Minutes of Meeting

ISO/TC97/SC22/WG5 Fortran

1 to 4 July 1985

Bonn, Federal Republic of Germany

X3J3/179

CONTENTS

RESOLUTIONS CONSIDERED

ISO/TC97/SC22/WG5 (Fortran) at its meeting of July 1 to July 4, 1985 in Bonn, West Germany, has considered the following resolutions:

R1. That WG5 believes that the work of X3J3, which has resulted in the development of S8, is in general representative of the needs of the Fortran community worldwide and encourages X3J3 to complete its work of forming S8 into a draft standard. WG5 further believes that any attempt to make a major downward revision would be retrogressive and detrimental to the interests of Fortran users. (17 yes - 0 no - 0 undecided - 12 abstain)

R2. That WG5 believes that Section 1.1 of S8 should be revised to include the meaning described by the following paragraph:
This standard specifies the form and establishes the interpretation of programs expressed in the Fortran language. The purpose of this standard is to promote portability, reliability, maintainability, and efficient execution of Fortran programs for use on a variety of computing systems. This standard is intended as an upwardly compatible extension to the preceding Fortran standard, X3.9-1978, informally referred to as Fortran 77. All standard-conforming Fortran 77 programs are standard conforming under this standard. (29-0-0-0)

R3. That WG5 requestes that C. Ampt prepare a proposal for a CONVERSION NONE statement (or similar optional functionality) to be submitted to WG5 and X3J3 for consideration. (17-5-6-1)

R4. That WG5 requests that the DIN Fortran Group prepare a proposal for stream I/O to be submitted to WG5 and X3J3 for consideration as a supplementary or incremental standard. (20-0-7-2)

R5. That WG5 believes that the text of the document should be such that the deprecated features can easily be left out. (26-1-2-0)

R6. That WG5 supports the inclusion of type declarations with attributes as they appear in Section 5 of S8. (20-0-8-1)

R7. That X3J3 reconsider the definition of the INITIAL attribute and the INITIALIZE statement so that only one of these constructs is necessary, by either:

(1) disallowing the initialization of sub-objects and subsequently deleting INITIALIZE

(2) extending the sytax of the INITIAL attribute

(3) introducing any other solution which prevents the (partial) duplication of functionality.

(17-3-9-0)

R8. That WG5 believes that it should be possible to initialize sub-objects. (24-1-4-0)

R9. That WG5 supports the inclusion of keyword and optional arguments in user defined procedures. (26-0-3-0)

R10. That WG5 believes an INTENT(IN) argument should be explicitly stated as being constant throughout each invocation of a procedure. (19-2-8-0)

R11. That WG5 wishes that, when a processor supports lower case characters, their collating sequence be defined in the standard. (27-0-2-0)

R12. That WG5 believes that the Fortran Character Set must contain only characters from Table 1 of the Basic Code Table of ISO 646 (1973) (7-bit coded character set for information processing interchange). In the case of those whose place has been reserved for national use, an alternative representation not including such characters must be defined in the standard. (20-1-7-1)

R13. That WG5 believes that examples should be permitted in the body of the standard. (19-3-7-0)

R14. That WG5 supports the statement that features denoted as deprecated in Fortran 8X should be considered for possible deletion during the development of the next Fortran standard. (27-0-2-0)

R15. That WG5 recommends that environmental intrinsics to provide information (e.g. ASCII/EBCDIC, bits per character) about the character data type should be added. (12-0-16-1)

R16. That WG5 recommends that a RANDOM intrinsic procedure should be added. (21-4-4-0)

R17. That WG5 supports the current list of array intrinsic functions as being satisfactory. (21-0-5-3)

R18. That WG5 wishes to express its concern to X3J3 about the possible performance effects of passed on precision and requests that X3J3 clarify the possible implementation mechanisms which can ensure effective efficient implementations. (20-0-8-1)

R19. That WG5 urges X3J3 to try to ensure that the design of Fortran 8X does not prohibit the calling of existing Fortran 77 object code. (24-3-0-2)

R20. That WG5 believes that EXIT should apply to CASE and block IF statements. (12-12-5-0) Failed

R21. That WG5 believes there should be some processor-independent means in Fortran 8X of determining input record lengths. (16-0-12-1)

R22. That WG5 requests that its convenor forward SC22 a request to create a new work item to deal with a clarification of the file handling facilities which may be assumed by a high level language. (25-0-4-0)

R23. That WG5 requests that BSI prepare a proposal for requirements on processors for the issuing of error and warning messages related to the use of extensions to the ~~high~~ standard and of deprecated features. (12-7-8-1)

R24. That WG5 believes that it should be possible to use the procedure interface block both to define a procedure and to describe a reference to the procedure. (18-0-11-0)

R25. That WG5 requests that the Canadian Fortran Working Group prepare a proposal for the extended call to the character function ISCAN, VERIFY, and INDEX as discussed in Geneva and described in the Canadian submission in the minutes of that meeting to be submitted to WG5 and X3J3 for consideration. (27-0-2-0)

R26. That WG5 believes that Fortran 8X should be consistent as far as possible with ISO 2382/15, Data Processing Vocabulary/15 Programming Languages. (15-2-6-5)

R27. That WG5 supports the definition of BIT and CHARACTER as string data, both having length-specification, and all applicable aggregate features such as assumed/supplied lengths and ALLOCATE-ability. (10-12-6-1) Failed

R28. That WG5 wishes to allow a scalar of any type to be allocatable. (22-0-7-0)

R29. That WG5 supports the inclusion of indexed sequential file access as described in document number 28 in the minutes. (11-5-13-0)

R30. That WG5 wishes to extend the standard so as to be able to make the keyword CALL optional and to be able to pass an argument list in the form of an I/O list. (9-18-1-1) Failed

