Minutes of the

Fortran Experts Group

Meeting at Vienna, 14-17 June 1982

X3J3/148

This electronic ecord of the meeting was derived from the paper version using OCR in January 2010. A few minor typographical errors have been corrected. The accompanying documents (W-1 to W-33) are not included.

CONTENTS

Page

Agenda 1
Attendance List 5
Synopsis of the Meeting 6
- Administrative: Opening Business 6
- Overview of S6 7
- Control Structures 14
- Data Structures 16
- Array Processing 17
- Program Form 19
- input/output 19
- Compile-Time Facilities 20
- Language Architecture 21
- Character Data Type 22
- Bit Data Type 22
- Precision Data Type 23
- Procedures and Functions 24
- Internal Procedures and Interprocedure Data Sharing 25
- Dynamic Storage and Recursive Procedures 26
- Other Features 26
- Administrative: Closing Business 36

Documents 37
- (Preliminary) List of Delegates 37
- (Provisional) Agenda 38
- Munchhausen: Naming of Control Structures 41
- UK National Activity Report 44
- Muxworthy: Comments on X3J3 Proposals 45
- FORTEC Forum 51
- Netherlands National Activity Report 69
- Ampt: General Remarks Concerning S6.81 71
- Ampt: Event Handling, Scoping, and interfacing 80
- Canadian National Activity Report 85
- Canadian Comments on S6.81 86
- Reid: Matters of Importance for Numerical Subroutine Libraries 92
- Reid: Fortran 8X Array Features on, the Parallel DO 99
- Delves, et al: Core Plus Modules in Fortran 8X 105
- Selberherr: Comments on Array Processing 109
- Kulisch, et al: Fortran for Contemporary Numerical Computation 111
- Clarke: F77-F8X Incompatibilities 131
- Clarke: Integer Range Values 132
- Clarke: What is Missing from S6? 133
- Clarke: UK Report -Appendix A 134
- Metcalf: Bit Data Type in Core 136
- German Comments on S6.79 138
- Schonfelder: User Written Application Modules 144
- Delves: User Defined Operators 150
- Wilson: Array Examples 153
- Sobiesiak: Event Handling in Fortran 8X 156
- Sobiesiak: A Proposal for Event Handling 160
- Clarke: Methods for Defining the "Macro" Attribute 170
- ECMA: Questionnaire on the Present Usage of Fortran 171
- Document List (Incomplete) 179
- French National Activity Report 181
- Weisz: Comments on Compile-Time Facilities 182
- (Final) Attendance List with Addresses 184

AGENDA

Vienna, Austria

June 14-17, 1982

Monday, June 14, 9:30 AM

1. Opening of the meeting
(Jeanne Martin, Convener)

2. Welcome of the delegates

3. Roll call of delegates

4. Adoption of agenda

5. National activity reports

6. Procedural matters

7. Brief summary of proposals passed at the 82nd meeting of X3J3
(Loren Meissner, Secretary, X3J3)

B. Objectives of Review of S6.81
(Joanne Brixius, X3J3, U.S.A.)

9 Election of chair

- Monday, June 14, 2: 30 PM

10. General Overview of Proposal Document X3J3/S6.81

G. Schmitt, Austria
J. Humar, Canada
C. Mas, France
K.H. Rotthauser, Germany
J. Kalman, Hungary
C.G.F. Ampt, Netherlands
I. Dahlstrand, Sweden
D.T. Muxworthy, United Kingdom

11. Feature Sessions -Proposal Document X3J3/S6.81

- Tuesday, June 15, 9:00 AM

Control Structures	Moderator: J. Wagener, X3J3
Data Structures	Moderator: R. Oldehoeft, X3J3
Array Processing	Moderator: A. Wilson, X3J3

Fortran 8X Array Features and the Parallel DO
(J.K. Reid, U.K.)

Comments on the Array Processing Proposals
(S. Selberherr, Austria)

-Tuesday, June 15, 2:30 PM

Program Form	Moderator: J. Brixius, X3J3
Input/Output	Moderator: J. Brixius, X3J3
Compile-Time Facilities	Moderator: J. Martin, X3J3

Methods for defining the Macro Attribute
(P.A. Clarke, U.K.)

Comments on Macros
(W. Weisz, Austria)

Language Architecture	Moderator: J. Wagener, X3J3

- Wednesday, June 16, 9:00 AM

Character Data Type	Moderator: J. Wagener, X3J3
Bit Data Type	Moderator: J. Wagener, X3J3
Precision Data Type	Moderator: L. Meissner, X3J3

Procedures and Functions	Moderator: J. Martin, X3J3
Dynamic Storage	Moderator: L. Meissner, X3J3
Recursive Procedures	Moderator: L. Meissner, X3J3
Internal Procedures	Moderator: L. Meissner, X3J3
Interprocedure Data	Moderator: L. Meissner, X3J3
Sharing Name Management	Moderator: L. Meissner, X3J3

Interfacing and Scoping
(Kees Ampt, Netherlands)

ECMA Fortran Questionnaire
(C. Mas, France)

- Thursday, June 17, 9:00 AM

12. Other Features

Matters of importance for numerical subroutine libraries
(J.K. Reid, U.K.)

Abstract Data Types
(J.L. Schonfelder, U.K.)

Fortran for Contemporary Numerical Computation
(U. Kulisch, Germany)
(C. Ullrich, Germany)

Event Handling in Fortran 8X
(W. Kneis and H. Sobiesiak, EWICS)

- Thursday, June 17, 2:30 PM

ECMA Fortran Questionnaire
(C. Mas, France)

Overview of Recommendations Made
(J. Humar, Canada)

Compatibility of Object Code, Fortran 77 and 8X
(G.E. Millard, U.K.)

Syntactic Lists
(P.A. Clarke, U.K.)

13. Summary of X3J3 Members' Comments on S6 Review
(Members of X3J3)

14. Requirements concerning a subsequent meeting

15. Closing Remarks

16. Adjournment

Other Events

- Tuesday, June 15, 1:00 PM

Meeting of X3J3 Task Group 2

- Wednesday, June 16 - Thursday. June 17

Meeting of EWICS TC1

- Friday, June 18

Fortran Forum

Meeting of ISO/TC97/SC5/WG1

ATTENDANCE LIST

1 Jeanne Adams, United States
2 Cornelius G F Ampt, Netherlands and EWICS
3 Joanne Brixius, United States
4 P Alan Clarke, United Kingdom
5 A M Decroix, France
6 Jagmohan I Humar, Canada
7 Kalman Janko, Hungary
8 Wilfried Kneis, Germany and EWICS
9 Werner Koblitz, Austria and EWICS
10 U Kulisch, Germany
11 Jeanne T Martin, United States
12 Christian J Mas, France and ECMA
13 Loren P Meissner, United States
14 Michael Metcalf, Switzerland and ECMA
15 G E Millard, United Kingdom
16 Willard L Miranker, United States
17 Meinholf Munchhausen, Germany and ECMA
18 David T Muxworthy, United Kingdom
19 R R Oldehoeft, United States
20 Klaus Plasser, Austria
21 A Pollicini, Italy and CEC
22 John K Reid, United Kingdom
23 K H Rotthauser, Germany and ECMA
24 Gerhard J Schmitt, Austria
25 J Lawrie Schonfelder, United Kingdom [*corrected from 'Austria '*]
26 S Selberherr, Austria
27 M K Shen, Germany
28 Jan A M Snoek, Netherlands and EWICS
29 Hieronymus Sobiesiak, Germany and EWICS
30 Ch. Ueberhuber, Austria
31 Christian Ullrich, Germany
32 F Vapne, France
33 Jerrold L Wagener, United States
34 Willy Weisz, Austria
35 Gunter Wiesner, Germany and EWICS
36 Alan Wilson, United Kingdom and United States
37 Juergen Wolff von Gudenberg, Germany
38 L G J ter Haar, Netherlands

SYNOPSIS OF THE MEETING

1. The meeting was opened at 9:30 AM by the Convener, Jeanne Martin.

2. The delegates were welcomed by Prof. Hans Stetter of the Technische Universitat Wien and Frederich Long representing the Austrian Standards Organization.

