutines having global effects. Therefore, a decision was taken to align a dummy argument with a template that is a copy of its actual, rather than with the true template. This means (among other things) that dummy redistribution does not affect its actual or other arrays aligned with the actual. After a long debate over propagating redistributions of dummy arrays back to the caller (as an option, since the default now disallowed this) the group voted that idea down as well.

The suggestion was made to dump REALIGN and REDISTRIBUTE, since they could be implemented by explicit copy operations. A unanimous straw poll voted to keep those features.

The next discussion centered on global COMMON variables. In particular, the following pathological case was presented:

```
COMMON /FOO/ A(100), B(100), C(100)
COMMON /BAR/ E(100)
TEMPLATE T(100)
REDISTRIBUTABLE T
ALIGN WITH T :: A, B, C
ALIGN WITH T :: E
DISTRIBUTE T(BLOCK)
CALL BAZ
...

SUBROUTINE BAZ
COMMON /FOO/ A1(100), B1(100), C1(100)
TEMPLATE T1(100)
REDISTRIBUTABLE T1
ALIGN WITH T1 :: A1, B1, C1
DISTRIBUTE T1(BLOCK)
...
REDISTRIBUTE T1(CYCLIC)
```

The $64 question is, "Does E get remapped?" Since it is invisible in BAZ, the principle of least astonishment would argue against remapping. But this causes an effective dealignment of E from A, B, C, and T (as a consequence of having T and T1 being identical). This is, of course, not possible either since E is not declared REALIGNABLE. Guy pointed out that the other choice was possible for answering the question, but led to other undesirable behavior. Marc Snir suggested that a linguistic theory is needed for explaining these cases; Guy agreed, but stated that none had been agreed to yet. Vince Schuster pointed out that the problem was the scoping of T and T1, to general agreement from the audience. Guy noted that the current proposal is that all templates are global, and this has other nasty consequences. Ken Kennedy kept the discussion moving by stating that these matters were too technical for the whole group to decide in the plenary

session; he suggested separating this out, and seeing if the group could agree with the rest of the document. Robbie Babb got in the last word by saying "We can't solve this, but I'm glad you bought it up."

Guy Steele continued with some other changes. The SMOOTH distribution (a arguably better-balanced version of BLOCK) had been eliminated. To allow easier handling of wraparound boundary conditions, template indexing now wraps cyclically. For example,

```
REAL A(100)
TEMPLATE T(100)
ALIGN A(I) WITH T(I+10)
```

maps element A(1) to T(11), A(90) to T(100) , and A(91) to T(1). To simplify implementation and semantics, conditions are added to the alignment that restrict wraparound to only one time through the template. That is, you can't ALIGN the even elements of A with the odd elements of T and vice versa.

Another extension, which was passed on to the intrinsic group, was to allow NUMBER_OF_PROCESSORS to take an array as argument. The return value would be the number of processors in the processor array associated with the data array. Passing a template or PROCESSOR array directly was not suggested, as these constructs are only allowed in HPF directives that appear as program comments.

The proposal would allow distributing COMMON blocks as single entities. The key to this approach was to treat named COMMON as a 1-dimensional array for use in ALIGN. This array would be indexed by storage unit (in the Fortran 90 sense), and the types of storage units in the COMMON must all match.

A new constraint was placed on the number of abstract processors declared in an HPF program. It must be legal to have a virtual processor array size match the value of NUMBER_OF_PROCESSORS(). A given implementation might accept other sizes as well (and the expectation was that this would happen), but the group chose not to force other numbers of processors to be implemented, as certain machines might find these options very inefficient.

Rob Schreiber pointed out that Fortran pointers were still on the table and no restrictions on them appeared in any document. Some constraint was needed; Guy agreed. Ken Kennedy, however, proposed a vote on at least some part of what was already in the proposal. The group agreed with this, but wanted to wait until after lunch to allow study of the new proposal. The time remaining before noon was taken by the (much shorter and less controversial) Fortran 90 subset proposal.

Language Subset Proposal

David Loveman and Mary Zosel presented their subgroup's proposal for a partial Fortran 90 subset to be used by HPF. A point made early and often in the presentation was that they were only isolating the sections of the Fortran 90 language that were directly applicable to the array sublanguage, which was the only task they had been set at the last meeting. Future HPFF meetings could thus extend the presented subset (and were, in fact, expected to do so). David had submitted a more general proposal to the mailing list earlier, but the group had explicitly decided to limit its scope on this issue.