R31. That WG5 believes that user defined operators should allow a reasonable set of unambiguous strings formed from the characters + - / = < >. (12-6-11-0)

R32. That WG5 requests that the DIN Fortran Working Group prepare a proposal for the specification of a record length in the OPEN statement for sequential files, either for fixed length records or for the maximum record length for files with variable length records, to be submitted to WG5 and X3J3 for consideration. (11-2-13-2)

R33. That WG5 would like to express its appreciation to the Convenor, Jeanne Martin, the Chairman, Jeanne Adams, the organizer, Karl-Heinz Rotthauser, and to the GMD and its staff and those organisations who have provided further support. (29-0-0-0)

Agenda

Meeting of ISO/TC 97/SC 22/WG 5 - Fortran

Sankt Augustin (Near Bonn) West Germany

1985-07-1/4

9.30 - Monday morning, 1 July 1985

1.	Opening of the meeting	Prof. Dr. Kruckeberg (DIN/GMD)
2.	Convenor's opening remarks	Jeanne Martin
3.	Welcome of delegates	
4.	Roll-call of delegates	David Muxworthy (Secretary)
5.	Election of Chair	
6.	Minutes of the Geneva Meeting (April 1984)	
7.	Adoption of the Agenda	

11.00 - 11.15 COFFEE BREAK

8.	National activity reports	Heads of delegations
	Liaison with ISO/TC 97/SC 22/WG 1	Kees Ampt
9.	Procedural matters	
10.	Report from the SC22 Planning Meeting (April 17-19, Washington, DC)	Jeanne Adams
11.	Summary of X3J3 Actions since April 1984	Jeanne Martin (Secretary, X3J3)
	Concerns of the Canadian Fortran Working Group	Bert Buckley
	Comments by the Netherlands Fortran Specialist Group	Leo ter Haar
	Comments on Fortran 8X	M K Shen

12.30 - 14.00 LUNCH BREAK

14.00 - Monday afternoon

12. General Overview - Document X3J3/S8
(S8 is a preliminary working document that is being developed by ANSI/X3J3 as the draft for the revised Fortran standard. It is the task of WG5 to evaluate this document as a proposed International Standard and to determine any obstacles that might stand in the way of its adoption).

Section of S8	Discussion Leaders
Section 1, 2, 3 (Introduction)	Wagener/Brainerd
15.15 - 15.30 COFFEE BREAK	
Section 4 (Data Types)	Freeman/Smith
18.00 COCKTAIL PARTY at the GMD	Prof Dr Szyperski (Managing Director of GMD)

9.00 - Tuesday morning, 2 July

Section 5 (Data Objects) Schonfelder/Buckley

10.45 - 11.00 COFFEE BREAK

Section 6 (Expressions) Smith/Schonfelder

12.30 - 14.00 LUNCH BREAK

14.00 - Tuesday afternoon

Section 7 (Execution Control) Martin/Muxworthy

15.15 - 15.30 COFFEE BREAK

Section 8, 9 (Input/Output) Freeman/Johnson

9.00 - Wednesday morning, 3 July

Section 10, 11 (Program Units) Hirchert/Wagener

10.45 - 11.00 COFFEE BREAK

Section 12 (Intrinsic Procedures) Wilson/Marshall

12.30 - 14.00 LUNCH BREAK

14.00 - Wednesday afternoon

Visit to Cologne

14.00 Departure by Bus from GMD to Cologne

15.00 Visit the Roman-Germanic Museum, near the Cathedral

16.30 Excavations under the Cathedral

18.30 - Wednesday evening

18.30 Meet at Cologne Cathedral before going to dinner at one of Cologne's traditional beer houses, "Zur Malzmuhle"

22.00 Return by bus to Bonn and hotels

9.00 - Thursday morning, 4 July

Section 13 (General Concepts)	Brainerd/Johnson
Section 14 (Deprecated Features)	Marshall/Schenk

10.45 - 11.00 COFFEE BREAK

14. Liaison activities

- Graphical Kernel System, Fortran Binding	Ingolf Grieger
- ISO/TC 97/SC 22/WG10	Brian Meek
- ISO/TC 97/SC 1/WG5	Rainer Zimmer

12.30 - 14.00 LUNCH BREAK

14.00 - Thursday afternoon

15. Presentation by delegates

16. Comments from delegates

17. Any other technical items

18. Recommendations

19. Future meetings

20. Closing Business

21. Adjournment

Friday, 5 July 1985 - FORTRAN FORUM - Bonn University

Attendance List at ISO-FORTRAN-meeting
01.07.-05.07.1985 Kleiner Horsaal, C1-206

Name	Institution	Address
Jeanne C Adams	National Centre for Atmospheric Research	PO Box 3000 Boulder, Colorado 80307
Kees Ampt		Snijdersdreef 7 B-1900 Overijse
Pierre Andre Boutrouille	Honeywell Bull	68 Route de Versailles F-78430 Louveciennes
Walter S Brainerd		1580 Camino Redona Los Alamos, NM 87544
Bert Buckley	Dept of Mathematics and Statistics Dalhousie University	Halifax, NS B3H 4HB
Francoise Ficheux-Vapne	Electricite de France Direction des Etudes et Recherches	1, Avenue du General de Gaulle F-92141 Clamart Cedex
Murray F Freeman	Bell Communications Research	PO Box 411 Pavpack, PA 18451
Ingolf Grieger	Universitat Stuttgart	D-7000 Stuttgart 80
Kurt Hirchert	Computing Services Office University of Illinois	1304 W Springfield Urbana IL 61801
E Andrew Johnson	Prime Computer	500 Old Conneticut Path Framingham, MA 01701
Hanno Krainer	Computer Gesellschaft Konstanz	Max-Stromeyer-Str 116 7750 Konstanz
Levent Kurtarici	IBM Canada Ltd Dept 867	1150 Eglington Ave East Don Mills Ontario M3C 1H7
Herman Luttermann	Regionales Rechenzentrum fur Niedersachsen Universitat Hannover	Wunstorfer Strasse 14 D-3000 Hannover 91
Gerhard Manowsky	IBM Deutschland GmbH	Sommerhofenstr 2-4 7032 Sindelfingen

