HPFF Meeting Notes for the December 10-11, 1992 Meeting

High Performance Fortran Forum

CRPC-TR93316
May 1993

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

HPFF Meeting Notes

December 10-11, 1992
Dallas, TX - Bristol Suites Hotel
Notes taken by Chuck Koelbel

Executive Summary

This was the seventh meeting of the High Performance Fortran Forum working group. Its major purpose was to produce a "final" document by the end of December, and by the end of the meeting many remaining technical issues had been settled. A meeting will be held in March to discuss public comments on this draft, with possible revisions of the document. Preliminary plans were also made for a follow-on effort in 1994, and other activities to broaden the distribution of HPF.

Some of the more important outcomes of this meeting were as follows.

Data Distribution: The committee voted to require significant blanks in directives, and allow the syntax

```
!HPF$ ALIGN A WITH B
```

(which requires significant blanks for parsing). A mechanisms to allow global data to be aligned with variables local to a procedure (provided that the resulting distributions were consistent across) was also approved. A more general syntax for ALIGN expressions was introduced. Perhaps most importantly, the group of features for declaring alignment and distribution of dummy arguments was reworked to allow several possibilities:

- Declaring "Use the same mapping as the caller."
- Declaring "This was the mapping in the caller."
- Declaring "Use this mapping, regardless of the caller's."

PURE procedures: The committee voted to allow dummy arguments to pure procedures to be explicitly aligned.

HPF Library Functions: The committee voted to create a Fortran 90 module named HPF_LIB which would contain all HPF library functions and subroutines.

Subset HPF: The committee voted to include the EXTRINSIC feature in the HPF Subset, and remove the Fortran 90 character array sublanguage.

Dates and agendas for future HPFF meetings are as follows:

March 10-12	Discuss public comments on draft; possibly publish changed draft and/or responses to comments

Detailed Meeting Notes

Attendees

Alan Adamson (IBM Toronto), Robbie Babb (Oregon Graduate Institute), Ralph Brickner (Los Alamos National Lab), Marina Chen (Yale), Alok Choudhary (Syracuse), Tom Haupt (Syracuse), Maureen Hoffert (HP), Ken Kennedy (Rice), Bob Knighten (Intel), Chuck Koelbel (Rice), David Loveman (DEC), Piyush Mehrotra (ICASE), Andy Meltzer (Cray Research), John Merlin (Southampton), Rex Page (Amoco), Jean-Laurent Philippe (Archipel), P. Sadayappan (Ohio State), Rob Schreiber (RIACS), Vince Schuster (Portland Group), Rich Shapiro (Thinking Machines), Henk Sips (Delft University of Technology), Matthew Snyder (Lahey), Guy Steele (Thinking Machines), Richard Swift (MasPar), Joel Williamson (Convex), Hans Zima (University of Vienna), Mary Zosel (Lawrence Livermore National Lab).

Meeting Reports

The meeting started with two reports on recent meetings relevant to HPF. David Loveman outlined the results of the HPF workshop at Supercomputing '92, and Maureen Hoffert reported on the ANSI X3J3 meeting.

David Loveman reported that the SC '92 workshop had a generally positive response. This may have been a slight understatement, since he also reported that the room was overflowing at the beginning of the session, and the rear folding wall had to be retracted. The workshop outlined the language in some detail, and had very active audience participation. Loveman noted that one of the most interesting parts of that discussion was a debate "between two people, one who thought [HPF] was too high-level, one thought it was too low-level." At any rate, it was clear that the workshop was a success.

Maureen Hoffert then discussed a recent meeting of the X3J3 committee. The major result of this was a preliminary response (handed out at the meeting) to several questions from HPFF. The reasons for not including some specific features (such as the more general version of the MAXLOC intrinsic) in Fortran 90 were not technical, and these could be added to HPF. X3J3 noted that they did not have an up-to-date intrinsics chapter (one was being forwarded to them). The general consensus of X3J3 was that putting new functions into a module is the way to go (but they admitted they didn't examine alternatives in detail). In general, the committee was supportive of the HPF effort. The response noted that X3J3/WG5 will have a revision in 95, and was meeting in June to gather input. The HPFF group suggested that we should recommend FORALL and new intrinsics as future requirements; other recommendations came forward later in the meeting.

Data Distribution

Guy Steele led off the technical part of the meeting with a discussion of comments on the data distribution chapter. There had been lots of comments over the net, many purely typos and editorial matters. Here Guy just discussed the major technical points.