3. Delegates were present from the following countries: Austria, Canada, France, Germany, Hungary, Italy, Netherlands, Switzerland, United Kingdom, and United States. ECMA, EWICS, and CEC were also represented. The absence of Ingmar Dahlstrand of Sweden, due to illness, was noted with regret.

4. The agenda (document W-2) was amended to include a presentation on "Fortran for Contemporary Numerical Computation" by Kulisch and Ullrich on Thursday morning and adopted. (The actual agenda appears on page 1.)

5. National activity reports were presented by delegates from each of the countries, and from ECMA and EWICS. See documents W-4, W-7, W-10, W-31.

6. It is customary to take straw votes to record opinions. Everyone may vote. Straw votes are recorded as (yes-votes, no-votes, undecided).

Jerry Wagener, Chairman of FORTEC, the Fortran Technical Committee of the Special Interest Group on Programming Languages (SIGPLAN) of the Association for Computing Machinery, announced the first issue of FORTEC FORUM (See W-6). This quarterly publication will deal with all aspects of the Fortran language, including development, implementation, standardization, and use. Loren Meissner is the editor.

Loren requested that material in the form of book reviews, articles, notices of products, etc., be sent to him for publication. Loren formerly edited the Fortran Newsletter, "Forword", which is being discontinued. The last issue is Volume 8, Number 2.

A membership in FORTEC is $6 for ACM members and $16 for non-ACM members.

Kees Ampt objected to the affiliation with ACM which is not an international organization and does not recognize or allow discounts to other national organizations.

Loren would be willing to send the publication to someone in London or Amsterdam to be distributed in Europe. It would only require clerical assistance to maintain the mailing list.

Jerry Wagener requested that letters be sent to ACM headquarters to see if bridges could not be built to other not national computing organizations. Other committees within ACM would benefit as well.

John Reid felt that $16 was very reasonable.

Kees was not objecting to the amount involved, but to the principle which tended to put road blocks in the way of international communication.

7. Loren Meissner summarized the proposals passed at the 82nd X3J3 meeting. (See W-6, page 15.)

8. Joanne Brixius addressed the objectives of this meeting's review of the S6 document which is a collection of all the proposals passed so far for Fortran 8X.

9. David Muxworthy was elected to Chair the meeting; Jeanne Martin was elected Vice-Chair.

10. General Overview of S6.81

Schmitt: S6 has gotten very large, very fast. X3J3 is adding too many new things to the language. There is a need to cut out something, but on the other hand, some facilities need additional capabilities. We have specific comments to make later about the array features, compile-time features, and the requirements for numeric features.

Humar: S6 indicates the directions in which Fortran is going: compact, simple, efficient, structured, diverse, compatible. The core plus modules model provides the best compromise. The core criteria are well defined, but I am concerned about the conformance to these criteria. There seems to be a scramble to get things in. One example of this laxity of conformance is the entity-oriented declaration. A more conservative approach should be taken to new features. I am reminded of fashion designers promoting fads to create jobs.

Ampt: S6 shows real improvement over Fortran 77. I have comments on each chapter. These are detailed in document W-8. [Editor's Note: The following is a brief summary of Kees' recommendations.]

a. Control Structures
    1. Prohibit the use of GO TO's in blocks unless the destination is also in the block.
    2. Allow EXIT in all blocks (don't restrict it to DO-blocks). Also allow EXITs from procedures.

3. Eliminate the "fashionable" CASE construct. ELSE IFs can be used instead. It is just the difference between a chain and a tree.

b. Data Structures

1. Find some way to explain data structures other than an undefined meta language.

2. Clarify the intent with regard to the nesting of FORMs and the use of a FORM as a type.

c. Array Processing

1. Allow shape to be a dynamic property. Conformity should depend on size only.

2. Eliminate the WHERE statement. Replace it with a keyword phrase WHERE= in the PACK and UNPACK statements.

3. Problems: Defaults for EOSHIFT when applied to character data. Definition status when the array stored into has a vector subscript (multiple stores into the same element are possible)

4. Eliminate some of the intrinsic functions.

d. Procedures and Functions

1. Require that the interface information appear in a SHARED DATA block and be inherited, but permit keyword names (dummy argument names) to be overridden by redefinition inside the inheriting procedure.

2. Extend INTENT to apply to structures.

e. Program Form

1. Specify the maximum number of characters in a statement.

2. Clarify the use of comments on continued lines.

f. Precision Data Type

1. Specify precision attributes for operators not operands.

2. Allow conversion across subprogram boundaries (as is now allowed across equal signs)

3. Introduce a new statement CONVERSION NONE to prevent both (conversion across equal signs and subprogram boundaries).

g. Dynamic Storage

1. Clarify the meaning of SAVE for recursive procedures.

2. Resolve present difficulties so that RECURSIVE could become the default in the future.

h. Bit Data Type

Eliminate

i. Internal Procedures and Interprocedure Data Sharing

1. Internal Procedures are too complicated. Simplify.

2. The SHARED DATA block is a good concept. Extend it.

3. INHERIT forms and structures.

j. Input/Output

1. Put a good part of the I/O features into an I/O module. There is too much for all of it to go in core.

2. Eliminate one of: file or unit.

k. Compile-Time Facilities

1. They should not be a part of the language, but part of another level such as the operating system. Since this level is not currently standardized, portability may require that these facilities be standardized with the language.

2. I/0 is required at this level.

l. Character Data Type

Since CHARACTER is the basic type, adopt more operators that apply to the character data type, rather than many intrinsics.

m. Language Architecture

Add an additional criterion: For every new feature, remove two old ones.

Muxworthy: It is gratifying to see that some of the suggestions made at previous Fortran Experts Meetings have been accepted by X3J3. Although the press predicts that ADA will become the predominant language, that is not the view of users. It is important that Fortran 8X be built upon the experience gained through the use of Fortran 77. For 10 years, the UK has pushed for more regularity and less permissiveness. See W-5 for comments on each chapter of S6.