Neldon Marshall	EG&G	PO Box 1625 Idaho Falls, ID83415
Jeanne Martin	Lawrence Livermore National Laboratory	L-300 Livermore, CA 94550
Christian Mas	IBM France	Tour Septentrion Cedex 9 F-92081 Paris La Defense
Brian L Meek	Computer Unit Queen Elizabeth College	Campden Hill Road London W8 7AH
Meinolf Munchhausen	Siemens AG K D ST SP	Otto-Hahn-Ring 6 8000 Munchen 83
David T Muxworthy	CAST University of Edinburgh	1 Roxburgh Street Edinburgh EH8 9TA
George Paul	IBM T J Watson Research Centre	PO Box 218 Yorktown Heights NY 10598
Aurelio Pollicini	JRC-ISPRA-Informatics Centre	I-21020 Ispra
x Karl-Heinz Rotthauser	Gesellschaft fur Mathematik und Datenverarbeitung mbH	Postfach 1240 5205 St Augustin 1
Dipl-Ing Gerhard Schmitt	EDV-Zentrum TU Wien Digitalrechenanlage	Gusshausstr 27.29 A-1040 Wien
Lawrie Schonfelder	Computer Lab University of Liverpool	PO Box 147 Liverpool
M K Shen	Bayerrische Akademie der Wissenschaften	Postfach 340 238 D-8000 Munchen 34
Brian T Smith	MCSD Argonne National Laboratory	9700 South Cass Ave Argonne, IL 60439
H Sobiesiak	KFZ Karlsruhe	Postfach 3640 D-7500 Karlsruhe
Mario Surdi	IBM	8 Locust Lane New Paltz, NY 12561
Manfred Tabler	Siemens Ag K D ST SP	Otto-Hahn-Ring 6 D-8000 Munchen 83
Leo ter Haar	Kon/Shell Expl.+Prod.Laboratorium	Postbus 60 NL-2280 AB Ryswyk

Prof Dr Christian P Ullrich	Institut fur Angewandte Mathematik Universitat Karlsruhe	Postfach 6380 7500 Karlsruhe 1
David Vallance	Computing Services University of Salford	Salford M5 4WT
Jerrold L Wagener	Amoco Production Co	PO Box 3385 Tulsa, OK 74102
Dr Christian Weber	Siemens AG K D ST SP	Otto-Hahn-Ring 6 8000 Munchen 83
Dr Gunter Wiesner	Hahn-Meitner-Institut	Glienicker Str 100 1000 Berlin 39
Alan Wilson	ICL	ICL Computers Ltd ICL House, Putney London SW15 1SW
Rainer Zimmer	Siemens AG K D ST SP	Otto-Hahn-Ring 6 D-8000 Munchen 83

DOCUMENT LIST

First column: document number used at the meeting
Second column: appendix number in these minutes (if appropriate)
Third column: title

1.	-	Agenda
2.	-	Excerpt from Report of TC97/SC22 Planning Meeting (paper 18 below)
3.	B 2	The Canadian Position of Proposals for A New Standard for Fortran
4.	C 1	Letter on GKS Fortran Binding to Gallop from Muxworthy
5.	-	Issues for the PHIGS Binding to Fortran, 1985.05.22
6.	C 2	SC22/WG16 (now WG10) Guidelines under consideration
7.	C 3	Letter on WG16 Guidelines to Meek from Muxworthy
8.	C 4	Letter on WG16 Guidelines to Meek from Buckley
9.	C 5	Letter on 1986 WG5 meeting to Kearney from Buckley
10.	-	ISO/DP 8651/1 GKS Language Bindings - Part I:Fortran (document dated 1984 December 5)
11.	C 6	Fortran Forum Agenda, Bonn, July 5, 1985
12.	B 4	Proposals for Fortran 8X, Netherlands Fortran Committee
13.	A 3	Results of a survey on Fortran 8X features in Japan
14.	A 1	China National Activity Report
15.	A 2	German National Activity Report
16.	-	Attendance List
17.	B 5	Comments on X3J3/S8 version 95, M.K. Shen
18.	C 7	Report on TC97/SC22 Planning Meeting
19.	A 5	U.K. National Activity Report
20.	A 4	Netherlands National Activity Report
21.	-	Draft British Standard method for specifying requirements for Fortran Processors, dated July 1985.
22.	B 3	Supplement to paper 3
23.	C 8	Preliminary information for meeting in Halifax, Canada, 1986.08.18-22
24.	-	Source Form from Matheny to Canadian CPL Fortran Working Group
25.	B 1	Summary of X3J3 Actions, Martin
26.	B10	ISO/TC97/SC22/WG10 Guidelines for the preparation of standards within SC22
27.	B 6	Procedures, S9.93 section 11, and Conditions, S9.93 section C, K.W. Hirchert.
28.	B12	Proposal for Fortran 8X from DIN Fortran Working Group
29.	B 8	DO Loops, J.L. Wagener
30.	B13	Modules for Scientific Computation, C.P. Ullrich
31.	B11	Comments on Fortran 8X Terminology, R. Zimmer

I. OPENING BUSINESS (AGENDA ITEMS 1 TO 11.1)

The meeting began at 9.30 am on Monday July 1, 1985. Participants were welcomed by the host, Karl-Heinz Rotthaeuser, and by Professor Krueckeberg, who is a member of the executive committee of the DIN NI Technical committee on Information Processing Systems and is manager of the GMD Institute for Foundations of Information Technology. Professor Krueckeberg gave a short presentation on the work of GMD.