The features included in the subset were given in a handout (Subset , available by anonymous FTP as explained above). In brief, the list included

- -the Fortran 77 standard except storage and sequence association (at least in their full generality)
- -MIL-STD-1753 (including goodies like DO-ENDDO)
- -array sections, including subscript triplet notation and vector-valued subscripts

-arithmetic and logical operations on whole arrays and array sections
-array assignment and masked array assignment (WHERE)
-array-valued external functions
-automatic arrays
-ALLOCATABLE arrays, ALLOCATE and DEALLOCATE statements
-assumed-shape arrays
-scalar, inquiry, elemental, and transformational intrinsic functions
-attribute specification statements: INTENT, OPTIONAL.
-INTERFACE blocks
-a few syntax improvements

The natural question to ask after that list was, of course, "What isn't in there?" David responded with another list, which he did not guarantee was complete.

-control flow
-"object-oriented" declarations (listing all attributes in the declaration line)
-some syntax improvements, including long variable names and free-form source
-KIND parameters
-pointers
-modules and user types
-internal procedures
-defined operators
-array constructors (as he said, "We ducked the issue")

After a call for clarification, Ken Kennedy responded that this would be a "permissive vote" --- the working group can be add to this subset later. Indeed, Mary Zosel commented that "We do believe that other things want to be here."

Peter Highnam commented that array constructors are very useful and questioned why they had been left out. David Loveman responded that full constructors (with multiple implied DOs) are very complex and they didn't want to delay HPF implementations unnecessarily. A short discussion resulted in adding constructors without nested DOs to the subset.

Vince Schuster moved to remove anything to do with optional arguments (in particular, the inquiry functions checking for the presence of optional arguments) from the proposal. This was seconded by Andy Meltzer. After clarifying that optional arguments in intrinsics would still be allowed, a vote was taken. 8 were in favor of removing the optional features, 10 were opposed, and several abstained.

Finally, a vote was taken on the whole proposal. 22 were in favor, 0 were against, and 2 abstained. The proposal was therefore accepted, and as Ken Kennedy put it, the group took a coffee break feeling that it had accomplished something.

Data Distribution Redux

After lunch the working group reconvened to consider the data distribution proposal in detail. This was considered to be the key point of the meeting, as it related to the defining new feature of HPF. As in the morning session, Guy Steele led the discussion, running through the proposal one section at a time.

1. Summary

The goal of this proposal, these features are to give advice on data placement in parallel machines with memory hierarchies. The basic model is a 3-stage mapping from data array to abstract template to virtual PROCESSOR arrangement to physical processor. HPF only defines the first two stages of this mapping; vendors may supply the last stage, or use a default. Every array is always aligned (although its template may be anonymous). Similarly, every template is always distributed (may be by default).

Robbie Babb started the discussion by asking if all data *objects* have some alignment, even scalars. Guy agreed they should.

Henk Sips objected to a statement in the proposal that "distribution statements are carried out at entry to a program unit." He believed it was more accurate to say that distribution and alignment were compile-time operations. Guy Steele pointed out that specification expressions (for example, array bounds) might only be available at runtime. He did concede that the current wording might be confusing, but did not know of a clearer way to make the point accurately.

Marc Snir claimed that HPF should require that default alignments and distributions must be expressible in HPF. Richard Shapiro amended this to say that the defaults were expressible by the alignments and distributions for the machine under consideration (thus allowing complex analysis). This point would reappear later in the discussions.

A few editorial changes were also suggested for the figure on the first page.

2. Lexical conventions

This section consists of BNF for the general HPF directives.

David Presberg asked if empty lines were allowed in the middle of HPF directives. Maureen Hoffert asked the same question about comments in the middle of directives. A straw poll found nobody in favor of interspersed comments and lots of "don't care" responses. The proposal was amended so that the next comment must be part of the same HPF directive. The whole discussion was regarded as arguing over the inheritance of warts from older Fortran versions.

3. TEMPLATE

A TEMPLATE is an abstract space used for defining mappings. It has an optional shape list (implying that a program can have scalar templates). It can also be defined using the DIMENSION attribute as for arrays.

David Presberg started the discussion by asking, "What is the scope of TEMPLATE identifiers?" Marc Snir brought up the related issue of name conflicts with other program identifiers. Guy Steele said they should be in the same namespace as variables (if local), to Pres replied, "or externs if we go that way." Ken Kennedy and David Loveman suggested sidestepping the issue in the general meeting by defining distribution and alignment within a subroutine now, redistribution and global mappings later.

As David put it, "the document loudly says nothing about equivalence between templates." Guy Steele agreed HPF can't declare equivalent templates now, and will need changes in this area in the future. He suggested piggybacking that feature on the Vienna Fortran language proposal.

The issue of scope came up again when Maureen Hoffert pointed the group to Section 14 of the Fortran 90 standard. Checking the reference quickly, Guy Steele noted that "if we were to say that templates were local entities of class 1, then everything is perfectly clear." After some prompting, he explained that this means templates would follow the same scoping rules as local variables and several other related entities. After some discussion, the group agreed that this point was getting too technical to hash out in a plenary session.