Summary of recommendations:

a. Control Structures

Clarify what happens when the case expression matches no block selector. The fall-through default is not acceptable.

b. Array Processing

Make sure the array features put into core do not inhibit implementation on small processors.

c. Procedures and Functions

Specify what happens when the OPTIONAL and INTENT declarations do not agree with actual interfaces.

d. Program Form

1. Specify a minimum and maximum line length.
2. Reconsider allowing the underscore to be treated as alphanumeric and the use of 31 character external names. This might result in the creation of a Fortran environment that is in conflict with current usage.

e. Precision Data Type

Reconsider the implications of the present proposals with regard to efficiency particularly where software implementations are required because the native data type is inadequate.

f. Input/Output

Once decisions are made on event handling, reconsider I/O errors and how the subsequent actions taken can be made portable.

g. Compile-time Facilities

Adopt a comprehensive macro facility.

h. Character Data Type

To avoid redundancy, adopt a STRING facility and deprecate the CHARACTER facility.

i. Architecture

Do not give up on the core plus modules approach.

j. Unresolved Items
Work on the compatibility module, which is of particular concern to the UK, cannot proceed until at least tentative

decisions are made with regard to event handling. The form of event handling that X3J3 adopts will give guidance for further work in this area.

Discussion:

Shen: Do not reduce the size of core. Event handling will permit the use of Fortran for real-time applications and primitive debugging; compile-time and interpretive facilities are needed; string manipulation facilities (as in SNOBOL) are needed; character data type facilities are needed for text processing and report writing; the CASE statement permits better optimization. These features will make Fortran a general purpose language rather than a straight scientific language.

Metcalf: The bit data type is needed.

Ampt: The only things that are really needed are the assignment statement and a conditional transfer of control. (I/O is really an assignment.) So there is always a duplication of functionality. We need to choose carefully from a list of nice features. LOGICAL and a string capability can replace BIT.

Wagener: We need a much richer string facility. Aside from intrinsic functions, what else is needed, precisely?

Schonfelder: I object to the implications of assignment across subprogram boundaries. Automatic coercion is totally wrong in numeric terms.

Ampt: The assignment statement specifies coercion to the type of the left hand side. Shouldn't passing an argument have the same implications (argument converted to the type of the dummy). Numerical people are not the only ones who use Fortran. They should be forced to specify the exact type.

Schonfelder: But it is now the way numerical analysts want it.

Meissner: Some one needs to determine how bits are going to be used, before any decision can be made on whether they are necessary or redundant -perhaps a comprehensive survey.

Humar: Will core be smaller than Fortran 77?

Adams: What will be in core is not a consideration in our present task of merging S6 and Fortran 77. We hope that the merger will find any incompatibilities. Loren took a different approach when he wrote "Weathervane". He started from scratch to define his view of "Futuretran". Loren does not think that merging S6 and Fortran 77 will produce a draft of the standard, and he may be right, but it will help us see where problems exist. Later when we know what the entire language entails, we can determine the core content.

Meissner: We would like for an elite group to write the first draft of the standard, but that is not possible, and we must proceed.

Clarke: Why is it not possible?

Adams: We cannot get someone funded to work totally on the 8X standard. We must use the whole committee.

Munchhausen: Could you find a better name than "Obsolete Features Module"?

Weisz: COBOL initially used "Transitional". They are now using "Obsolete Features".

Adams: Of the 5000 letters of comment on the draft COBOL standard, 99% were duplicates. Fortran will not have that problem. There has been pressure to publish S6, but there are problems.

Metcalf: The alternate RETURN and BLOCK DATA and DATA are in the obsolete module. What replaces them?

Martin: Entity-oriented declarations in SHARED DATA blocks take care of DATA and BLOCK DATA. There is nothing in S6 to replace alternate RETURN. It is a bit unstructured.

Metcalf: Unstructured or not, users want it.
Ans: Event handling may take away the need for alternate return.

Metcalf: How is Fortran distinct from other languages if all these features are added to it? Are languages going to converge?

Adams: Fortran serves the scientific community. We can take some of the features from other languages, but not all. Fortran must be implementable on small computers.

Martin: Fortran was not the greatest possible language when it was developed, but now there are many libraries and programs that depend on it. We need to improve the language, but in such a way that existing programs won't suddenly be unusable, but can gradually be upgraded and improved by making use of the new language elements and eliminating the older unreliable, unreadable elements.

Ampt: The priority list for Fortran used to have "efficiency" at the top. Now it seems to have "compatibility with old Fortran". There were once a lot of codes in assembly language too. Can't we move away from it? Some complain that the only resemblance between old Fortran and all the new features is the name. Is that bad?

Meissner: People's priorities can be determined not by what they say, but how they vote.

Shen: Even natural language could benefit from a subset of SNOBOL.

Snoek: Jerry wanted more specific data on characters. Loren said we should ask the users what they want on BIT. Users of character type are editors, etc., and they do not use Fortran because it does

not have what they need. We need to ask potential character users what they want. The question is not whether BIT data type is necessary, but whether it is good enough. Maybe a general data or string type would be better than character or bit. Core should be small.

Weisz: Association of arguments is important. The standard should say they must agree and if not there is automatic coercion of actual to dummy argument type. There are two things that SAVE could mean in a recursive environment, both of which are needed. There is a need to manipulate BITs in a standard way.

Schonfelder: Most people want to manipulate strings and find the length after assignment, so they need a dynamic string capability. You cannot have all the things that are wanted and a small core. The only way to go is to generalize. Complex could be a derived data type - not a part of the primitive language. It could be implemented as a special library that can be called from the small core.

Adams: We had the same discussion 10 years ago - whether BIT was a basic data type and character could be derived from bit.

Meissner: Derived complex would require operations on structures, and we do not plan that.

Reid: I disagree (with Kees) that the proposal for internal procedures is complicated. It is well-structured and it solves the problems of interfacing with libraries very nicely.

Humar: The proposal goes much further than it should. Two restrictions have recently been removed (allowing internal procedures to be nested, and passing their names as arguments).

Reid: Both of these are needed. The hiding of names is important.

Ampt: An internal procedure is some code that is to be used in more than one place in a program unit. We do not need to add a lot of facilities to accomplish this. A limited dependent compilation is all that is required.

11. Chapters of S6 were discussed in the order indicated in the Agenda. For each chapter, the moderator made a few remarks to introduce the topic. This was followed by discussion. Due to the press of time, there was not as much discussion of the topics that appeared toward the end of the review.

Control Structures

Discussion:

Clarke: The construct name is optional in several places. Why is it required in only one?

Ans: If it appears on the first statement of the construct, it must appear on the last. Others are optional.

Munchhausen: The compiler should check the construct names for proper nesting.

Schmitt: The END name may be ambiguous.

Ampt: Because other languages have it is not a sufficient reason for including a WHILE construct. If it were a real WHILE construct, such that the exit was taken whenever the condition changed to false, that would be different. It would be a kind of event handling. It might be a surprise to the programmer. Re Reid's parallel DO - preventing side effects is good, but I don't like the mechanism. The solution lies in another direction. The CASE construct is not needed. With the DEFAULT case, anything works.

Weisz: The real DO WHILE is dangerous. The condition could change inside a called procedure. CASE DEFAULT is very important to trap run-time errors. It is a kind of built-in event handling.