The convenor of WG5, Jeanne Martin, welcomed the delegates. Jeanne Adams was proposed by Kees Ampt as chairman for the meeting and was elected unanimously.

After minor changes had been agreed, the agenda was approved.

The minutes of the meeting at Geneva on April 9 to 12, 1984 were approved.

National activity reports were given by the heads of delegations. Written reports appear in Appendix A.

Thanks were accorded to sponsors for making possible the activities on the Wednesday afternoon of the meeting.

Jeanne Adams reported on the SC22 planning meeting; a written report is in Appendix C7.

Jeanne Martin presented a summary of X3J3 actions since the previous WG5 meeting; a copy of her slides is in Appendix B1. There followed a discussion on the policy for distribution of the Fortran Information Bulletin following the Geneva meeting. This led to a discussion on how to get WG5 decisions to X3J3. It was decided to have a formal set of resolutions to be voted on at the end of the meeting. To this end, a recommendations subcommittee was appointed with the remit of identifying appropriate issues and of drafting the resolutions. Those appointed were Kees Ampt, Bert Buckley, Meinholf Muenchhausen, David Muxworthy and Gerhard Schmitt.

Jeanne Adams introduced a discussion on liaison between WG5 and X3J3. She characterized the main points as:

- The convenor corresponds with WG5 members
- The convenor reports to X3J3
- Member bodies are on the X3J3 mailing list
- WG5 meets each year
- X3J3 welcomes international members
- WG5 papers and minutes are send to X3J3 members
- WG5 can send proposals to the Convenor to be put in the pre-meeting distribution; the Convenor can find a sponsor within X3J3, the sponsor to correspond with the originator.

There followed an inconclusive discussion in which several people expressed concern at the low probability of those not on the committee being able to have a proposal accepted by X3J3. (Another discussion on general policy occurred under the heading S8 Section 8, a new looping proposal, below.)

II. TECHNICAL MATTERS (AGENDA ITEMS 11.2 TO 17)

GENERAL PRESENTATIONS BY DELEGATES

The following made presentations commenting on the Fortran 8X proposals:

Bert Buckley, for the Canadian Fortran Working Group (see Appendix B2, B3)
Leo ter Haar, for the Netherlands Fortran Group (see Appendix B4)
M.K. Shen (see Appendix B5)

Discussion leaders: Buckley, ter Haar, Shen   Scribe: Muxworthy

Summaries:

Buckley: The Canadian Fortran Group is concerned by the size of the language. We note that Algol 68 and PL/I were large and were not accepted, Pascal and C are small and were accepted. Fortran is a language for getting things done, so storage association is essential for things like linear algebra work. Fortran is no longer used for teaching in universities because it lacks recursion and pointers.

ter Haar: The Canadian paper reflects the feeling of many Fortran users in the Netherlands. Shell has written 30 million lines of code in 20 years and does not want to lose its investment. People are thinking of moving to C or Pascal. The language is too complex (cf the array facilities). The document needs to be made understandable by programmers. There is a need for full-screen I/O. We have additional proposals.

Shen: There is a need for pointers, WHILE loop, separation of loops, stronger typing. The document should be more easily understandable.

Discussion:

Adams: What would you (Buckley) take out?

Buckley: IDENTIFY, vector valued subscripts, some intrinsic array functions and so on.

Adams: Most members of X3J3 think the language is too big but noone can agree on removals. I think it is O.K. but the deprecated features list is too long.

Schonfelder: Algol 68 is a very small language with a big description. A long deprecated list is needed for features which need not necessarily be removed.

Adams: (to ter Haar) I would like people saying the language is "too large" to have specific recommendations for removals.

Wagener: (to Shen) Do you think the language is too big?

Shen: No.

Schmitt: The style of different sections is very different and the same thing is described in different places with different wording.

Please remember non-native English speakers and use the same wording for the same things. There could be a style appendix in the document, to be deleted at the last minute.

Adams: To SPARC "deprecated" means practically certain to be removed.

Schmitt: The size of the language is partly due to redundancy, for example entity-oriented declarations. Reducing the number of intrinsic functions will not reduce the size of the language. The examples are useful for students. Variable strings would be useful. Do not have pointers. They are 50% of Pascal but have no more functionality than automatic arrays give.

Meek: Algol 68 is neither big nor complex, it is small and regular. X3J3 had a design in 1978, core and modules, but this went to the wall. No new philosophy has appeared other than "it mustn't get too large". The deprecated features are really obsolete. Some redundancy is due to attempting to replace old by new. There could be backward extensions to cope with deprecated features. I suggest we now go for regularity. Too many decisions have been taken without an overall philosophy.

Adams: X3J3 voted down the idea of core and extensions.

Schonfelder: There is an unwritten rule that all language extensions must be regular.

Meek: The second Canadian paper says that 8X should contain only already existing features in Fortran or other language implementations. But it is no good trying to standardize existing practice - X3J3 has to be a language design committee; X3J3 has a duty to choose between existing practices. We should ask X3J3 to go back to basic philosophy, to ask what is the traditional user base and to ask if additions are in that tradition. We should use mixed-language programs to get at non-Fortran programming concepts.

Ampt: We can reduce a language by variety reduction or by variety control. I object to the Canadian position on standardization by development. Variety control is necessary. Reliability, quality and safety are also related to standards. These points can make a language more popular than others.

