

X3J3 PREMEETING DISTRIBUTION  
 105th Meeting, X3J3  
 Liverpool, England  
 August 10 - 14, 1987

					PAGE
				Index .....	1
				Agenda .....	3
1	105	(*)	JCA-1	Between Meetings Letter .....	5
2	105	(*)	JCA-2	Part I of the X3 Submittal .....	7
3	105	(*)	JCA-3	Part II of the X3 Submittal, X3JE & WG5 Responses .....	24
4	105	(*)	JCA-4	Glossary Proposal from X3K5 .....	177
5	105	(*)	JCA-5	Secretary Appointment to SMC .....	193
6	105	(*)	JCA-6	Master Plan SD-1, May 1987, Order Form .....	195
7	105	(*)	JCA-7	News Release on C Language Binding to C .....	197
8	105	(*)	JCA-8	Century in SQL2 .....	199
9	105	(*)	JCA-9	Van Snyder Letter and Adams Acknowledgment ..	201
10	105	(*)	JCA-10	SC22 Tag Appointment .....	203
11	105	(*)	JCA-11	Letter to W. Rinehuls of SPARC .....	205
12	105	(*)	JCA-12	Johnson Appointed IR for X3J3 .....	207
13	105	(*)	JCA-13	Boeing Opinion to X3 .....	209
14	105	(*)	JCA-14	New Fees .....	211
15	105	(*)	JCA-15	KANJI Documents .....	213
16	105	(*)	JCA-16	News Release, FORTH Language .....	217
17	105	(*)	JCA-17	New TC97 SC on Graphics .....	219
18	105	(*)	JCA-18	Preliminary SQL Results .....	221
19	105	(*)	RCA/COB-1	Remove the EXPONENT LETTER Statement .....	239
20	105	(*)	COB-2	Some Notes on the Implementation of an Intrinsic BIT Module .....	241
21	105	(*)	COB-3	Some Pseudorandom Thoughts on a Pointer Facility	243
22	105	(*)	COB-4	Unbuffered Bit I/O .....	247
23	105	(*)	TAH-1	D.G. Customer Comments (FYI) .....	251
24	105	(16)	JHM-1	Derived Type I/O .....	255
25	105	(16)	JHM-2	KANJI Type .....	259
26	105	(*)	JKR-1	Meeting Minutes .....	263
27	105	(*)	JKR-2	X3J3 Meeting in Seattle .....	269
28	105	(*)	JLW-1	Standing Document on Membership Policies .....	273
29	105	(*)	JLW-2	Responses to WG5 Resolutions .....	275



**Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\***

Jerrold L. Wagener  
Amoco Production Research  
P. O. Box 3385  
Tulsa, OK 74102  
(918) 660-3978

**NOTICE:** X3J3 Meeting Number 105

**TIME:** August 10-14, 1987, 8:15am TO 6:00pm each day  
Agenda on reverse side

**PLACE:** Faculty of Education Conference Room  
Abercromby Square  
University of Liverpool  
Liverpool, England

**HOST:** Lawrie Schonfelder  
Computer Laboratory  
University of Liverpool  
Liverpool L69 3BX  
England  
(44)-(51)-709-6022 x2954

**HOTEL:** The Britannia Adelphi Hotel  
Ranelagh Place  
Liverpool L3 5UL  
England  
(44)-(51)-709-7200

**REGISTRATION FEE:** 45 pounds, charged to all attendees

Monday, August 10, 1987

8:15-12:15 Full Committee: Opening Ousiness (J. Adams)  
WG5 Liverpool Resolutions (J. Martin)  
X3J3 Organization (J. Adams)  
Assignments for Public Review (J. Adams)  
Preparing Comment Responses (J. Adams)

1:15 Subgroup Meetings

Tuesday, August 11, 1987

8:15-12:15 Full Committee: S8 Audit Against Fortran 77 (J. Adams)  
Fortran 77 Interpretations (E. Johnson)  
Standing Document on Membership (J. Wagener)

1:15 Subgroup Meetings

Wednesday, August 12, 1987

8:15-12:15 Full Committee: Tutorial on Kanji (K. Harris)  
Tutorial on Pointers (C. Burch)

1:15 Subgroup Meetings

Thursday, August 13, 1987

8:15-12:15 Full Committee: Derived Type I/O (J. Matheny)  
Binary I/O (C. Burch)

1:15 Subgroup Meetings  
4:00 Subgroup Heads Meeting

Friday, August 14, 1987

8:15-12:15 Full Committee: Topics from S9 (J. Matheny)  
Varying Character Module (J. Wagener)

1:15 Full Committee: Closing Business (J. Adams)  
Other Business

1

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH  
Scientific Computing Division/Advanced Methods Section  
P. O. Box 3000 • Boulder, Colorado • 80307  
Telephone: (303) 497-1275 • FTS: 320-1275 • Telex: 45 694

105(\*)JCA-1

MEMO TO: X3J3  
FROM: Jeanne Adams, Chair *Jeanne Adams*  
DATE: June 1987  
SUBJECT: Between Meetings Letter

The Liverpool meeting will not be as fast-paced as our recent meetings, when we debated forwarding S8 as the draft standard. The next meeting will catch up on topics like interpretations and how the Japanese request for Kanji can be solved. A solution would be an excellent contribution from X3J3.