As Guy said, "First we revisited the TWITHEAD problem." This is the long-standing problem with insignificant blank ambiguity, illustrated by the example:

```
            ALIGN TWITHEAD WITH MORON
            ALIGN T WITH EADWITHMORON
```

(Actually, ALIGN did not have this ambiguity, but other constructs did and this form of ALIGN was often mistakenly used as an example.)

Ken Kennedy noted that he would just as soon not use this name in publications, but as it had already shown up on the net (and Guy had fruitlessly searched on-line dictionaries for less insulting examples), it seemed destined to be remembered this way. A fix had been suggested in the subgroup meeting the day before, which consisted of changing all “prepositions” in directives (WITH, OF, ONTO) to the character string “->” as shown below:

```
            old:
            ALIGN A(K) WITH B(K)
            VIEW OF B :: Q
            DISTRIBUTE B(BLOCK) ONTO Z

            new:
            ALIGN A(K) -> B(K)
            VIEW -> B :: Q
            DISTRIBUTE B(BLOCK) -> Z
```

The recommended name for the new symbol/operator was the “thingee,” at least until something better was suggested. Several other ideas were suggested, including

```
            ALIGN A .WITH. B
            ALIGN A, WITH B
            ALIGN A # B ! (# is not in the Fortran character set)
```

And dropping the statement forms entirely, based on the idea that language evolution is toward the attribute form. Rob Schreiber suggested spelling “with” as “comma W-I-T-H” to general groans. The proposal from the committee was to use the “->” notation.

Andy Meltzer moved that the vote on this issue be done as 2 phases: “Should we change this?” and (if that passed) “What to?” This was agreed, and an official vote on the first phase was quickly taken. The vote was 10 in favor of changes, 6 against, and 7 abstaining. A discussion of the realistic alternatives then proceeded, eventually producing the list:

- Require significant blanks in directives (current state), but add no-parenthesis form of ALIGN (not currently allowed)
- ALIGN A -> B
- ALIGN A .WITH. B
- ALIGN A, WITH B
- Delete the single-align form, single-distribute form, single-view form

Guy Steele warned “We’re trying to eliminate a minor wart in the language by introducing a bigger one.” It was finally moved and seconded to require significant blanks in directives , but add the no-parenthesis form of ALIGN. The official vote was 15 yes, 0 no, and 8 abstain.

Guy Steele then moved to the next topic: COMMON and SAVE with respect to disappearing processor arrangement. The problem was that a programmer would often want to align COMMON to a procedure variable when there are no modules (this could be used, for example, to copy distributions consistently everywhere). The subgroup recommended that HPF

- Not allow SAVE to be DYNAMIC

• Allow explicit mapping of COMMON and SAVE variables to local processors arrangements that “disappear” on exit of the subroutine as long as those arrangements “reappear” in the same form.

This implies that simply replicating ALIGN and DISTRIBUTE directives has the expected effect, as long as the sizes of arrays and templates do not change. An official vote found 17 in favor of this change, 0 against, and 4 abstaining.

Ken Kennedy noted that changes like this meant that the next HPF draft should have a companion document of changes to the 0.4 document. It was later decided to add such a list to the beginning of the next draft.

Guy then moved on to the matter of assumed distributions for dummy parameters. He noted that the subgroup “spent a long time - way into the night - on this” without a completely satisfactory result. The basic problem is that a programmer may want a way to say “I know this parameter comes in distributed BLOCK.” This is possible to express using TEMPLATE, but a more elegant way was desired. The previous night’s meeting had ended with two proposals on the table. Rob Schreiber promised a single proposal after lunch, when the subcommittee had a chance to meet. (The results of that meeting are below, under the headings “Dummy Argument Distributions” and “More Dummy Distributions.”)

The next topic was which BNF grammar to use for align-subscript-use. The document had two grammars, a “generous” version that allowed many expressions that simplified down to linear expressions, or a “restrictive” version similar to the FORTRAN IV limitations on subscript forms (a*I+b, where a and b were integers). Guy (and others) noted that even with the “generous” form there would still be expressions for linear expressions that would not be recognized as linear. For example,

```
I**3+2*I**2-I*(I+1)**2
```

which, of course, is equivalent to `I`. This led into a brief discussion of the limits of formulations in BNF and other formalisms. Guy Steele said he would generate exact rules later if a poll favored an even more “generous” formulation than the current options. A vote between the existing syntaxes found 18 for the “generous” grammar, 1 for the “restrictive” grammar, and 4 abstaining. A straw poll to allow even more general expressions (subject to seeing Guy’s new grammar) found 12 in favor, 6 against.