Rob Schreiber asked if a template had the same scope as arrays, to which Guy responded that scope is attribute of names, not objects. He promised to adopt the Fortran 90 language to define scoping of templates.

4. PROCESSOR

HPF provides abstract processors arrangements that look syntactically like a type declaration. The processors are arranged in a Cartesian grid. The scope of a processor grid name (although it did not appear in the draft) was taken to be the same as class 1 local entities (i.e. variables, or templates).

Robbie Babb started the discussion by asking, "Where can processor grid names be used?" Guy Steele responded that at present the only place was the ONTO clause in an ALIGN statement. Future extensions might find a place for it in intrinsic calls, or in the FORALL ON clause.

Robbie's next question was whether the processor array can change size dynamically. Chuck Koelbel noted that this conflicts with the static distributions already defined in HPF. Some further discussion led to the final answer that processors were allocated staticly at least within routine. This relates to the question "What processor arrangements are accepted?"

Marc Snir asked that Guy remove the explicit physical processors examples from the proposal. Guy agreed, saying they were "intended as exhortations to the vendors here," not as standard HPF examples. After a short digression into how various processor arrangements could be implemented, Ken Kennedy took the opportunity to say "I see a motive in footnotes to compiler writers. An annotated version of this document would be useful." Mary Zosel noted this was an advantage of this group --- we don't have to stay away from implementations. Guy agreed, saying that this proposal was a working document, not the final HPF draft.

The group then broke for lunch.

5. ALIGN and REALIGN

ALIGN and REALIGN specify the mapping from data arrays to templates. ALIGN defines a static mapping, while REALIGN can dynamically bind arrays to different templates. Syntax is given to align bunches of arrays to same the same template, as this is expected to be a common case. An alternate align-subscript syntax was given inspired by Bert Halstead; the intent was to restrict the alignment mapping to a linear expression. Some language is also given to map array dimensions to template subscript triplets by generating a new mapping expression. Wraparound indexing of templates was added, with constraints to avoid interlacing of subscripts and ensure that the array only wraps around the template once. For purposes of this meeting, the section about alignment to dummy parameters was ignored.

The first subject of the discussion was why wraparound indexing was needed. Henk Sips pointed out that efficient matrix algorithms often use cyclic shifting. Ken Kennedy noted that Fortran D has the possibility for new errors if you can't wrap subscripts (or otherwise handle "off-the-end" template elements).

Robbie Babb asked about the semantics of a scalar that is replicated. Guy Steele responded that it is a compiler hint only, as everywhere else. The consistent sequential semantics of HPF therefore mean that the scalar is changed by a shared update. In any case, all processors will see a consistent value for any data object.

David Presberg and Robbie Babb objected to the phrase "same template element" in the draft, saying that templates, as abstract entities, had no elements. Guy promised to catch this.

Piyush Mehrotra asked for clarification: can a 2-dimensional array be aligned with a 1-dimensional template by an expression? Some discussion explored the constrained functions allowed in an ALIGN statement. In essence, the expressions are chosen so that the subscript in each template dimension is a linear function (with wraparound) of at most one array subscript dimension. Due to the length of the discussion, Guy concluded that more English was needed in this section of the proposal to make the form of the expression clear.

Peter Highnam pointed out a problem with the alignee syntax in the proposal. Guy Steele modified the syntax to move the COMMON block name case from align-name to alignee

6. REALIGNABLE

REALIGNABLE is an attribute of an array; the program can only apply the REALIGN operation to a REALIGNABLE array. This is in analogy to ALLOCATABLE in Fortran 90. One constraint is put on these arrays --- a program cannot ALIGN to a REALIGNABLE array.

There was little discussion on this section, as the dynamic distribution features were not being voted on at this meeting.

7. DISTRIBUTE and REDISTRIBUTE

DISTRIBUTE and REDISTRIBUTE define mappings from templates onto processor grids. As with ALIGN and REALIGN, DISTRIBUTE is static and REDISTRIBUTE is dynamic. BLOCK and CYCLIC patterns are defined, as well as * for undistributed dimensions. All have equations performing the assignment to processors. The SMOOTH distribution proposed at the last meeting was discarded. The ONTO clause is optional, defaulting to an anonymous processor grid. If ONTO is omitted, then nothing can be said about relationship of templates or arrays to each other.

Rob Schreiber started the discussion by asking Guy to make explicit that distribute isn't needed for simple cases, i.e. arrays of the same size are almost guaranteed to align with each other. David Loveman argued that Rob's comment belonged in the summary rather than the body of the proposal. Andy Meltzer went somewhat further, saying there was a need for descriptions of host defaults behaved together. This led to a lively discussion, Robbie Babb arguing that "We can't legislate against incompetence" and Rich Shapiro claiming "The user should expect the compiler to do the obvious." Guy Steele Claimed the question was whether HPFF should legislate defaults.