Schonfelder: I agree with Meek and Ampt. Because of time scales we are writing for the 1990s, up to 2000 and beyond. We cannot just standardize existing practice, we must develop and adopt material from computer science research on procedural languages. Algol W, Algol 68 and Pascal proved that certain features work, but the languages are not sufficiently accepted. If an extension exists but is not adopted, X3J3 is doing a disservice to users. Vendors do not like it but will have to accept it.

Buckley: I agree with adopting new features from other languages such as Algol 68.

Shen: I also agree with this. We could use "antique" for "deprecated". By "too large", do people mean too big to learn or too big from the point of view of compiling? Mixed language programming is not a solution. You can't necessarily trust other code and the learning effort is too large.

Mas: It is a pity to have to use other languages. There is a need for a wider Fortran language. However compatibility with other languages is important. There is a need for similarity between languages to allow bridges to PL/1, Pascal, Ada. It is important also that the same names be given to the same features in different languages.

S8 SECTIONS 1, 2 AND 3
(INTRODUCTION, FORTRAN TERMS AND CONCEPTS, LEXICAL ELEMENTS)

Reference: S8.95 Sections 1 - 3

Discussion leader: Wagener Scribe: Brainerd

Summary:

Section 1 contains the purpose, structure and scope of the standard. The syntax rules are different from Fortran 77. There is the concept of "core" which is the whole of the language other than the deprecated features. Modules are described in section 11. Intrinsic modules can be an independent standard. Some of the features now in the standard could be modules, thus reducing the size.

Discussion on section 1:

Buckley: It should be made clear that []... in the metalanguage means zero or more times.

Ampt: Should we say that the core language has potential as a teaching language?

Hirchert: X3J3 has consistently voted down any proposal to call any subset "standard-conforming".

Adams: An intrinsic module could be developed and standardized at times other than when the main Fortran Standard is processed.

Ampt: Is IRTF an intrinsic or a standard module?

Ans: If packaged as a module, it would be a standard module. If adopted by X3J3, it would be an intrinsic module.

Ampt: Then they are the same?

Ans: Yes, it is just a matter of who standardizes them.

Schonfelder: A standard module does not have to be provided by a standard-conforming processor.

Hirchert: Intrinsic module has the implication of name registration. I think we have not formally adopted this.

Schmitt: Must a module be written entirely in core Fortran?

Ans: Yes.

Johnson: This is one difference between this facility and the original idea of a language extension module.

ter Haar: The declared purpose of the standard (section 1.1) should include reliability as well as portability. The definition of "standard-conforming processor" is not restrictive enough. It should say that extensions must be flagged and there should be an option to flag core conformance.

Meek: I concur with Haar. We are willing to create a secondary standard placing additional requirements on a standard-conforming processor.

Summary:

Section 2 describes program unit concepts, statement order, data concepts.

Discussion on section 2:

Ampt: There is text describing BLOCK DATA which is not in obsolete font.

Ans: There are many editorial corrections to be made. Please point them out to us.

Ampt: Section 2.3.5 Data Type is very important, i.e. recognizing that a data type consists of a set of values, operators and representation.

Zimmer: With derived types, the difference between scalar and array is diminished. Also, there is no consistency between the different types of aggregate data. The Fortran 77 character is an aggregate data type.

Freeman: A derived data type is always scalar.

Summary:

Section 3 describes lexical elements, that is low level syntax and source form. There are nine new characters, names may have up to 31 characters and in general lower case is equivalent to upper case but there is no requirement to support lower case.

Discussion on section 3:

Ampt: There is no collating sequence defined for lower case letters. Why?

Ans: Perhaps this should be done.

Shen: What does semicolon do in:

```
if (...) then; call this
         else; call that
endif
```

Ans: It would be ambiguous in some cases if ";" were not there.

Schmidt: Brackets are special characters in national character sets. Why do you use them?

S8 SECTION 4 (DATA TYPES)

Reference: S8.95 Section 4

Discussion leader: Freeman

Scribe: Smith

Summary:

The intrinsic data types in Fortran 8X are essentially those of Fortran 77. Long and short integers have been discussed but not adopted. There is an extension to real (and complex) types to allow for generalized precision; this feature is unchanged since the Geneva meeting. A bit type has been added. This has a single bit per element and is similar to logical but has a different form of constant and has different precedence of operators. There is also the derived data type, the elevation of structures to a concept as a type with operators defined on it.

Discussion on section 4:

Schonfelder: The visibility attributes of derived data types have been clarified. The whole structure can be public, but the internals can now be private.

Meek: Specifying bit data type in this way, when character data type is specified differently, is an irregularity. Unfortunately, the irregularity is the way character is currently done. It would be worth considering strong typing. Derived data types allow scientific dimensions, such as mass, length and time to be put into languages, and allow distinctions to be made between mass and weight, as in Shen's paper (17 in the distribution).

Buckley: Derived data types are a good idea but I am concerned about parameterized data types and variant types. The former are needed for matrices of different sizes, otherwise we have the Pascal limitation for arrays. What is the cost of these?

Schonfelder: The bounds on arrays are cheap. The costs are to do with passed precision.

Hirchert: Variants are cheap to implement.

Schmidt: The description of the nature of values etc in section 4.2 is different from that in section 2. Please use the wording from section 2 throughout. The description of the numerical approximation method should be in section 4 with data types, not in section 13 with inquiry functions. Moreover there should be required to be at least two precisions for real and complex data types.

Smith: We now have at least two approximation methods.

Schmidt: There must be at least two precisions in the core, i.e. not in the deprecated features.

: Characters which may be in the character constant need to be specified, e.g. new line character.

: Fortran derived data types are better than those in Ada. Fortran does not inherit operators of the underlying type.

Ampt: Some applications need strong typing, others do not. I suggest a CONVERSION NONE statement which would prohibit implicit type conversion at assignment and the coercions for implicit types defined otherwise in the language. CONVERSION NONE together with IMPLICIT NONE would give most of the facilities of strong typing.