HPF Subset and Fortran 90

Mary Zosel led the discussion of public comments regarding the HPF subset and Fortran 90 matters. Again, there were lots of editorial comments that were not discussed.

The first technical matter was a suggestion to add LEN to subset. This was not controversial, and was adopted by a vote of 19 yes, 0 no, 3 abstain.

The next technical question was “Is the character array language in the HPF subset?” Mary had thought so, but it was left out of the draft. A fast search of the on-line minutes (thank heaven for large disc space) did not find a vote to include those features. Richard Swift argued that including the sublanguage was not consistent, since other parts of character arrays were not in the subset. Rich Shapiro saw little added value in these features in the subset. David Loveman argued we shouldn’t try to add new features at this meeting, but Piyush Mehrotra noted we were going to go over the full draft as well. Rob Schreiber moved that character arrays be excluded from the subset, and that character array intrinsics also be removed. The vote fell 15 yes, 4 no, 4 abstain, and the appropriate changes were made.

The next topic was the observation that limiting arrays to 7 dimensions creates problems for processor inquiry intrinsics (because of the shapes of arrays that might be returned). The precise problem appeared to be a misunderstanding of the returned values, but the possibility of allowing many-dimensioned arrays was discussed on its own merit. After much talk, a straw poll to add arrays with more than 7 dimensions to HPF failed by a vote of 3 yes, 21 no, and 1 abstain. A further poll to forward this matter to X3J3 for the 1995 revision did pass, 19 yes, 1 no, 4 abstain.

Next up was a suggestion to add conditional compilation to HPF. A motion to recommend this to ANSI. in future standards was approved without formal vote.

EXTRINSIC procedures then came up for discussion. The subgroup felt that the EXTRINSIC chapter needed clarification, and were not sure whether it should be in the subset. In particular, the questions “What exactly does EXTRINSIC mean?” and “How is this different from just calling other language routine?” were unclear to the subgroup. Guy Steele explained that EXTRINSIC guaranteed that the called routine cannot observe non-synchronized actions (or cause HPF to see them on return), EXTERNAL could only guarantee this for (native) FORTRAN actions. Richard Swift observed that it was hard to separate language features and implementation features in this area. Mary Zosel worried that without EXTRINSIC there was no way to call graphics (and other) libraries. Guy Steele replied that vendors can (should be encouraged to) implement more than just the subset. Joel Williamson claimed this was not a problem as it was “almost impossible” to call a parallel library except from a single-threaded part of the code; Chuck Koelbel countered that Joel’s statement assumed an implementation model, and produced problems if the library written with another model. Several attendees worried HPF needs an external interface, and recommended that EXTRINSIC be in the subset to provide this. On this basis, the committee instructed the subset and EXTRINSIC committees to coordinate and propose a coherent model in March. Bob Knighten asked what EXTRINSIC was supposed to solve, and Guy Steele replied that it guarantees that data is where you said it was (and not moved by a compiler transformation, for example).It was finally moved and seconded to add EXTRINSIC to subset (and to treat this motion as a first reading, to be finally voted on in March). The straw vote in favor of this motion was 11 yes, 10 no, and 4 abstain.

The group then took a well-deserved coffee break.

FORALL and PURE

Chuck Koelbel started the discussion of comments on the FORALL statement and construct. It was very short, primarily reflecting presentation changes and technical points. Chuck then turned the floor over to John Merlin.

John proposed allowing dummy arguments to pure procedures to be aligned with other dummy arguments. His reasoning was that, since pure functions could be invoked independently and concurrently on sets of processors owning the arguments, dummy and local variables could be stored on those processors without introducing implementation overhead. For example,

```
!HPF$ DISTRIBUTE a(BLOCK,BLOCK)
FORALL ( i=1:n) a(:,i) = fun(a(:,i))
```

should not cause redistribution of the array on calls to fun. He then gave some strategies for implementing distributed arguments:

• Compile all arguments as undistributed

  Advantage: Caller can set up

Disadvantage: Storage, no concurrency
- Compile as though all dummies are distributed
  Disadvantage: slow

His proposal was to allow “ALIGN WITH *” for dummy arguments, and allow aligning locals with dummies and other locals. He also noted that this was a useful specification mechanism for library routines. Ken Kennedy considered this as major proposal, and recommended treating it as a first reading with more discussion in March; John and the rest of the committee concurred. Hans Zima supported the proposal’s objectives, but noted that the current language didn’t fit well. He proposed changing the restriction on dummy arguments to “DISTRIBUTE *” only. Piyush Mehrotra (after some discussion) suggested a more general phrasing: allow inheriting mapping, and come back to the required syntax after resolving the distribution changes. After more discussion, much of it arguing over how much of John’s reasoning was specific to his system and how much was general, a straw poll was taken. 16 voted to add this feature to the language, 1 against, and 8 abstained. On the question of syntax to be allowed, “ALIGN with *T” (with additional declarations of the actual distribution) netted 16 yes votes, 0 no votes, and 9 abstentions.