David Presberg asked what had become of BLOCK_CYCLIC, and was informed that it could be specified by a parameter to BLOCK.

Rob Schreiber gave a surprising example of the use of ALIGN and distribute:

```
ALIGN B(I) WITH A(2*I)
DISTRIBUTE A( BLOCK(3) )
```

The result : processor 1 gets elements 1 and 2, processor 2 gets only element 3, and so on. The group was not expecting this to work, but agreed that it was legal. Rob then offered an example he couldn't do (but that was needed for certain real codes on the CM-2). The picture Rob gave is difficult to reproduce, but in essence he was mapping a 1-dimensional vector onto a 2-dimensional array of a different size so that each element of the vector corresponded to one block of the array (in column-major order). Because of the constrained ALIGN and REALIGN expressions, it did not appear that this mapping could be achieved by the HPF patterns. Ken Kennedy kept the discussion moving by asking anyone who thought of a brilliant way to fix this to add that feature later

The suggestion was made to delete one syntactic example

```
DISTRIBUTE ONTO Q :: /CRUFT/
```

as no longer valid because of the decision to treat COMMON as arrays, not as TEMPLATEs. Marc Snir pointed out that an array name can stand for its template, bringing an audible "Oooh" from Guy Steele. After a discussion of what is legal where, the proposal was changed to reverse the earlier syntax change and reinstate the CRUFT example. Guy's reaction was "the wheel of scientific progress keeps on turning."

Mary Zosel asked if it was intentional that this proposal did not apply to blank common; it was.

8. REDISTRIBUTABLE

REDISTRIBUTABLE is an attribute of templates, just as REALIGNABLE is an attribute of arrays. REDISTRIBUTE can only be applied to a template that has been declared REDISTRIBUTABLE.

Rob Schreiber claimed that

```
REDISTRIBUTE A(BLOCK)
```

was nonintuitive because it drags aligned arrays along with A. A long discussion ensued, including more pathological examples such as

```
ALIGN A(I) WITH T(*,3*I+1)
REDISTRIBUTE A(BLOCK)
```

Nobody had a ready idea how this should be interpreted, other than that it probably did something to the second dimension of T. Rich Shapiro and Piyush Mehrotra proposed disallowing redistribution of an array if template was available in the same scope. Guy Steele suggested the rule that you cannot use same name in the same program unit as both alignee & distributee. Ken Kennedy suggested revisiting the issue at a later date.

The group then took a well-deserved break.

6. OOPS!

After the break, Guy Steele admitted that he had blown it on the REALIGNABLE syntax. Despite claims to the contrary in the proposal, there was a possible ambiguity due to insignificant blanks

```
REALIGNABLE MISH
REAL IGNABLEMISH
```

(This was considered somewhat of a victory, however, since neither of the above was ambiguous with "REALIGN ABLEMISH"; that statement needs a set of parentheses.) Robbie Babb jokingly suggested the "fix" of requiring REALIGN to always operate on at least two templates. Guy went back to the syntax drawing board.

9. Combined Directives

This section deals only with Fortran 90 object-oriented syntax for DISTRIBUTE and ALIGN. F90 can only mix attributes in type specifications. The question is, do we want to keep that restriction in HPF, or do we want to allow more general combinations?

After some discussion, Rob Schreiber moved that HPF allow multi-attribute statements. We can merge this feature with Fortran 90 later. The motion was accepted by acclamation.

10. COMMON and EQUIVALENCE

Essentially, this section borrows the sequence association proposal from Group 1.

Rob Schreiber asked if we need both sequence and storage association proposals. David Loveman claimed that this was a subtle difference that users won't get. Others agreed, most memorably Rich Shapiro. His description of responses to an informal storage and sequence association survey was "What the bleep is this stuff?" This led to a proposal to limit the directives to only SEQUENCE and NO SEQUENCE (blank optional). Some question was raised over whether this conflicted with the previous use of SEQUENCE in Fortran 90. Guy Steele was unconvinced this was a problem, but promised to check on it off-line.

Rich Shapiro asked how this related to the group 1 proposal. Mary Zosel replied that Group 1's proposal tells how to determine if something is sequential, this one tells what to do with it. Peter Highnam questioned how the default should be set - SEQUENCE or NOSEQUENCE? Rob Schreiber said according to their proposal the compiler should allow setting and/or changing the default; Guy Steele deferred to their judgment.

Peter Highnam then asked if objects with the SEQUENCE attribute can be aligned. Rob answered that variables with SEQUENCE cannot be explicitly distributed, and one way to get SEQUENCE is being in a sequential COMMON. Mary Zosel added that if a local array is equivalenced to other local, then you can't distribute either one. Piyush

Mehrotra pointed out that users may want to send safe sections (i.e. a reshaped section of a distributed array they know is local) to subroutines. This argument would return later.