Ampt: Scoping rules are important and pertinent to this discussion. I will go into that tomorrow. You shouldn't use CASE DEFAULT for event handling. That would corrupt the idea of the functionality of core.

Reid: I am strongly against the remote code block. Internal procedures take care of this need. There is no merit in another syntax for the same functionality.

Humar: WHILE is in heavy use; we should standardize it. We favor the CASE construct, but the logical type for the selector should not be there. We prefer GO TO to multi-level exits and cycles.

Schonfelder: CASE may be redundant, but if you need a multi-way branch, it is much clearer with CASE. The same is true for EXIT and CYCLE. There is a strong argument for readability. Allowing multi-level returns from subroutines would be very dangerous.

Shen: I prefer DO WHILE to the alternative IF, GO TO. DEFAULT saves ink. Can case have array-valued expressions?

Ans: No.

Snoek: The block-DO with the EXIT statement is more general than WHILE and UNTIL. You have all the functionality you need.

Clarke: There is an irregularity. We have all the named constructs, except the simplest - a sequence of statements. We need to provide that

also even if there is no use for the name. I will make a proposal later that depends on such a facility.

Wagener: How is that different from a remote block?
Ans: It is the same.

Ampt: There is need for a block with one entry and one exit.

Pollicini: The exit from a subroutine is fundamentally different from the exit from a loop.

ter Haar: Internal procedures can be used to simulate the remote block, but the remote block more closely resembles current practice. Conversion to the new language would be simpler if the default were INHERIT ALL.

Reid: In simple cases, INHERIT ALL is nice, but in others the effects are dangerous.
Ans: In other languages internal procedures are totally different from this.

Schonfelder: This is the ALGOL vs. Fortran controversy. With the ALGOL structure, maintenance is difficult. You can get a complete mess. We should preserve Fortran's strengths.
Ans: That is a good point. I have no objections, I was merely stressing compatibility.

Snoek: A macro facility provides the functionality of a remote block.

Weisz: The remote block is a bad construct in Cobol; the Fortran subroutine is better. It is a bad mix, when a block of code can be gotten to by a jump and return and also sequentially.

Reid: Macros could waste storage.
Ans: Yes.

Pollicini: There is an incompatibility with Fortran 77 if a real variable is not allowed in the control clause of a block-DO. The standard should state what happens if the CASE DEFAULT is not present. The multi-level exit using the block name is much better than the use of a number to indicate the level of the block.

Wagener: CASE DEFAULT preserves independence. It can be the first block, the last, or in the middle. It is order independent.

Pollicini: I would prefer the construct name to appear on the left.

Martin: With the new source form, it is not clear there is a left (Statements may start in position 1).

Munchhausen: I would like to see EXIT allowed in all control structures.

Humar: The construct name should have some functionality. It may be hard to see since multiple statements per line are permitted.

Snoek: Currently the value is not allowed not to match (in block -CASE). An error should be raised instead. The alternate RETURN creates havoc in a hierarchical situation. It should be left to multi-tasking.

Schmitt: Instead of solving the DEFAULT CASE, it is more necessary to have a conformity module.

Ampt: I would like to see the new block constructs defined in terms of strict, closed blocks, i.e., one entrance, one exit.

Meissner: That is incompatible with Fortran 77.

Ampt: We have the new DO.

Meissner: What about the IF-THEN-ELSE?

Ampt: We could change the syntax of IF to get a new form.

Meissner: Then we could never use STOP.

Straw Vote: Should block constructs be defined in terms of strict blocks? (4-14-10)

Straw Vote: Include a DO WHILE construct. (9-14-4)

Straw Vote: Allow multi-level EXITS. (17-7-2)

Straw Vote: Allow multi-level CYCLES. (11-8-8)

Straw Vote: Take the block-CASE construct out of the language. (5-15-5)

Straw Vote: Retain logical as a type for the block selector. (4-13-9)

Data Structures

Discussion:

Schonfelder: Can you equate two structures with different form?
Ans: No.

Reid: We must use period as the qualification symbol and in the traditional way (A.EQ.B). This means X3J3 must prohibit the use of keywords (i.e., EQ) for form and field names.
Ans: Fortran does not restrict keywords.

Snoek: A dimensioned structure with a dimensioned field would make it possible to construct an array in the normal way rather than Fortran's canonical order (columnwise). If there is any ambiguity with the use of period to indicate qualification, an editor can fix it up. It is nonsense not to use period.

Weisz: It can be handled, but you may get 5000 letters of complaint.

Ampt: Structures should be extended to abstract data types.

Munchhausen: I would prefer a different syntax - like the character data type with *.

Schonfelder: We are so close to extended data types we should go all the way and have infix operators and secondaries defined in terms of primaries.

Straw Vote: Use period as the qualification symbol. (22-0-5)

Straw Vote: Require full qualification of names (no aliases). (3-15-10)

Straw Vote: Should we allow dimensioned fields within structures and dimensioned structures? (17-0-12)

Straw Vote: Should the use of qualification and dimensioning be extended to structure-valued and array-valued functions? (e.g., STRVAL(A,B,C).F4 and ARRVAL(A,B,C)(I,*)) (16-1-12)

Wagener: I like the suggestion of abstract data types but I am uncomfortable about voting on it now. In order to define operations on structures, more would be required than we have so far.

Schonfelder: They would make the language simpler - a small core could d be created. Application modules could be defined in terms of these types.

Array Processing

References: W-25, Wilson, Array Examples
W-13, Reid, Fortran 8X Array Features and the Parallel Do
W-15, Selberherr, Comments on Array Processing

Discussion:

Ampt: The intrinsic ABSSUM can be created from the ABS function.
Ans : Yes.

Selberherr: It may be necessary for code transparency, but there is no need for DOTPRODUCT. It may be used often, but it is something you can get along without.

Schonfelder: General intrinsics that don't depend on the nature of the argument should be there. Expressions that arise from assumptions about the objects in the arrays should not.

Humar: Whole array assignments should go in core. The others should go in an extension module. Other than the IDENTIFY statement, is there a way to reference non-contiguous arrays?

Ans: Yes, section selection for rectangular shapes. Skewed shapes (such as a diamond) require the IDENTIFY mechanism.

Selberherr: For compatibility it is necessary that mini-computers be able to emulate what is in the standard.

Snoek: For arrays we need definitions and operations. These are the same characteristics that are required for abstract data types. Why don't we do everything the same way rather than specializing each case? There are more intrinsics for arrays than for characters because X3J3 knows more about arrays than characters. Some of the array processing features should be in a language extension module, not all in core.

Selberherr: I am opposed to a data structure approach because of the lack of efficiency. A 10% loss is an immense loss. The fact that it is more elegant isn't relevant. Programs need to be developed fast and they must run fast. If another language does it better, I will switch to that language.

Ampt: I disagree. Maintenance time is the crucial element - not development time or execution time. Besides the language will run on faster and better hardware in the future.

Snoek: A general approach is not necessarily slower. What is needed is a programming language that is easy to write, learn, etc., with general concepts that everyone can understand.

Selberherr: I agree in principle. Programmers will still be thinking in the old way. You need to make the upgrade from one standard to the next easy.

ter Haar: Array processing features are needed for software engineering and for run-time efficiency. This proposal mixes essentials and niceties for special applications and processors. It should be split into distinct parts.