Intrinsics and Library

Rob Schreiber led the discussion of library and intrinsic functions. As always, there was lots of editorial trivia.

The most important substantive comment was relayed from X3J3 by Maureen Hoffert earlier. Piyush Mehrotra asked how adding an explicit module would interact with the HPF subset. Rob did not see a problem, since neither MODULE nor the library was in the subset. Rich Shapiro didn’t want to preclude the library being implemented as intrinsics, but the consensus was that this was not a problem with the explicit module approach. A vote to accept the library as a Fortran 90 module, named HPF_LIB received 18 yes votes, 0 no votes, and 5 abstaining. Marina Chen suggested changing the names of the new intrinsics to IALL, IANY, and IPARITY (by analogy to the reductions for .AND. .OR. .EOR.). The vote on this proposal was 8 yes, 0 no, 4 abstain.

Hans Zima posed two questions regarding the mapping inquiry intrinsics: the result if compiler ignores mapping directives and the result of inquiry for array sections. In the interest of time, Ken Kennedy responded “I’ll allow the questions to be posed but not the answers.” The intrinsics committee indicated they would resolve the questions off-line.

The group then broke for lunch, agreeing to reconvene at 2:30pm to allow the distribution subgroup meeting.

Publication of the HPF Language Specification

After lunch, a proposal was made to publish the final draft of the HPF language specification in “Scientific Programming,” a journal edited by Robbie Babb. To ensure that the draft would remain freely available, the copyright will stay with Rice University. In return for agreeing to publish the document, it was agreed that no other journal could publish it first, although later publication would be allowed. The proposal was approved without dissent. A tentative schedule for publication in Scientific Programming in June, and Fortran Forum in July was set; distribution by FTP and technical report was OK before June. The editorial committee would get the final version (after public comment) to Robbie Babb in March.

Distribution Redux

Guy Steele was the next presenter, giving the new grammar for alignment expressions. Linear expressions were defined by a recognition algorithm:

```
Let D be an align-dummy
Let m, n, p be any expressions not involving an align-dummy
An expression is linear in D if it cam be reduced to one of the following
    forms:
    n
    D
    D+/-n
    m*D
    m*D+/-n
using the following long list of reductions:
    (Complete list omitted here, but includes commutative, associative,
        and distributive laws, but not cancellation.)
```

An informal proof of termination was given (by defining a “charge” of any expression based on the number of operations involving D, then noting that the charge of an expression is never negative and that every reduction reduced the charge). The algorithm was also “sound by inspection” and “complete by hope.” After some questioning, the committee was convinced that the algorithm recognized linear expressions that were not expressed using higher powers (or other functions) of D. For example, “D**2-D**2+D” was not recognized because the squares could not be canceled. A vote was taken in favor of adding this functionality to the HPF specification, giving 17 yes, 2 no, 4 abstain.

After that decision, it was then moved and seconded to leave the added functionality out of subset HPF (thus simplifying implementation considerably). A first attempt at wording this was that “align-exprs must be of form a*D+b”, but Joel Williamson and Piyush Mehrotra argued for including the more general original case in the subset. This was taken as a friendly amendment, resulting in the wording “only one instance of align-dummy is allowed.” Vince Schuster argued for the simpler form, apparently persuasively. The vote to retain the more general linear expressions in subset HPF was 17 in favor, 2 against, and 4 abstentions. Another vote to restrict to expressions of the syntactic form a*D+b came out 16 yes, 2 no, 4 abstain, and Guy promised to rewrite the distribution chapter on that basis.

Dummy Argument Distribution

Rob Schreiber then presented the new proposal on distribution features for dummy arguments. The intent was to present the latest thoughts, but move any vote to the next day to allow committee members to think about the proposal.