Randy Scarborough unveiled his objection to the Group 1 proposal --- a user can't distribute sequential arrays, but can distribute sequential COMMON blocks. Piyush Mehrotra asked why are there two different treatments, and received the answer that there were two proposals by different people.

Voting

Ken Kennedy started steering the group toward a vote on the data distribution proposal by summarizing the amendments to it. The treatment of COMMON was deferred for discussion with the Fortran 90 subgroup. Also deferred to future meetings were decisions on subroutine interface issues, distribution of COMMON and other globals, template scope, and REALIGN and REDISTRIBUTE. Not covered in the current draft were the identity of processor grids, and POINTER, TARGET, SAVE, and ALLOCATABLE arrays. It was then moved and seconded that this report as amended was on the floor and open for amendments.

Guy Steele compared this approval process to producing a document by simulated annealing.

A motion appeared to defer a decision on the specification of default ONTO (it is currently completely unspecified; some would rather that some information about alignment is retained). The amendment was accepted by a vote of 17 to 4 with 2 abstentions.

A final vote on the (now heavily amended) proposal was then taken: 23 in favor, 1 abstain. The proposal was accepted.

Sequence and Storage Association

Mary Zosel started the discussion of sequence and storage association. The first topic was user feedback on the "conservative" approach proposed at the last meeting, in which each COMMON block was either all distributable or all nondistributable. Mary's informal survey of Livermore users indicated that almost all liked the simple, conservative approach. The major dissenters were users porting large codes that were full of storage association tricks. Rich Shapiro reported that answers to his survey fell into 2 classes: "What does this mean?" and "Association is a bad idea." Peter Highnam liked the clean distinction between types of COMMON blocks, but had received a few strong comments that this would make for tough porting.

Rob Schreiber then presented a slightly revised version of the storage and sequence proposal. This is available as the file Association by anonymous FTP. In overview, it defines classes of "sequential" and "nonsequential" variables and COMMON blocks. A variable is distributable if it is allowed to appear in a DISTRIBUTE or ALIGN statement. There are a few rules to determine if a variable is sequential or not, and if a COMMON is nonsequential or not. The most controversial rule is that all variables in a sequential COMMON are sequential. Another interesting rule is that because compilers can't check nonconforming definitions in separately compiled modules, array-valued functions can't be sequential. (This avoids certain problems in funky call chains.) The only substantial changes from the proposal at the last meeting was that character arrays are not special cased any more; they are treated just as other arrays. In particular, there are now rules for "bad character arrays" and "good character arrays."

The discussion was spirited. The most vehement objections were the restrictions on an otherwise distributable array in sequential COMMON. The effect of these rules is that all variables to be distributed need to go into "pure" COMMON blocks. Several implementors pointed out that there were many cases that are now classified as sequential that can be handled by current compiler technology. This discussion did not reach a conclusion, and was postponed until after dinner.

Another long discussion centered on whether storage association or no association should be the default for arrays. General consensus was that for new programs, no association should be the default. For old programs being ported, storage association should be the assumed situation. Both approaches seemed reasonable for different cases. Since nobody seriously suggested compilers checking the file creation date, the group's recommendation was that compilers have a switch for either option. The more honest ones in the crowd admitted this was ducking the issue.

At this point the group broke for dinner, reconvening later for an evening session.

After dinner, Ken Kennedy presented what he called the "most liberal COMMON proposal." Joel Williamson immediately reminded him that "you can't say that in Dallas." Nevertheless, Ken continued with his modification of the Fortran 90 subgroup's storage association proposal.

1. The default for COMMON is nonsequential storage
2. Nonsequential COMMON is a list of (possibly aggregate) objects such that
   a. The list of contained objects is everywhere the same length
   b. Corresponding objects in the lists have the same size
   c. Corresponding objects have the same sequentiality

   If any of (a-c) does not hold, the programmer must declare the COMMON to be sequential (or interprocedural analysis must do so).
3. Nonsequential COMMON may intermix sequential and nonsequential objects.
4. Objects linked by equivalence are considered part of the same aggregate, but only the objects named in EQUIVALENCE are associated.

He also gave an example:

```
REAL A(100), B(100), C(100), D(100), E(100), F(100)
COMMON /EX/ A,B,C,D
EQUIVALENCE ( B(51), E(1) )
EQUIVALENCE ( E(51), C(1) )
EQUIVALENCE ( D(51), F(1) )
```

Arrays B, C and E form an aggregate, as do D and F. (The Fortran 90 standard actually makes the second, redundant EQUIVALENCE illegal, but many compilers accept if since it is consistent.) A is nonsequential, but the other arrays are sequential. Ken proposed that A could therefore be distributed, and possibly the aggregates (treated as a whole) B-C-E and D-F.