Weisz: For complex functions such as these, it is easier for the compiler to produce efficient code than for the user. Implementation techniques are getting better.

Munchhausen: It will be nice to be able to write loop free programs. Why is the WHERE construct so limited? It can't be nested; can't contain I/O, etc. I don't like the IDENTIFY statement.

Meissner: WHERE is dangerous. I am opposed to extensions to the WHERE construct. It should be replaced by a merge operation.

Snoek: I am against the enormous number of array intrinsics when all we need is a syntax to define operations on data types.

Program Form

Discussion:

Snoek: Students are encouraged not to use multiple statements per line. I am disappointed that it is allowed in Fortran.

Ampt: The language should not solve the problem of separating lines that come in from a terminal.

Metcalf: Multiple statements per line is a retrograde development.

Weisz: I am also against it. Such programs are horrible to debug. Portability suffers if the line limit is less than 72 characters; this compounds the problem. Comments should be visible - single line or end of line.

Schonfelder: I disagree. You can only see a limited number of lines on a terminal screen. The problems can be solved by adding restrictions, such as not allowing a second statement to be continued. Embedded comments are unreadable.

Metcalf: I recommend that the standard set minimum sizes and limits.

Ampt: Minimums often become maximums, so we must be careful. We don't need comments between continuation lines.

Meissner: Some restrictions might be appropriate, such as not allowing statements that terminate blocks inside lines.

Humar: Restrictions make it hard to learn a language. One statement per line should be the restriction.

Straw Vote: Allow multiple statements per line. (5-19-3)

Straw Vote: Allow blanks to be significant. (12-8-5)

Straw Vote: Allow names to be split across lines. (11-11-5)

Input/Output

Discussion:

Ampt: Do we really need both file and unit?

Meissner: I would like to quit talking about units, unfortunately we do not have consistent operating systems with consistent file naming conventions. We have to tie the file to some other specification. Unit numbers work for the time being.

Weisz: I don't understand what happens when you open a file AS IS. What is the position? It seems the only way you can assume a position is if you possess the file.

Meissner: AS IS means nothing.

Millard: What about a prompt command?
Ans: We looked at such a mechanism, but did not pursue it.

Schonfelder: We need a simple device to permit a dialogue between the user and the program.

Shen: There is a problem with a parameter in a FORMAT statement, e.g., E8 may be a symbolic constant.

Compile-Time Facilities

References: W-28, Clarke, Methods for defining the Macro Attribute
W-32, Weisz, Comments on the Compile-Time Facilities

Discussion:

Metcalf: These features provide for source code management which is needed. There should be a way to declare a global compile-time variable.

Weisz: The USING statement is imprecisely defined. Macros should have parameters.
Ans: S6 is incomplete.
Ans: I'm glad to have a chance to see it at this stage.

Ampt: These facilities do provide a way of structuring the source, but you must change at least one line and reprocess to achieve a different structure. We need I/0 at processing time instead. it should be clearer what the levels are. This is usually done by an operating system, but they are not standardized. We must be careful not to limit implementations. Why does the USING statement refer to different files? It should function more like the SHARED DATA block. I object to the term "compile-time". It is an implementation-oriented term rather than a language-oriented term.

Humar: These features will burden the language if put into core.

Reid: We should leave macros to preprocessors and keep the core small.

Snoek: They allow you to define your own syntax. It is the only way to insert a block of non-executable statements.

Weisz: There is a great advantage in having one source that can cross machines, stated in a standard way that is portable. Also efficiency at run-time is important. You can more easily give up processing time.

Reid: Preprocessors can be written in portable Fortran.

Schonfelder: A language is non-portable because of built-in machine dependencies. You need to get them out rather than devise a way to "manage" source. You are inventing two languages, one on top of the other. This makes it less portable.

We need to know how to define a module, and how to include a library. The USING statement is supposed. to provide this facility. We need to put data structures in libraries, perhaps also executable code. A library mechanism must be defined in the language. It is fundamental to the architecture of the language.

Straw Vote: Should macros be in the language? (5-2-10)

Language Architecture

References: W-14, Delves, DuCroz, Reid, Core Plus Modules in Fortran 8X
W-17, Clarke, F77-F8X Incompatibilities
W-23, Schonfelder, User Written Application Modules

Discussion:

Humar: I would like to see application modules built on the core only not the core plus extension modules

Schonfelder: Standard extension modules with new syntax won't be implemented. So you can't build applications on them. They may be specific to a machine type. I would be happier if the extension modules didn't exist.

Snoek: Then you would have to have all the syntax in core. It would be an impossibly large language.

Wagener: The extension modules were provided to solve problems for specific application usages. Putting everything into core makes people unhappy; nevertheless, we will probably have a large core.

Ampt: We need general solutions.

Meissner: Private opinion: X3J3 hasn't accepted macros. They will never accept abstract data types.

Clarke: Where does source form fit?
Ans: It is the one incompatibility.

Snoek: We all want the same thing in the end - a small core with general powerful facilities. I would accept extension modules with redundant syntax for a specific area

Character Data Type

Discussion:

Meissner: What we should have had is STRING.

Pollicini: You can do what you want with arrays of characters. Why have two different character types - fixed and varying?

Snoek: Only one should go in core. Why put fixed length character in core?
Ans: It is more efficient.

Metcalf: Fortran 77 character type is a mistake, but we should make it better by extension (character type with a varying attribute). We should not have two separate types.

Meissner: The issue is what happens across subprogram boundaries.

Bit Data Type

Reference: W-21, Metcalf, Bit Data Type in Core?

Discussion:

Munchhausen: I am opposed to bit strings being right adjusted. It makes it very difficult to extend to varying length bit strings. If you want to convert them to integers, use functions.

Snoek: Don't make the same mistake you did with character - start with varying length bit strings. Having them right adjusted is terrible. There should be no association with storage. Use functions to compare or transpose them.

Weisz: You need to transfer bit data to character data for control characters.

Schonfelder: I oppose violently. You should not get characters by knocking about with bits.

Meissner: You can go through integers - bit to integer to character.

Weisz: I am in favor of left justification with a function to get the actual length of bit strings.

Schmitt: I prefer a hierarchy for operations rather than .BAND., .BOR., etc.

Weisz: I would like them to look different. Remembering precedence rules is not easy.

Adams: Should we have bit data type at all? It is useful to hear your views.

Ampt: We are now getting rid of storage association. Let's not introduce a new type that relates to storage so strongly.

Schonfelder: The need for bits and strings vanishes if we have a general facility. All we need is a flexible array of characters.

Meissner: That is theoretically an excellent approach, but it won't sell.

ter Haar: Is bit more than the logical data type?

Meissner: You would have to add operations to logical arrays. It is simpler to add a new data type.

Metcalf: Data conversion requires the bit data type.

Weisz: I agree with Mike. We must deal with the real computer. This is a basic underlying representation - not the theory of reals, etc., where the internal representation doesn't matter. This is a practical concept that must be supported by Fortran. For applications in the real time world this is very important.

Mas: We need bit data type as it is in PL/1, especially for signal processing. I want the same facility in Fortran.