The fundamental problem that the group addressed was that HPF always distributes a template at the lowest semantic level. In particular, when a programmer says to distribute a dummy argument it is not clear whether this means to distribute the dummy argument’s default template or (a copy of) the actual argument’s template. In addition, programmers would want to force remapping or assert a mapping. The result of all this was a syntax with the following form:

```
DISTRIBUTE P - use the natural template of the dummy
DISTRIBUTE *P - use the template of dummy, aligned as the actual is
    (i.e. inherit a copy of the actual’s template)
```

DISTRIBUTE [above] (CYCLIC) - remap to match given pattern, if needed
DISTRIBUTE [above] *(CYCLIC) - assert that the actual is cyclic, & leave it there
DISTRIBUTE [above] * - leave the actual where it is, and * must be replaceable by a valid distribution
DISTRIBUTE [any of above] [no ONTO clause] - accept where it is
DISTRIBUTE [any of above] ONTO PR - force onto processor array PR
DISTRIBUTE [any of above] ONTO *PR - assume it's already on PR

(This really made absolute sense at the meeting, thanks to a four-color poster with lots of arrows.) One of the committee members clarified that "*" was meant to mean "match with what goes here." The options could be combined in any way, allowing one to say, for example, that the actual was distributed by blocks but the dummy might be a subsection of that with a cyclic distribution. Ken Kennedy wanted more details, but Guy Steele and the committee insisted the discussion should move on, since "this was the easy part." A lot of eyes glazed over.

The presentation now moved to ALIGN. In these cases, P, Q, and R are dummies, X is a local (possibly a TEMPLATE), and Y a global variable.

ALIGN P WITH *Q - assert elements of actuals are aligned
ALIGN P WITH Q - remap data to conform to alignment
ALIGN P WITH *X - asserts alignment; X must have an explicit mapping
ALIGN P WITH *Y - asserts alignment; Y must have an explicit mapping

Ken Kennedy asked if the committee was putting this forward as a clarification or as new functionality? The answer was that it was new functionality. Some question was raised whether a change like this should go out in the next draft. Rich Shapiro claimed that HPFF would have to plug this hole before March. A straw poll found 21 in favor of adding it to the next draft, 1 against, and 2 abstaining.

Hans Zima argued against the whole feature, saying it was very complex, against the goals of simplicity, and had no implementation experience. He anticipated big problems with it in compilers. Rich Shapiro countered that CM Fortran (and other dialects) have similar features (layouts). Telling compiler not to remap is seriously needed. Thinking Machines' users have been screaming to add inherited layouts; he took this as an argument that HPF has to include them. Chuck Koelbel asked how much inheritance is needed (for example, could HPF get by with only a limited inheritance feature?). Rich said full arrays and array slices (dropping one or more dimensions) were definitely needed; he was less definite about general array sections. Vince Schuster noted that just alignment and distribution consistency checking would help seriously. Andy Meltzer suggested that the matrix of possible mapping statements might be too large, maybe we should stick to just a few boxes. Hans Zima claimed that Vienna Fortran users just need distributions (without ALIGN), and some discussion of this ensued. A subgroup meeting was scheduled for after dinned that night.

The group then took a coffee-and-aspirin break.

Editorial Matters

David Loveman presented the decisions of the editorial committee. In an effort to produce a somewhat consistent style throughout the document, there were lots of style/editorial comments. Maureen Hoffert presented a list of style guides

Refer to Fortran 90 standard as "Fortran 90", not ANSI or ISO
References to F90 BNF

- in italics
  - not Courier font
  - use correct terms
- Programming language keywords - uppercase, Courier font
- Use FORTRAN 77 (uppercase) and Fortran 90 (mixed case)
- Constraints - Use the "constraint" environment for consistent indentation
- Comments - Use "!" not "C", "c", or "*"
- Chapter structure - first paragraph is an overview, not numbered
- List conventions:
  - Lead in with ":"
  - ";" after each item
  - Next to last item "; and" or "; or"
  - Last item ends with "."
  - Initial word capitalized; watch excessive caps
  - Bullets if no ordering or reference requirements
- Use "HPF" and "HPFF" instead of English words
- Take care in usage of English word vs. HPF keyword
  - Keyword capitalized
  - English word lower case
- Use "dependences" consistently
- "Align" means ALIGN only, "distribute" means DISTRIBUTE only, "map" means their combination
- Be cognizant of implicit vs. explicit mapping
- Code follows free-form source for continuation lines (& at end of line); use universal format where possible
- Do not use tabs; use spaces (6) for paragraph indentation; 2 spaces in examples for level indentation (if possible)
- Spaces:
  - No space following "(" or preceding ")"
  - Spaces around assignment operator;
  - Spaces around highest level operator, not around other operators
  - Spaces on either side of "::"
  - No spaces around "," in x(i,j)
  - Best judgment in lists
- Always go for best readability
- \ICODE - italicizes lower case, makes subscripts active
- Capitalize keywords, best judgment on variables

(I just include this to demonstrate the care that went into drafting the document.)