At this point, Randy Scarborough argued that COMMON by its nature has to have the notion of storage association, because the underlying mapping used in EQUIVALENCE implies the association of arrays not mentioned in EQUIVALENCE. Ken agreed that changing this was a radical suggestion, but claimed that it was reasonable given that COMMON was nonsequential by default in his proposal. Richard Swift noted that this changes the semantics of programs (to which the audience basically groaned "yeah, we know").

Joel Williamson and Guy Steele noted that a quick and dirty implementation of nonsequential COMMON with equivalence was to make every aggregate in COMMON into its own COMMON block.

In answer to a question, Ken Kennedy identified his target user as someone running a program on both a workstation to a supercomputer, possibly in the process of porting the program. Since a legal HPF program can't assume storage association (under his proposal). any legal HPF program is also legal Fortran 90 if the distribution directives are ignored. The objection was, of course, that legal Fortran 90 was now not necessarily legal HPF. Ken replied that the only problem arises when the F90 program depends on storage association.

Piyush Mehrotra asked for clarification: Now that COMMON was divided into aggregates, must the aggregates (not merely the COMMON blocks themselves) be same in all procedures? Ken answered yes. Mary Zosel pointed out that in that case a new EQUIVALENCE in another routine could invalidate a program. Ken claimed that could happen with the previous COMMON proposal.

Marc Snir suggested the simpler rule that distributable arrays could be in COMMON if their declarations were always consistent

Ken Kennedy then started setting up a series of straw polls among the proposals. He listed the active proposals as

Group1 - Any sequential variable makes the entire block sequential.

Ken0 - Relax Group1 for variables not involved in an EQUIVALENCE (after a bit of thought, Ken withdrew this because there was no difference from Ken1).

Ken1 - Relax Group1 for variables that are not part of aggregates.

Ken2 - Aggregates distributable if the EQUIVALENCE set that created them matched everywhere.

Snir1 - The default COMMON distribution is used for each aggregate or unmatched objects. (once Marc explained this, Ken withdrew Ken2).

Richard Swift quickly campaigned for the KISS principle before the polls were taken.

The results of the first straw was

Group1 versus all other proposals - Group1 9, others 14, several abstains

Discussion then erupted again. Rob Schreiber pointed out that Group1 doesn't preclude liberalizing later, and Richard Swift noted that modules work without problems. Richard then pointed out that the group that just voted for more feature functionality needed an advocate in the Fortran 90 group, since there had been little dissension there.

Rich Shapiro made the argument that if we're going to break association, let's be as blatant and obvious as we can. He suggested doing this by making any EQUIVALENCE, anywhere in the COMMON make the whole block sequential. A great cry went up that this was precisely the Group1 proposal.

Ken Kennedy said his goal in starting this discussion was to show that the Fortran 90 subgroup was throwing out a lot, maybe too quickly. He volunteered to work with them on liberalizing their proposal. After some discussion, there was agreement to accept the current proposal as a "lower bound" proposal, similar to the subset proposal. A revision will be presented at the next meeting.

The discussion then moved to sequence association. This turned out to be much less controversial than storage association, perhaps because it is a simpler case. Two straw polls were taken:

Rule SE-1 : Disallow passing sequential objects to nonsequential dummy arguments. 24 in favor, 2 abstain.

Rule SE-2 : Nonsequential character arrays. 18 in favor, 4 abstain.

A new policy question arose: should HPF preclude distributing sequential COMMON blocks and sequential variables? The argument made was that lots of association doesn't really require sequential storage. Marc Snir noted that, by the current draft, every array has some distribution which must be describable by the particular implementation. He questioned whether this was consistent with distributable common. HPF should be very careful when deciding not to have distribution for some object. Two straw polls were taken:

Should sequential common be distributable as a 1-dimensional array? 14 yes, 8 no

Should sequential variables be distributable? 2 yes, 16 no

Rob Schreiber noted before the second poll that a yes vote would open 5 new thorny problems. Ken Kennedy tried to impose the rule that yes votes would be forced to serve on the storage association committee.

Marc Snir asked one final question: Can HPF query any variable and get a meaningful answer? He claimed the directives were weak if the answer was "no." This could mean that the distribution function space was not closed under subsection, for example. This would, in turn have serious effects on defining the distributions inherited by called routines and in other places. The group believed that the set of distributions was closed as Marc wanted, but were too tired to provide a formal proof. There was agreement that the question should be investigated, however.

A Really New Proposal

Before adjourning for the evening, Andy Meltzer presented "a suggestion that may solve most of our problems." His slide was entitled "Low Performance Fortran"

I General guidelines

Explicit syntax and semantics are not necessary - they should be inferred from a few examples

TEMPLATE must be scalar, processor array size zero or one

Distributions only distribute onto a single processor but affect the order of the elements

Dummy arguments (heretofore referred to as parameters) may refer to a TEMPLATE not declared in program unit's scope

II Examples:

```
LPF$ DELAY 100
LPF$ TEMPLATE T,S
LPF$ PROCESSORS P(1,1,1,1), P2(0)
LPF$ STORAGE WITH GAPS :: X
LPF$ DISTRIBUTE, RANDOM, BY_REFERENCE T ONTO P
```

The group quickly decided to call it a night.