Straw Vote: Should the bit data type be in the language? (19-0-9)

Straw Vote: Should the bit data type be in core? (3-12-11)

Straw Vote: Left or right adjustment? (left, 14 - right, 0 - undecided, 12)

Precision Data Type

Discussion:

Meissner: Argument association becomes more complicated because the dummy argument may have generic precision.

Reid: Numerical I libraries currently have double and single precision routines. The appropriate copy is obtained at load time.

Meissner: There is potentiality for a combinatorial explosion, thousands of versions of routines. To limit this, it would be reasonable for all arguments to have the same precision.

Weisz: What about the value returned?
Ans: It must be the same.

Ampt: If a routine is invoked more than once, would it always be the same routine?
Ans: There may be different types In another invocation.

Shen: You don't need user generics. The types can be specified in an interface information block.
Ans: Take for example a Bessel function routine. It may use different algorithms for different precisions. This would only be needed for the sophisticated user, or maybe there is a privileged user mechanism (too horrible to contemplate) such that others are not permitted to use such features. The ordinary user doesn't need this, but that does not mean that it shouldn't be in the language.

Weisz: How do you know what the precision is?
Ans: There are intrinsic functions that return that information.

Meissner: This is not really dangerous because people will not do dangerous things as they don't today.

Procedures and Functions

Discussion:

Snoek: The keyword form is confusing. It seems to be an assignment to the dummy argument name.

Humar: Can the INITIAL attribute be used to set a default value on the calling side? What specifies a default on the called side?
Ans: The PRESENT function allows you to specify a default on the called side. It would be very bad practice to change the keyword names on the calling side. The Interface Block is practical. Why does it help to put it in a SHARED DATA block?
Ans: The processor has to look in fewer places to find the information it needs.

Ampt: If the information appears only in a SHARED DATA block, then both the called and calling routines can access it there. Readability is a good reason to use keyword names. Association by position is no problem for the compiler, but it is harder on the programmer who is maintaining the program.

Shen: If you allow the keywords to change, you should also allow the subprogram name to change.

Weisz: I support the objections to changing the keyword names.

Meissner: The interface information block must be available somewhere. If it appears only in the called routine, you have dependent compilation. If it appears in the calling routines as well, you have duplicate

declarations. If it appears in a SHARED DATA block that is inherited, you still have a limited dependent compilation.

Snoek: There seem to be three choices for places to locate the Interface Block: the called routine, a SHARED DATA block, or the calling routine. When you are calling a routine from many different places, you may want to override the names of the keywords in one of the calling routines. You might want the arguments to have different defaults in one of the calling routines.

Wagener: Libraries are built from the bottom level. People want to use these features in libraries. You must accommodate both points of view.

Schonfelder: There is a potential redundancy in the language. USING provides a method to access modules and their interfaces. You can also use the SHARED DATA block. Do we need both to get at this information? The USING statement seems to have higher potential.

Straw Vote: Should interface information be required to appear in a SHARED DATA block? (4-2-16)

Straw Vote: Should there be a mechanism to rename keywords? (3-14-10)

Internal Procedures and Interprocedure Data Sharing

Discussion:

Meissner: Have internal procedures met the original goals. One of the goals was to find a replacement for statement functions and the extended range of the DO, both of which had open scope. The Internal Procedure has a closed scope. There is a problem with IMPLICIT, otherwise INHERIT ALL would be easy.

Martin: One of the goals was a replacement for ENTRY. It has not met that.

Weisz: It is important to be able to hide names in an internal procedure. INHERIT is backwards from other languages.

Ampt: The GROUP concept provided a nice replacement for ENTRY.

In other languages, variables are explicitly declared.

Meissner: This is the key issue. There are default attributes for undeclared variables. This is incompatible with the idea that everything is global unless declared local. X3J3 always votes against the addition of a simple remote block.

Ampt: With COMMON we had association by storage location. With GLOBAL DATA, we had association by name. With SHARED DATA, we have association by name and type and interface information. Next will be shared programs.

Dynamic Storage and Recursive Procedures

Discussion:

Meissner: Are recursive procedures needed?

Ampt: I am still waiting for examples of the need for recursive procedures. But should there be a difference between recursive and non-recursive procedures? Recursive should be the default.

Metcalf: I found one example. In the survey we took, there was a very negative response to recursion.

ter Haar: It is the only way to solve some problems.

Weisz: You should be able to declare that a routine is non-recursive for efficiency.

Schonfelder: There is a lot of simulation of recursion in Fortran programs. Recursion has a definite place. Once you allow a stack, it is free.

Snoek: No, they are not needed - recursive algorithms can be made cyclic.

Straw Vote: Should recursive procedures be in Fortran 8X? (16-5-5)

Meissner: In Fortran 77, things are stated ambiguously. In a recursive environment, there is more than one possible obvious extension. Recursion can be looked at in two different ways: a copy of the text or a reentrant routine. Each view gives a different interpretation.

12. Other Features

a. Interfacing and Scoping (C. F. G. Ampt)

Reference: W-9, Ampt, Event Handling, Scoping, and Interfacing

Summary:

The scoping problems could be solved if we had a model of the language in which data objects were determined by their attributes, many of which could be dynamic. The properties of a data object are its value, name, size, and reference order.

Discussion:

Shen: It would be difficult to modify Fortran to get dynamic scoping. I don't see much application for it.

Ans: Variant data structures do permit dynamic typing. We may go further.

Weisz: To apply this Fortran would have to undergo thorough investigation. Are the underlying concepts fulfilled by Fortran up to now?

Ans: We are talking about the value of a data object, but we don't have a general framework for the discussion. An operation appears to be different for different types thus it is an attribute. Some machines store integers and reals in the same way and only the operation (e.g., divide) changes the results.

Schonfelder: Operations are an integral part of the definition of a data type. A complex X behaves differently from a real array X(2), even though they are both two reals.

Ampt: The interface problem could be solved if we had a general framework for the language. Now there are many exceptions and idiosyncrasies. Some things are dynamic; some are static.

Wiessner: "Weathervane" explored some of the same issues. In Fortran 77, the attribute is type. We kept forgetting about character with an additional attribute of length. With Fortran 8X we have to change the property of type to attributes because many objects have more than one characteristics. We must always think of the other attributes. Objects with the old type may match objects with the new attributes, but there are many rules to determine.

Ampt: A .EQ. B has different meanings when A and B are integers or reals or arrays or bits or characters.

Adams: What are you recommending we do with 8X? We must begin putting a document together in August.

Ans: Make a framework to describe data objects. Show what happens to data objects. It's difficult to read the 77 document. 8X will be impossible.

Meissner: Chapter 4 of "Weathervane" is an attempt to describe the framework.

Adams: Should Fortran 77 be rewritten this way before we start to merge 8X features into it?

Schonfelder: It may be possible to remove things from the Fortran 77 description, that no longer need to appear.

Adams: We have to worry about the user community's acceptance of Fortran 8X.

Ampt: The description of a statement in Fortran 77 was different from its description in Fortran 66, but the meaning was the same. It is impossible to keep the same words in Fortran 8X. You must create a new framework.