A call went out for volunteers to read the draft when it was complete. Joel Williamson, Piyush Mehrotra, Andy Meltzer P. Sadayappan, Alok Choudhary, Mary Zosel, Alan Adams, Maureen Hoffert, Chuck Koelbel, David Loveman, and Guy Steele promised to proofread the document before it was distributed.

The official name of the next version was set to be the "1.0 draft" (hoping to distinguish it from the "version 1.0" that have typos corrected in March). A new section was added just before the table of contents to describe substantive changes from previous versions. Several other procedural questions for fixing errors were also settled.

The final agenda item for the day was setting the structure of the March meeting. As the purpose of this meeting was to respond to public comment, it was expected that more time would be needed for the subgroup meetings than usual. After some discussion, plans were made for a full 2-1/2 day meeting starting at noon March 10. In case of overwhelming volume of comments, plans were also made to hold reservations for April 14-16 for a further meeting.

The group then broke up for dinner at Pappasito’s, a well-known Mexican restaurant (actually, a chain of restaurants across Texas).

More Dummy Distribution

Friday started with Guy Steele presenting a new proposal for distributing dummy arguments. He began with a simple example of the previous night’s syntax:

```
!HPF$ DISTRIBUTE *TSAR*(*) ONTO *CZAR
! Or, in another form
!HPF$ DISTRIBUTE * * (*) ONTO *CZAR :: TSAR
```

After this and several more examples, the group decided that “*” was too overloaded. Some stars in DISTRIBUTE mean the same as in ALIGN, while some mean “leave this alone.”

Fortunately, a new solution was at hand: the subgroup had defined a new attribute called “INHERIT.”

```
!HPF$ DISTRIBUTE TSAR*(*) ONTO *CZAR
!HPF$ INHERIT TSAR
!HPF$ DISTRIBUTE *(*) ONTO *CZAR, INHERIT :: TSAR
```

The full proposal was as follows:

- Attribute INHERIT applied to a dummy means the dummy’s template is a copy of the template of the actual; without it, the dummy’s template is the “natural” template, coincident with index space of dummy.
- Syntax of distribute is

```
!HPF$ DISTRIBUTE D [(CYCLIC), *(CYCLIC), *]
   [ONTO P, ONTO *P, ONTO *]
```

- DISTRIBUTE may not have both options empty.
- Meanings in DISTRIBUTE are as yesterday: “*” means assert, otherwise move to conform
- INHERIT implies default of

```
DISTRIBUTE * ONTO *
```

Guy then gave several examples.

To say “just leave dummy D where the actual is” and nothing else:

```
!HPF$ INHERIT D
```

To say “leave D where it is, and the actual’s template was cyclic”:

```
DISTRIBUTE *(CYCLIC), INHERIT :: D
```

To say “leave D where it is, and its natural template is cyclic”:

```
DISTRIBUTE *(CYCLIC) :: D
```

Chuck Koelbel asked what limits (if any) were placed on statements made about the natural templates of dummies associated with actual arguments that were array sections. For example, could a programmer use devious strides to change a CYCLIC distribution into a BLOCK distribution and vice versa? Guy Steele replied that there would need to be explicit limits; generally, the proposal limit cases that must be handled as in linear expressions. He promised an exact description later in the afternoon. Joel Williamson noted that implementations were much more likely to look up distributions than to do heavy number theory. If a programmer got too tricky, “it won’t matter if you’re right, the compiler won’t do it.” Mary Zosel and Rob Schreiber claimed that explicit interface questions raise the issue of correctness again. Guy Steele explained this as a consistency issue - the compiler may ignore directives “if it does so consistently.” For example,

```
REAL a(100)
!HPF$ DISTRIBUTE a(CYCLIC)
CALL sub( a(40:50:2) )

In sub:
REAL x(6)
! error - this means the 6 elements are cyclic
!HPF$ DISTRIBUTE *(CYCLIC) :: x
! correct - x aligned with 100-element template
!HPF$ DISTRIBUTE *(CYCLIC), INHERIT :: x
```

Asked “What’s the point?” Guy replied that such declarations give additional information to the compiler, by giving the user the ability to make falsifiable statements. After a short digression into the philosophy of Popper, the practical difference emerged. INHERIT gives less information about the template, and therefore may require dope tensors, etc. that aren’t needed without it. Faced with objections that these features were not clear to the reader, Rich Shapiro gave an impassioned speech, ending with the statement “We should sacrifice clarity of notation for performance.”