FORALL group

Chuck Koelbel started the meeting on Wednesday with a discussion of the FORALL subgroup's discussions on Monday. Essentially, the group had become somewhat stymied because of a disagreement over the breadth of coverage of the FORALL and its relatives. Single-statement and block FORALL constructs with SIMD semantics had previously been accepted by the working group. The current FORALL proposal seemed adequate for Geoffrey Fox's synchronous problems, but little else. Chuck argued that FORALL should also encompass loosely synchronous and embarrassingly parallel problems. Others had differing views, up to and including Clemens Thole's MIMD capabilities discussed at the last meeting. One of the major points of disagreement was the inclusion of reduction statements in FORALL constructs and DO INDEPENDENT loops. Although parallel methods for these operations are well-known, they definitely involve loop-carried dependences, thus violating the usual restrictions on parallel constructs.

Rob Schreiber argued in favor of a fairly restricted role for FORALL. In his view, array operations corresponded to synchronous computations, and with relatively few extensions also handled loosely synchronous and embarrassingly parallel problems. FORALL should therefore concentrate on asynchronous problems using function calls. Some dissent from the audience claimed that asynchronous problems were defined by requiring dynamic synchronization, which was at odds with the SIMD semantics of FORALL and the KISS philosophy of HPF.

Andy Meltzer commented that the advantage of INDEPENDENT assertions that they could specify where work was done. Marc Snir believed this was the right issue, but that the problem was the underlying execution model. The user's model for control

parallelism is an underlying load balancing system. Mary Zosel commented on the need for constraints on called functions in this context. Unconstrained functions could create intractable compilation problems.

Ken Kennedy, thinking that the discussion was turning to features proposed for PCF, commented, "If one person says `parallel regions' they're dead."

Rich Shapiro pointed out that anything except simple assignment and functions opens a whole can of worms in language definition and implementation. David Presberg disputed this. Min-You Wu stated that the advantage of his proposal (marking blocks of statements that were guaranteed to be independent within FORALL constructs) had the advantage that any statement could be put in a FORALL if the user ensured its independence.

Ken Kennedy asked if the group would endorse SIMD style FORALL with INDEPENDENT assertions. Randy Scarborough claimed there were 2 meanings for INDEPENDENT: with or without private variables, and asked that they be considered separately.

Before any poll could be taken, John Levesque interrupted with the statement to Ken, "Yesterday you said something that made sense to me." It was a few moments before the group recovered from this break in character. John's point was that FORALL is an extension to Fortran, and therefore eliminates portability. Ken Kennedy's response was that one option was not to put FORALL in the HPF subset. David Loveman remarked that scalar implementations of FORALL were not hard to do (although efficiency might be challenging).

The straw poll on INDEPENDENT was then taken.

Should there be some INDEPENDENT in language? 21 yes, 5 no, 5 abstain

...with private variables? 15 yes, 12 no, 1 abstain

It was pointed out that private variables would not necessarily need to be explicit; for example, local variables in called functions could serve this purpose.

Load balancing was again discussed. Marc Snir stated that the concept of FORALL with load balancing interferes with data distributions. To evaluate this language feature, he felt the need to know what the implementation would be like on distributed memory machines. Rob Schreiber noted the capabilities of independent execution were already in single-statement FORALLs with functions. Peter Highnam agreed, saying all the user was doing was giving information by making the loop FORALL.

Ken Kennedy interpreted the results so far as an endorsement of single-statement and block FORALLs, and INDEPENDENT assertions without explicit private variables. The discussion then turned to reduction statements. Chuck Koelbel and David Presberg each gave examples of syntax to add reductions to FORALL constructs. Neither met with enthusiasm. Ken Kennedy pointed out that adding reductions was already on thin ice because it added lots of language features. David Loveman pointed out that the Fortran D reduction came from a language based on Fortran 77, which had different needs from the Fortran 90-based HPF. Rob Schreiber campaigned for exclusive use of the Fortran 90 reduction intrinsics; a discussion followed regarding the amount of memory required for temporary arrays to use these features. A straw poll on reduction statements in FORALL constructs netted 3 in favor, 16 against. Accordingly, the FORALL subgroup decided to drop reduction statements from their proposals.

Randy Scarborough inquired regarding the meaning of function calls in a FORALL. Rich Shapiro suggested arbitrary order semantics. David Loveman noted that the Fortran 8X semantics only ruled out side effects between the right- and left-hand sides. Richard Swift pointed out that these semantics are vague. Rob Schreiber proposed that HPF always specify the behavior of a construct, if possible in a deterministic way.