Straw vote: Do you approve of the functionality of CONVERSION NONE? (18-7-8)

Shen: Variants cannot be assigned except when the tag is assigned its proper value. The language should provide a way to manipulate the tag separately.

S8 SECTIONS 5 AND 6 (DATA OBJECTS)

References: S8.95 Sections 5 and 6

Discussion leader: Schonfelder Scribe: Buckley

Summary:

Since the April 1984 meeting the main change in these areas has been the addition of type BIT.

Discussion:

Shen: The ordering for arrays of arrays in structures seems wrong.

Schonfelder: There are advantages and disadvantages. Compare Pascal which has structure.component ordering and user's row-major ordering. We must keep the Fortran column-major order. This would make component%structure a natural order but X3J3 would not accept this.

Buckley: Array elements are actually subscripted separately on the component and structure fields, so why does it matter?

Schonfelder: A structure component can be identified to or passed as a multi-dimensional array. Then it matters.

Buckley: Intent IN arguments should be changeable, not constants, although their values should not be passed back.

Martin: We considered whether to be compatible with Ada or Pascal; we chose Ada.

Schonfelder: Intent IN arguments are effectively constants in the procedure, not initialized variables.

Freeman: They are not 'by value' arguments; they are 'read only'.

Hirchert: Consider arrays!

Shen: Can you initialize a local variable with the value of an IN argument, using INITIALIZE?

Schonfelder: That has not been passed but it sounds reasonable.

Schmitt: Each basic concept should be described in only one place in the document. There should be a list of special cases.

Schonfelder: There are distinct differences according to context, e.g. SAVE is slightly different as an attribute and as a statement; common blocks can be saved in the statement form.

Schmitt: ARRAY is used on page 5-1 but is not defined earlier. Is ARRAY a keyword? It is not used in an example.

Schonfelder: Note REAL(*,*) relates to generalized precision; it does not by itself declare an array.

Schmitt: I repeat, please do not duplicate definitions.

Meek: The intent IN argument should be like a symbolic constant. To say it "must not" be redefined within a procedure is insufficient. Change it to "cannot", moving the onus from the programmer to the processor.

Schonfelder: That is a stylistic comment which could apply to the whole document. We are writing for the programmer, not the implementor.

Meek: No. It is more fundamental than that. It must be absolutely clear.

Brainerd: The standard is written to tell programmers what is a standard-conforming program - not how a standard-conforming program has to be implemented.

ter Haar: How does this help reliability?

Brainerd: Just follow the rules.

Ampt: How long can we stay in the stone age and put all the onus on the programmer, not the processor?

ter Haar: Entity-oriented declarations are stated as being different. Does section 5.1 duplicate the functionality of 5.2?

Schonfelder: Section 5.1 is an extension of the type statement. It gives the ability to write more readable programs. Everything could probably be written in separate statements but this makes reading the program more difficult.

Shen: On page 5-1, "may be" implies that there exist other possible causes. Is this intended?

Vallance: I agree with Ampt that the onus should be on the processor. Also there are potential problems with spelling, e.g. INITIALIZE or -ISE.

Adams: Spelling is a non-trivial matter. It must be uniform. We will come back to that.

Freeman: Most modern languages specify the argument passing mechanisms but historically none have been specified in Fortran. It is therefore difficult to use arguments as initialized variables in some cases.

Schonfelder: We should be more specific.

Ampt: Do we need both the INITIALIZE statement and the INITIAL attribute?

Schonfelder: INITIALIZE is needed to initialize part of an object. It is very hard to do this with INITIAL.

Ampt: Are the restrictions on INITIALIZE, to do with confirming implicit typing, on page 5-11 for the benefit of a one-pass compiler?

Schonfelder: It is a carry-over from Fortran 77; it is modelled on PARAMETER.

Freeman: A problem is that implicit typing is still wanted. When INITIAL is processed, the type must be known.

Schmitt: BIT data type is not described in section 5.

Ampt: It is mentioned on page 5-1.

Schmitt: Other types are explained. Why not BIT?

Schonfelder: There is a missing paragraph.

Johnson: An intent IN argument could be allowed in INITIALIZE if it were considered as a constant.

Muenchhausen: I want character varying in the language. It can't be done in a module. An example of use is reading strings from an input device, including trailing blanks, concatenating and printing. This is easy with varying character, but hard otherwise. I would also like arrays of varying length characters.

Hirchert: What is the "actual input length"?

Muenchhausen: The number of characters typed.

Hirchert: On some operating systems that is not really defined. The record is effectively padded with an infinite number of blanks.

Muenchhausen: There is a maximum length.

Schonfelder: This is one problem you have no control over in Fortran. End of record is not defined. Otherwise varying character can be handled by a module.

Shen: You can always include apostrophe to delimit strings.

Mas: Some Fortran 77 processors can give the length of a record found in an unformatted READ.

Muenchhausen: That is not the right facility.

Vallance: On INITIALIZE, the question of one-pass compiling is irrelevant but it is bad practice to type a variable after use in earlier specifications.

Meek: Is an initial value an inherent property of an object? We should get rid of the INITIAL and PARAMETER attributes and allow variable expressions as well as constant expressions. This would clear up a number of things. The intent IN problem can be solved by declaring it to be a constant. A constant is an object that cannot be assigned to; it is different from a variable.

Schonfelder: I disagree. The point of the standard is twofold, (1) to constrain the supplier, (2) to provide the programmer with useful tools. PARAMETER statements separated from related type statements and scattered throughout a program are inconvenient.

Meek: Give it a CONSTANT attribute when declared.

Hirchert: Going back to character varying, it will not solve the I/O problem. The real problem is that not all operating systems have the concept of input records.