John Merlin raised the objection that if a 1-D array can have a 2-D distribution through INHERIT, then compilation faces a combinatorial explosion in possible cases. Rob Schreiber responded that John’s argument implied a particular implementation, which might then become untenable but other implementations were possible. A long argument ensued over what could or could not be expressed or compiled under the proposed system. Andy Meltzer summarized by saying that this shows there are limits to any system, and we have to accept limits. Ken Kennedy suggested thinking of this as compiler information that allows nice cloning, and generally favored the idea.

Finally, the entire proposal (as outlined above) was moved and seconded. In addition, the proposal removed the syntax “ALIGN WITH *” as redundant (but kept “ALIGN WITH *T”). The vote to accept this proposal was 17 yes, 2 no, 3 abstain. Richard Swift immediately proposed to eliminate the “*” and “ONTO *” options in DISTRIBUTE from the HPF subset. INHERIT would stay in, to allow the important case of “DISTRIBUTE * ONTO *” (meaning “don’t move the data”). Rich Shapiro amended this further to eliminate the “(CYCLIC)” and “ONTO P” cases (i.e. the forced remapping cases). Piyush Mehrotra pointed out that Rich still needed to throw out INHERIT to avoid problems, and Rich took this as a friendly amendment. A vote was taken on Rich’s amendment, finding 6 in favor, 11 against, and 5 abstaining. Coming back to the Swift amendment, a vote was taken, finding 7 yes, 8 no, 7 abstain. Vince Schuster moved to strike INHERIT from the subset, but was dissuaded by Ken Kennedy.

Guy Steele next presented a new proposal on what could be said about an actual template.

```
standard example:
!HPF$ DISTRIBUTE a(BLOCK,CYCLIC)
CALL sub(a(..,..))
```

```
SUBROUTINE sub (x)
!HPF$ DISTRIBUTE x *(..,..)
```

The goals in this proposal were

- Allow: passing same locations on all processors
- Allow: anything in undistributed dimensions
- Not allow: funny strides that happen to work
- Not allow: only locations on some processors
- Not allow: eliminating parts of processor array

This led to the following set of rules regarding what can be passed:

Subscript on actual	Actual distribution	Natural dummy distribution
any	none	OK
scalar	*	disappears
l:h:s	*	*
l:h:s	block(n)	block(n/s)
l:h:s	cyclic(n)	cyclic(n/s)

The last two lines are valid provided that l is less than s away from the lower bound of the array and that s divides n. After a question from Chuck Koelbel, Guy added the restriction that the array must exactly cover the template. For any other cases, there are no language-level guarantees (number theory may work out anyway, but we will ignore that).

The vote to adopt this proposal was 12 yes, 0 no, and 10 abstain.

The group then took a coffee break.

EXTRINSIC

After the break, Maureen Hoffert and Mary Zosel asked about a feature of the EXTRINSIC chapter. That chapter included two directives - LOCAL and EXTRINSIC - where it seemed only one was needed. Guy Steele promised to check with Marc Snir (who was not present) and modify the draft accordingly. He further promised to add a section on calling the outside world from HPF to further clarify the role of EXTRINSIC.

Administrative Matters

Ken Kennedy then began a discussion of administrative matters, starting with plans for an HPF follow-on effort. It was well-known that many issues in the Journal of Development needed more work; in addition, many topics had been suggested during HPFF meetings but not pursued for lack of time. There was unanimous agreement that a second effort (dubbed HPF 2, for lack of a better name) should be organized. The remaining issues were

- When should the effort start?
- Who would be the new Executive Director?
- Who would fund the effort?

Chuck Koelbel announced that he would like to step down from the Executive Director post to devote more time to other research interests. There were no immediate volunteers to fill his shoes, but some expressed interest if they could see what they were getting into. Chuck promised to write up a job description.

The funding question arose due to problems with HPFF's past strategy of starting the technical meetings before getting a firm monetary commitment from government agencies or corporations. This mistake will not be repeated; in the meantime, suggestions for covering the funds shortfall were solicited.