Richard Swift started the last topic of discussion by asking what statements could go in a FORALL. Andy Meltzer answered that INDEPENDENT assertions give lots of freedom, but otherwise the rules would be very complex. Rich Shapiro offered a plea: the

more features are allowed in FORALL, the harder it will be to implement. Therefore, HPF should avoid adding to FORALL. As Rich put it, "With single statement FORALL you can write statements that will make anybody's head spin!" The advice was taken seriously.

Intrinsics

Rob Schreiber presented the preliminary results of the intrinsics group. They proposed new intrinsics in 7 areas:

1. Extend MAXLOC and MINLOC to be like MAXVAL and MINVAL (by adding an optional DIM argument)
2. Enlarge the set of reduction intrinsics
3. Send with combine operations
4. Elemental operations for "manipulating bits in ways that the NSA finds interesting"
5. Parallel prefix and suffix operations
6. Sorting
7. System inquiry functions

Ken Kennedy asked Rob to concentrate on the areas of controversy; Rob replied that they hadn't had any controversy.

Marc Snir pointed out that these functions, while in the spirit of Fortran 90, were not related to data distribution and parallelism per se. He asked how appropriate they were, particularly in view of the burden they would put on other compilers. Vince Schuster took this further, asking what the value of making these operations intrinsics, as opposed to library calls, was. David Loveman suggested considering these operations as "if you have it, this is how to spell it" specifications. Rob Schreiber suddenly looked up and said "I forgot that this was controversial." Rex Page suggested ensuring these intrinsics could be implemented as Fortran 90 functions. David Loveman pointed out that users would want some functions (particularly the system inquiry functions) to appear in specification statements. The intrinsics subgroup agreed to investigate dividing the functions into true intrinsics and library functions.

Scheduling and Logistics

The last topic before the meeting broke into subgroups concerned scheduling future meetings. The dates had already been set, but details of the agendas were not fixed, or needed adjustment. The previous schedule for approving feature drafts was

July 23-24: Storage Association, Dynamic Redistribution
September 9-11: FORALL (and Similar) Constructs, Local Subroutines, Intrinsic Functions
October 21-23: Official Fortran 90 Subset, Miscellaneous Features
December 2-4: Approval of Final Draft

It was noted that no slack had been left for the data distribution features that had been deferred at this meeting. With Guy Steele's agreement, the first reading of those features was scheduled for the July meeting, with final approval planned in September. The rest of the schedule stayed as given. It was noted that the next meeting would be very busy, with four full-scale presentations to be made.

The question of parallel I/O was brought up, in particular that no time had been allotted for it in the schedule. Bob Knighten noted that the hpff-io group had been inactive, and recommended disbanding if no new action arose by the next meeting. Piyush Mehrotra volunteered to write something for the group. If the group remained together, it will present its proposal at the September meeting for possible approval in October.

The next major topic was logistics of the meeting in Washington. Piyush Mehrotra noted that lots of people at ICS would be interested in HPF, and the group should use this opportunity to involve the public. Ken Kennedy noted that David Loveman would be making an invited speech at ICS to "whet the appetite" of attendees, and suggested a reception Wednesday night to promote further informal conversation. Piyush Mehrotra expressed a fear of extraneous discussions in the plenary meeting if it were completely open, but the group felt that David Presberg's suggestion of routing all outside comments through committee members would address the worst of this problem. A rather stricter adherence to parliamentary rules than HPFF meetings usually observed would also help. There was general agreement to open the meeting up in this way and to arrange a birds-of-a-feather session during ICS to promote HPF.

The next topic was the schedule for that meeting, as it did not fit the now-standard 3-day framework for meetings. After some discussion, the schedule was set as subgroup meetings Thursday morning (July 23), followed by a plenary session starting at 1:30 in the afternoon and running into the evening. To minimize down time for dinner, a sandwich dinner would be ordered into the meeting room. The plenary session would reconvene Friday morning at 8:30 and end about 4:00 that day, allowing attendees to catch their flights home.

The final discussion was started by Ken Kennedy's question "Who's in charge of miscellaneous features?" He further suggested that HPFF needed "someone with a mind like a vacuum cleaner" to check the consistency of the draft. Mary Zosel suggested picking someone when the draft document was closer to final form. This led to the suggestion that "create document" needed a slot on the HPFF schedule, and that a document subgroup needed to be created. David Loveman volunteered to convene such a group. Chuck Koelbel, Guy Steele, Richard Swift, Alok Choudhary, Maureen Hoffert, and Rex Page filled out the committee. Guy further volunteered to massage drafts that were "close" into a common form. LaTeX was reaffirmed as the format of choice, augmented by a set of macros that Guy had used to produce his draft at this meeting.