Schmitt: Yes, this is to do with the operating system rather than a language. There may be ten to fifteen intermediate computers between the terminal and the mainframe.

Mankowsky: Not only with I/O - there can be a problem in general in distinguishing significant data and blank characters.

Schonfelder: Varying character can be defined quite well as a derived data type, except at the moment for input/output.

Buckley: There are still parallel functionalities in section 5. I am opposed to entity-oriented declarations on the grounds that duplication is undesirable.

Schonfelder: I have tried to write it as a single facility.

Mas: There should be more examples of data types, especially to illustrate good use of Fortran data types.

Schonfelder & Adams: We agree.

Buckley: Can examples be within a standard or should they be separate, for example in an appendix?

Adams: The ANSI style manual allows examples in the text.

Shen: I repeat the need for character varying. Consider the problem of concatenating strings when their lengths are unknown. They are implementable because we did it. Algol 68 has them. If <cr> is allowed as a character there is no problem.

Freeman: Going back to examples, there must be no contra-examples in the standard.

Schonfelder: And it should be possible to remove the examples and still have a correct standard.

Wagener: I am confused about what input is intended for X3J3. There are two main issues (1) stream I/O, (2) varying length string with or without a specified maximum length. These are very different problems. I would like stream I/O, but I would vote against it at this stage.

Meek: This is a prime candidate for an incremental standard; that would be an acceptable way of doing it.

S8 SECTION 7 (EXPRESSIONS AND ASSIGNMENT)

Reference: S8.95 Section 7

Discussion leader: Smith Scribe: Schonfelder

Summary:

The structure and basis of the method of dealing with expressions and assignment was presented.

Discussion:

Schmitt: Sections 7.1.1 (Expressions) and 7.1.2 (Operations) should be interchanged. This would be easier to read.

Ampt: Shouldn't the non-terminals be defined nearer to their first use, or at least shouldn't there be a forward reference in the document?

Smith: Ideally yes, but it is difficult to achieve in practice no matter how you order the material.

Shen: Why have combination special symbol operators not been allowed for user-defined operators?

Smith: To simplify the facility and to avoid having to have a list of banned combinations which could be ambiguous, for example >- could be ambiguous with .GE.- .

Shen: Can we have recursive types and operators?

Smith: Recursive operators are allowed.

Hirchert: Recursive types are not permitted.

Vallance: Why do we have .BAND. and .BOR. for bits rather than simply use .AND. and .OR. ?

Smith: The logical operators have the wrong precedence and some expressions could become ambiguous. For example the expression:

```
          A .AND. B .EQ. C .OR. D      is evaluated as:
          A .AND.(B .EQ. C).OR. D      which is not what is required,

while     A .BAND. B .EQ. C .BOR. D    is evaluated as:
        (A .BAND. B).EQ.(C .BOR. D)    because of the precedence
defined for bit operators.
```

Meek: Truth values (LOGICAL) and binary digits (BIT) are totally distinct types. BIT is a numeric type and should use arithmetic operators.

Smith: That is a valid interpretation but not the only one. There are still problems with using + and * for BIT.

Shen: Can a user-defined operator be called .TRUE. or .FALSE. ?

Smith: At present yes but that is clearly a mistake. We must disallow this.

Buckley: Why are user operators restricted to the .<letter>. form?

Smith: If digits were allowed operators could become ambiguous with real constants with exponents.

Buckley: The irregular constraints that have to be applied to vector valued subscripts are a clear indication that the facility should be removed.

Paul: The problem lies not with vector valued subscripts. It is caused by the lack of specification of parallel assignment. Vector valued subscripts are needed for applications.

Adams: Vector valued subscripts, along with such things as variant types, keywords and optional arguments, are being suggested as candidates for removal.

ter Haar: Why is there no analogue for the ELSE IF block in the WHERE construct?

Smith: Both the WHERE and ELSEWHERE blocks are executed under the control of one logical mask. What would it mean to have a third block with another mask?

Wilson: WHERE is a masked assignment, not a control branch. Assignment is done when the mask is TRUE, not when it is FALSE. The ELSEWHERE block does the reverse under control of the same mask. You can't nest WHERE constructs.

ter Haar: We should allow things other than assignment, e.g. CALL statements in WHERE and FORALL. This would allow concurrent programming.

Smith: The purpose of FORALL is assignment.

Hirchert: WHERE and FORALL are performed in no defined order hence the content of blocks should have effects which are order independent. Subroutines necessarily have side-effects which are likely to be order dependent. We should not at this stage be writing languages to control explicitly parallel processing. We still don't know how to do this or which architectures are going to evolve.

Wagener: The expression in a FORALL statement could be a function call. This allows for the functionality albeit in a slightly clumsy way. The FORALL provides for parallel executions with barrier synchronization.

Shen: We should prohibit function calls. This will lead to inefficiency and problems of checking for side-effects.

Hirchert: The present side-effect rules cover the FORALL case and there is no requirement on the processor to check for side-effects.

Meek: More use should be made of the BNF and the use of constraints should be minimized or removed altogether.

Schonfelder: You can't remove constraints altogether for Fortran, which is so context-sensitive that a two-level grammar would be needed to specify fully the syntax in a formal metalanguage. Constraints are unavoidable with a single level BNF.

Wagener: The constraints allow a much simpler BNF. Most of the constraints that could be written in BNF would require a large number of production rules.

Hirchert: There are two classes of constraint, (1) to do with types, not expressible in BNF, and (2) such things as RETURN only in a procedure, EXIT only in a block DO. These would make for exponential growth in the size of the BNF and be more confusing than helpful.

Zimmer: Constraints could be used as in Ada, by adding a prefix to an identifier - scalar.expression or array.exprssion.

Schonfelder: Ada has a one-level grammar and pages of constraints.