The timing of the follow-on depended on two factors - the need to get experience (arguing for a late start), and the need to get consistent spelling on additions (arguing for an early start). Guy Steele suggested "I think Rice should host a meeting for 500 people in January 1994." David Loveman noted that starting with positions was very helpful the first time, and suggested that HPFF2 should do it again. Alok Choudhary made a similar statement regarding the Supercomputing '91 workshop. After some discussion of various strategies, the group agreed to the following schedule:

- Workshop at Supercomputing '93 (to combine implementors' and users' experience)
- A Monday tutorial on HPF (only the current language) at Supercomputing '93 (organizers to be picked in March)
- A 1.5 day meeting in January 1994 to talk about positions, similar to the January 1992 meeting at Rice
- Main effort in calendar year 1994

David Loveman had to leave near the end of the discussion, giving the group "proxy to vote or volunteer" him. What a trusting soul.

The administrative issues ended with a discussion of designing HPF t-shirts. Andy Meltzer volunteered to coordinate t-shirt collections. Several designs were suggested verbally, but no clear favorite emerged.

Low Performance Fortran

Andy Meltzer closed the meeting with his traditional LPF presentation. He prefaced the remarks by saying that LPF nearly didn't happen because "It was real hard to beat HPF this time."

Some committee members proposed changes:

- Have VIEW but no REVIEW, which is only allowed in an academic environment
  - Grades a processor arrangement as good, fair, poor, miserable, or LPF quality
  - No program above "LPF quality" is conforming
  - Suggestion rejected
- Have IPARITY, IANY, and IALL, what about ICLAUDIUS?
  - Suggestion rejected
- Note that due to the nature of LPF, no good suggestion may be accepted.

Initially, LPF intended to go with HPF semantics across subroutine boundaries.

- This was felt to be confusing enough so that no programmer could ever do useful work.
- It was then noticed that a few possibilities were missing
- Use "->" to take distribution (or processors arrangement) from somewhere (but where is compiler-dependent)
- Use "<-" to put distribution somewhere (undefined)
- Use "?"to distribute any variable in the program, compiler's choice
- These options can be combined, for example

```
DISTRIBUTE ? ->() <-ONTO->
```

(This means pick a variable and move from whatever processor arrangement it is on to a new distribution on a new processor arrangement.)

- LPF did, however, get many good syntactic ideas from HPF. In particular, LPF also likes “*”

HPF considered, but narrowly rejected, some alternative syntax: WITH, ONTO, OF, replaced by “->”. LPF like it, but it doesn’t go far enough.

WITH, ONTO, OF become “.->.”
ALIGN becomes “.”
DO becomes “./.”
END DO becomes “.(.”
IF becomes “.\.”
THEN becomes “.).”
ELSE becomes “.’.”
END IF becomes “.+.”
DIMENSION becomes “.!.”
“(“ becomes “/”
“)” becomes “**”

So, an example program fragment

```
SUBROUTINE HPF( X, B )
DIMENSION X(:), B(:)
!HPF$ ALIGN X WITH *
DO I = 1, 100
   IF (X(I*I) .NE. X(I)*B(X(I))) THEN
      A(I) = SUM( B(I:N) )
   END IF
END DO
END
```

becomes

```
SUBROUTINE LPF( X, B )
.!.X/:**,B/:**
!LPF$ .X.->.*
./.I=1,100
   .\.X/I*I**.NE.X/I***B/X/I******.).
      A/I**=SUM/B/I:N****
   .+.
.(.
END
```

Isn’t that clearer?

Since LPF got no comments (which is unfortunate since I didn’t get to shoot anyone), here are some hypothetical typical comments.

- Comment: I sent in my $1000, but I didn’t get a copy of the draft.
  Reply: Tough cookies
- Comment: My ten year old daughter died slowly and horribly. Her last request was that you capitalize the first word in each sentence.
  Reply: Tough cookies

- Comment (From the US trade delegation): You must change the semantics so that the FOREIGN routines which have the import restrictions don't include Iran, Iraq, or China, but don't let in any French variables.
  Reply: Tough Chablis
- Comment (from God): Looks great! Best thing I ever read. It'll replace the bible if you make me co-author.
  Reply: Fat chance, pal

Publication of LPF, as noted above, is in negotiation.
Finally, a point-by-point comparison of HPF and LPF

- Speed
  HPF: Speed of light in a vacuum (yeah, right)
  LPF: Speed of sound in a vacuum
- Performance
  HPF: Never slower than any one of its processors
  LPF: Never faster than any one of its processors
- Clarity
  HPF: When shown to a committee chair, he/she should be able to decipher it
  LPF: When shown to a committee chair, he/she will laugh
- Compile speed
  HPF: Will terminate sometime on all programs
  LPF: Who cares?
- Usefulness
  HPF: Committee members can buy HPF t-shirt
  LPF: Committee members can buy any t-shirt they want