Meissner: A user's view of how complicated a feature is depends on how well it is described. The description of Fortran 8X must be more organized than it was for 77. it cannot be taken feature by feature. You cannot possibly retain the text of Fortran 77. Just think of the implications of the removal of storage association.

Adams: It will take two years to do this. Perhaps 8 people could do it in a short time, but working with the whole e committee will take a long time.

Wagener: There is value in identifying a model. It's a healthy exercise. I think we should try both approaches. We should not divert effort from what we already have started. This kind of a framework would be highly desirable.

Straw Vote: X3J3 should create a general model for describing data objects
(yes, 11 - no, 0 - don't understand, 8 - undecided, 7)

b. Matters of Importance for Numerical Subroutine Libraries (J. K. Reid)

Reference: W-12, Reid, Matters of Importance for Numerical Subroutine Libraries

Summary:

A lot of progress has been made since the last Fortran Experts Meeting (October, 1980), toward providing the functionality needed for numerical libraries. Capabilities that are still not present are the declaration and manipulation of "ragged edge" arrays and user-defined operators for data structures. A smaller point brought out in discussion, was that the way the INHERIT mechanism is defined for internal procedures, it may not be possible for the internal procedure to inherit an automatic array from its host. This would seem to be necessary.

c. Abstract Data Types (J. L. Schonfelder)

References: W-23, Schonfelder, User Written Application Modules
W-24, Delves, User Defined Operators

Summary:

The combination of the array constructing facilities and the FORM facilities, already in Fortran 8X, go a long way toward providing "abstract data types". The FORM declaration gives a "name" to a structure template. The "name" can be used to declare objects. A data type is the definitions (declarations) of a set of values that are transformed in a specified way under a set of associated operators (or functions). There is a difference in convenience between operators and functions: operators can be used in infix notation and functions require a polish notation. An operator is like a two-argument function (input arguments only) plus a priority. See W-23 for an example syntax. Either existing operators could be overloaded, or some delimiters could be chosen to set off a name. (There are problems with using period, which would be upward compatible with Fortran 77, unless the currently proposed structure qualification symbol were changed.)

Derived data objects for which a natural INFIX notation Ion is needed are:

- non standard arithmetic

  extended or arbitrary variable precision
  extended range
  significance arithmetic
  interval
  rational

- complex

-2-dimensional, 3-D, and n-D vector algebra

- mathematical matrices

- tensor calculus

- relational algebra databases

The infix operator and the functionality exist. What's missing is a way to put things together to a create a different sort of library. If users had access to such a facility in a small language, they could adopt it to their own needs in their own terms.

Discussion:

Adams: It would be a mistake to call this added functionality "abstract data types". What is required that is missing from the current proposals?
Ans: The ability for the user to define operators on data objects. It is not essential that they be infix operators.

Reid: This would satisfy the new criteria, because if it were proposed, two other features (COMPLEX and BIT) could be taken out. LOGICAL with a variable length should be retained, but CHARACTER could go. I would like to introduce a new concept - SEQUENCE.

Wagener: I find this proposal intriguing. I like the use of the function definition we already have. John Reid's libraries are not necessary at compile-time; yours will be, not for the body, but for the definition.

Weisz: Put them in a macro library and call them macros.

Ampt: I like the expressiveness this would provide, but I am concerned about operator priorities in a complicated expression. I would like to see a paper with a lot of examples. Unary operators should be included; more than two operands are not needed.

Meissner: This proposal is like a macro facility.
Ans: It is more complicated than source replacement.
Ans: Macros can be more general than that.

Oldenhoeft: Use the facilities we already have - parentheses to specify order of operation. Then you avoid the compile-time issue.

Ans: You can't avoid the compile-time issue. An operator is its name and the types of its operands (arguments). This information must be known at compile-time in order to parse the expression. The operators may return structure results. it is important to allow overloading.

Shen: An operator definition facility would allow a user to define an error facility. An operator is just a special kind of function call. There is nothing new or strange with operator definition. Operators eliminate the need for parentheses. The order is generally left to right. Priority was introduced in Ada and Algol68.

Schonfelder: If an operator is overloaded, its priority can't change.

Weisz: If you define operators, there must be type checking. The compiler can check for appropriate operands. You can't do this checking for the functional form (except intrinsic functions), unless it is at load time. More of a burden would be put on the user.

Ans: It would be the same in either case.

Schonfelder: User-defined operators will permit a small core. X3J3 can define COMPLEX and CHARACTER in extension modules. Portability is solved. If a feature is not defined in the standard, the user can provide it.

d. Fortran for Contemporary Numerical Computation (U. Kulisch, C. Ullrich)

Reference: W-15, Kulisch, Miranker, Ullrich, et al, Fortran for Contemporary Numeric Computation

Summary:

Fortran will be used for quite a while by numerical analysts. The standard should say something about the computer arithmetic used. Programming languages have traditionally lacked precise definition of the arithmetic operations and roundings. This extended language provides exact precision to within the last digit. It requires extension of the character set to represent new operators (monadic and dyadic) and new data types: segment, complex segment, vector, matrix, tensor. It can be implemented on small or large computers. The advantages are shorter programs, easier debugging, and shorter compilation time. We invite everyone to view the demonstration system in the foyer.

Discussion:

Snoek: For other reasons we need to provide abstract data types within Fortran. Could we get this the same way?

Ans: No, you can't program rounding operators. They must be in the hardware.

Wagener: Is that the only necessary addition - rounding operators?
Ans: No, 15 operations must be available.

Schonfelder: Won't the extended precision capabilities in Fortran 8X give us the needed precision.
Ans: No, all calculations would have to be done in the highest possible precision, (i.e., 50 decimal digits).

Wilson: Compare this with IEEE.
Ans: The IEEE proposal is incomplete; they only solve the problem for scalars.

Schonfelder: The standard can't define how the hardware is built. We could put some of this information in an appendix of the standard, and provide an intrinsic function to inquire if it has been done.

Meissner: You should approach IEEE to standardize the arithmetic before approaching it from a language that depends on the arithmetic. What bothers me is the 3 missing operators. You would have to build an additional piece of hardware to get them. Scalar product hardware is needed.
Ans: Yes, you do need more. A 2K box is needed.

Shen: These operations are realizable both in hardware and software. I would like to see these in an extension module not in core. The implementor can decide to use it or not. Why separate matrix from tensor?
Ans: For efficiency.

Munchhausen: X3J3 should consider this for the Enhanced Scientific Module.

Snoek: This should be seriously considered. We need more accurate results, consuming less time to obtain.

ter Haar: Is it implementable in software?
Ans: Yes, the demonstration system is coded in assembly language.

Straw Vote: X3J3 should take notice of these ideas and incorporate them in 8X if possible. (18-2-9)

e. Event Handling in Fortran 8X (W. Kneis, H. Sobiesiak)

References: W-26, Sobiesiak, Introduction to Fortran Event Handling
W-27, Sobiesiak, A Proposal for Event Handling

Background: There are 11 technical committees within EWICS. Technical Committee 1 (TC1) has responsibility for a real-time subroutine package based on Fortran 77 (IRTF). TC1 is currently working within X3J3 (Task Group 2) in collaboration with X3J3 Subgroup 11 (Input/Output) to develop a proposal for an event handling capability in Fortran 8X.