Meek: As much as possible should be put in a one-level grammar.

S8 SECTION 8 (EXECUTION CONTROL)

References: Section 8 of X3J3/S8.95; Procedures & Conditions (Appendix B6); Presentation Slides (Appendix B7)

Discussion leader: Martin

Scribe: Muxworthy

Summary of ENABLE:

The ENABLE block allows for detection of circumstances when it is inappropriate to continue the normal sequence of operations. There are some intrinsic conditions and there may be user defined ones. A handler block may be provided to which control passes when a condition occurs or when a condition is explicitly SIGNALed.

Discussion on ENABLE:

Shen: Can an ENABLE block appear within a handler block? (Ans: Yes) What if the condition name is the same?

Hirchert: Such names are local and nesting is consistent. Control always passes to local handlers, i.e. one level out.

Martin: There is no crossing of subprogram boundaries.

Ampt: Can SIGNAL be part of a logical IF?

Martin: Yes, SIGNAL should be in the list of statements in section 2.

Meek: I welcome this subject being addressed in the standard.

Buckley: This section was the most difficult to read. A list of intrinsic conditions is needed.

Hirchert: The section 12 you have, with appended section on conditions, is not the latest. That will made available later in the meeting (paper 27, Appendix B6 to these minutes).

Summary of DO:

There are currently two looping constructs, the Fortran 77 DO which is deprecated and the DO-REPEAT; EXIT and CYCLE apply to both. There is a proposal to combine them into one construct: this would have the label optional in the DO and allow END DO or CONTINUE or the other Fortran 77 possibilities as the loop terminator. Deprecated would be the last of these three, non-integer DO-variables and multiple DOs sharing the same termination statement. However 8X would then have block constructs for selection but not for looping, except by programmer choice, and it would leave 9X with a block construct that can depend on a label. Moreover this still would not resolve the classic DO 10 I=1.10 (point for comma) problem, so this solution would make 8X short on both safety and regularity.

An alternative proposal would be to retain the Fortran 77 DO, add END DO to it, deprecate three or four features and introduce a separate loop construct, called for example LOOP-END LOOP. (This was not introduced until after the second straw vote on the first proposal).

Discussion on DO:

Straw Vote: Should the Fortran 77 DO and the proposed DO-REPEAT be merged into one construct? (10-2-19)

Martin: I would welcome comment on possible deprecation of labels in the DO and on the terminator statement.

Schonfelder: ENDIF can be jumped to; this is similar.

Shen: I favour deprecation of labels so long as EXIT can apply to CASE and block IF.

Martin: That has been voted down in X3J3.

Ampt: You should ask this group.

Brainerd: CONTINUE should be deprecated.

Martin: That would put 9X back with two separate constructs.

Hirchert: Can two DO loops end on the same END DO if labels are used? (Ans: No) There are still problems. The BNF does not allow for this currently.

Adams: We want the extension to DO to extend the life of current software.

Straw Vote: Should labels in the DO statement be deprecated? (25-0-8)

Martin: Another possibility is to rename DO-REPEAT to be LOOP-END LOOP, retain the Fortran 77 DO and add END DO to that.

Marshall: Why not simply deprecate the Fortran 77 DO?

Ampt: You are making people rewrite programs.

Adams: There has been criticism of 8X because of this enforced rewriting.

Ampt: But the deprecated features give an overlap period.

Adams: Some users say deprecated features will not be allowed at their installations.

Brainerd: What is the problem?

Adams: The standard will not be accepted, or people will use Ada.

Paul: They may move to Ada but they'll be back within a day.

Pollicini: I see no problems with deprecating the Fortran 77 DO.

Schonfelder: I suggest deprecating Fortran 77 DO and changing DO-REPEAT to LOOP-END LOOP.

Straw Vote: Do you prefer (a) two separate constructs with one of them deprecated, or (b) one collapsed construct, or (c) are you undecided? (19-6-8)

Straw Vote: Given two separate constructs, do you prefer (a) DO-REPEAT, or (b) LOOP-END LOOP, or (c) are you undecided? (6-17-10)

Shen: A WHILE clause should also be in the loop control.

Schonfelder: The functionality is already there. WHILE is useless because of the need to initialize variables before entering the loop. UNTIL would be better.

Straw Vote: Should WHILE be in the loop control? (1-26-4)

A NEW LOOPING PROPOSAL

References: Do loops (Appendix B8 to these minutes)

Discussion leader: Wagener   Scribe: Muxworthy

Summary of DO:

> At Fortran Forums one of the most frequently raised items, and it is a sore point with users, is to do with the deprecation of the current DO. Therefore we must have a new DO as an extension of the old DO to avoid problems. This proposal allows both old and new forms, will satisfy users and will help get the standard adopted.

Discussion:

Adams: The mood in X3J3 is to compromise to satisfy opponents of 8X. Andy Johnson is chairman of a subcommittee to investigate deleting incremental features and reducing deprecated features.

Johnson: We have found from a wide-ranging questionnaire that there are a few hot-spots to address. If the old and new can co-exist they should be left alone; we should deprecate only if the old gets in the way of the new facility. We decided we should pay more attention to compatibility to pacify users.

Adams: These proposals will be made to X3J3 next week.

Ampt: When X3J3 first came to Europe they were surprised at the good European reaction. Do not listen to conservatives - improve the standard.

Schmitt: Is Fortran 77 to be used until 2010?

Meek: Why make these proposals to X3J3 before WG5? Remember ISO/TC97/SC22 represents users too. Remember WG5 has given support to X3J3 over the years.

Adams: X3J3 does the technical work, WG5's role is review and advisory.

Brainerd: X3J3 overreacts to user comments. I have more confidence in X3J3 and WG5 than in users; they have lived with these proposals longer.