Discussion:

Wilson: Can events be "queued"?
Ans: Not as such. There is a way to suspend or delay the handling of an event until the programmer allows it. When a handler is entered, other events are suspended.

Reid: I am concerned about the interaction with the array proposals. What happens when a divide check occurs in a WHERE block.
Ans: There are several ways it could be dealt with. The event mark might be an array of status indicators, if it is important to know which element failed.

Ampt: Is the handler a separate program unit, a block, or what? Do you have to indicate how to call it?
Ans: It is essential that it be a separate part - a program unit or internal procedure, perhaps.

Meissner: Since WHERE applies to assignment, not to the divide operation, if the divide check occurs during an operation that is masked, it should be ignored. If it is not masked, it should be acted on. The description needs to be worked on.

Shen: For the event facilities to be useful for real time, you must allow different processors to set event marks. It must be as powerful as a semaphore. PL/1 of 1972 has a section that was removed from ISO PL/1 that shows the difference between PL/1 and the new view. There was an effort to extend PL/1 to real time. It would be useful to look at that.
Ans: We hope it will be as powerful as semaphores. We haven't addressed that yet. We need a mono-programmed solution first.

Kneis: Example with regard to the contention of resources:

Statements that address RESOURCEMARKS are:

```
LOCK (resource)
UNLOCK (resource)
TLOCK (resource)
```

TLOCK (test and lock) first tests the resource and if it is free locks it so no intervening action can occur.

Wagener: In IRTF event marks are in COMMON and are global to all program units. For Fortran 8X, are they local or in SHARED DATA?
Ans: We need both: local and global.

Wagener: Delaying activation of the handler also bothers me.
Ans: You can defer the reaction to an event until later, but you must be sure that the relevant information will be available to the handler.

f. ECMA Fortran Questionnaire (C. Mas)

The questionnaire (W-29) is based on the S6 document and X3J3 minutes. It is intended for distribution with a tutorial on Fortran 8X (in French). The object is to get preliminary opinions from 100-200 Fortran 77 users. The survey will take place in September or October in France. It would be beneficial if others would coordinate and take similar surveys in the last quarter of this year.

There was some interest in distributing the revised S6.81 with the revised questionnaire. S6 is an internal document, not really designed for tutorial purposes. It was suggested that Loren Meissner's summary article might be more appropriate.

Discussion:

Reid: Recursion and automatic arrays should not appear in the same question.

Metcalf: Space should be left for additional items.

9. Overview of Recommendations Made (J. Humar)

A few quotations (from comments made at this meeting, distributed documents, and X3J3 Minutes) can be used to set the stage.

Quotations:

Ampt: -- For every feature we add we must delete two --

Schmitt: -- We feel that you are adding too many new things to the language --

Humar: -- At this stage we have the chance to keep the Fortran 8X core very compact - -

Reid: -- I like the core to be concise and elegant and would therefore oppose the inclusion of macros and compile time features even though they may be useful and desirable --

Muxworthy: -- A persistent criticism of X3J3 at the 1978 and 1980 Fortran Experts Meetings was that the committee were moving too for, too fast and this criticism has been echoed to some extent at domestic U.K. meetings

-- We urge the committee when defining the core to ensure that it will be implementable on smaller machines --

Adams: -- We can invent more and more structured programming constructs forever. I don't think we need all this, and I'm afraid we'll have trouble selling it to the rest of the world --

I just feel the committee is in a runaway mode, proposing everything --

Meissner: -- My conception of core is that it should simple, compact and should have all my pet features --

There seems to be a lot of sentiment to keep the core compact. What did we do at this meeting to further that aim? We voted against:

1. DO WHILE

2. Logical CASE

3. Multiple statements in a line

4. Bit data type in core

On the other hand, we either proposed or did not discourage the introduction into the language (and possibly into the core,) of the following:

1. Entity-oriented declarations

2. Naming of control blocks

3. Multi-level EXITs

4. CYCLE and multi-level CYCLEs

5. EXIT from any type of control block

6. CASE statement

7. A large array facility

8. Event handling

9. User- defined lists

10. Several sizes of INTEGER

11. Abstract data type

12. User defined operators

13. Macros

14. Compile time features

15. Enhanced procedure CALL (OPTIONAL argument, PRESENT function, INTENT)

16. Recursive procedures

17. Keyword arguments for intrinsics

18. Name abbreviation in data structures, Array reference in structures

19. Splitting names between records

20. Extensions to internal procedures

21. Name-directed formatting

We should take a hard look at what we are recommending.

h. Compatibility of Object Code, Fortran 77 and 8X (G. E. Millard)

We must not overlook the practical aspects of moving to the proposed new language. Will it be a natural progression or a step change? In the case of Fortran 77, it was a simple progression. What will the transition be for 8X? Will such considerations influence X3J3 in determining the content of the language?

There are a number of existing libraries that should be accessible to both Fortran 8X and Fortran 77, which would indicate that the procedure interfaces be kept compatible. But there is pressure from the array area to force recompilation. We had a little experience with this in Fortran 77 with character data. It is important to the success of Fortran 8X that the transition be made as painless as possible.

Discussion:

Meissner: I call attention to the paper by Rich Ragan in Fortec Forum (W-6) on array argument handling.

Wagener: Up until 6 months ago we were not too worried about these practical implications, but in the last 3 months we have been paying close attention to them.

Martin: These are implementation considerations, that are not directly addressable by a standards committee, but many on the committee are implementors and are concerned. If implementors pay attention to the direction the standard will take and begin to plan ahead, they can do much to ease the transition.

Metcalf: It is easier to recompile than to worry about compatibilities.

i. Syntactic Lists (P. A. Clarke)

Lists appear frequently in Fortran 77, not always with the same syntax. If there were a unified method of constructing lists, then the language could be simplified and redundant features and syntax eliminated. Lists are used to specify sequence, range, and repetition (in DATA and FORMAT). For example there are three ways to indicate range:

1,10 in the DO statement
A-Z in the IMPLICIT statement
1:N in subscripts

13. Summary of X3J3 Members' Comments on S6 Review

The purpose of the summary was to make sure X3J3 took the correct impressions away with them. Each moderator summarized the suggestions made with regard to features in the chapters of S6. In general, the Fortran Experts are in favor of a small, concise core. User defined data types (with operations on those types) were suggested as a mechanism to achieve this goal. Jeanne Adams thanked the attendees for giving X3J3 a fresh point of view.

14. Two questions were discussed: When? and Where? It was resolved that the next meeting should be held some time between 12 and 18 months in the future, ie., June 1983 up through January 1984. We currently have two invitations, one from Geneva, Switzerland and one from Germany (probably Berlin). We will accept the invitation that was received first, when it can be determined which one that was.

There will be an informal get-together during the time not devoted to plenary sessions at the SC5 Meeting in Ottawa, Canada in September of 1983.

15. The delegates expressed appreciation to the hosts the Technische Universitat Wien, Gerhard Schmitt, and all others who assisted with local arrangements.

16, The meeting adjourned at 5:30 PM on Thursday, June 17, 1982.