Many of you have said that there are more editorial comments from X3J3 for the formal review. I have a suggestion. Members might prepare a list that the editorial committee could review, and then one person, perhaps Lloyd, could submit one public review statement with these comments. It might make it simpler to have one editorial document from X3J3. Only those that X3J3 passes need be registered in a formal comment. These comments could be handled now while the editorial processing we have just completed is fresh in our minds. However, if any of you have problems with this, we won't pursue it further. Editorial suggestions would be processed by majority, until the list returns from the X3 registration of formal public review comments. Any topic that X3J3 wishes to process collectively could of course use the same procedure.

I mentioned a possible reorganization of the work. First, there is consideration of the jobs to be done. The following are possible items:

- Processing the Public Review Answers
- Processing any Editorial Material
- Interpretations and Clarifications
- Writing Answers for Comments X3J3 Rejects
- Writing Proposals for Comments X3J3 Accepts and the Resulting Answer
- Planning for Any Intrinsic Modules in the Fortran Family of Standards

There should be a committee responsible for collating and coordinating public review answers. This can be a big job and one that, if done well, will save time and energy. There should be a group working on interpretations to complete the backlog. Members who have just had responsibility for the various S8 sections should continue while answers to the public review are generated. They have the background to answer comments in these sections. You might

5

identify the group where you would make the best contribution. No more than six groups should be identified, with least 5 members in each group. I will have a plan to present at the next meeting.

A bound copy of Part I and Part II of the submission package to X3 was completed and sent to Patti Steiner at CBEMA. I have also placed a copy of the submission package in the pre-meeting distribution, as JCA-2 and JCA-3. Jerry has received the final typesetting of S8 from Walt. It was checked by John Reid and Brian as well as Jerry. I expect the final draft revision to be in the mail to Patti in a few days. My hope is that the submission will be an agenda item at the next SPARC meeting in July. I want to thank Walt, Lloyd and Jerry for all their efforts in typesetting and mailing S8. We would not have a draft standard without their efforts, and the work of Lloyd and others who prepared the draft sent to Walt.

There will be an informal distribution to X3J3 of just S8 from Jerry. This distribution to X3J3 also includes members of WG5 who follow our work. The informal distribution will be the last one that does not have a fee, according to the policies of CBEMA and X3 on draft standards. Interested persons must then request copies from CBEMA. I will have the phone numbers that should be used to order a copy of the draft at the next meeting. Our internal documents will of course continue to be processed as usual, and that includes any processing of the document referred to as "S12".

I would like to remind those of you who have not yet mailed their letter ballot on SQL to me to send it as soon as possible. Even though it may be late, you should send it, since this requires a 2/3 vote and one more than half. I don't have even close to half as yet.

I hope you are having a pleasant and restful summer vacation, after the hard work of the past year on Fortran 8x.

X3J3

**X3J3 BALLOT RESULTS**

**SUBMISSION TO X3  
DOCUMENT NUMBER 207**

Part I

**X3J3 DRAFT FORTRAN STANDARD**  
 Submittal Document  
**TABLE OF CONTENTS**

Part I	
Public Review Transmittal Form . . . . .	1
Submission Memo . . . . .	2
Project Proposal Fortran . . . . .	5
X3J3 History . . . . .	10
Development Phases . . . . .	11
Abstract . . . . .	12
 Part II X3J3 Ballot Results	
Table of X3J3 Ballots . . . . .	16
Allison, Robert . . . . .	17
Response . . . . .	20
Ellis, T. M. R. . . . .	22
Response . . . . .	24
Freeman, Murray . . . . .	25
Response . . . . .	27
Harris, Kevin . . . . .	28
Response . . . . .	34
Hendrickson, Richard A. . . . .	38
Response . . . . .	42
Hirschert, Kurt W. . . . .	45
Response . . . . .	47
Hoover, Tracy . . . . .	48
Response . . . . .	50
Johnson, E. Andrew . . . . .	51
Response . . . . .	53
Lakhwara, Anil . . . . .	54
Response . . . . .	59
Martin, Bruce A. . . . .	62
Response . . . . .	64
Marusak, Alex . . . . .	65
Response . . . . .	68
Moss, Leonard J. . . . .	70
Response . . . . .	74
Philips, Ivor . . . . .	75
Response . . . . .	87
Ragan, Richard . . . . .	88
Response . . . . .	90
Reid, J. K. . . . .	91
Response . . . . .	92
Bowe, V./Rolison, Lawrence . . . . .	93
Response . . . . .	97
Schonfelder, Lawrie . . . . .	100



Response . . . . .	102
Smith, Brian T. . . . .	103
Response . . . . .	105
Swift, Richard C. . . . .	106
Response . . . . .	107
IBM (Richard Weaver) . . . . .	108
Response . . . . .	115
Part II WG5 Ballots	
Memo-X3J3 Responses to WG5 Ballots . . . . .	119
Table of WG5 Ballots . . . . .	120
Bourstin . . . . .	121
Response . . . . .	122
Hammerling, S. J. . . . .	123
Response . . . . .	125
Kan, Tadayoshi . . . . .	126
Response . . . . .	131
Kneis, Wilfried . . . . .	132
Response . . . . .	133
Meek, B. L. . . . .	134
Response . . . . .	136
Meier, Bruno . . . . .	137
Response . . . . .	138
Muxworthy, David . . . . .	139
Response . . . . .	141
Schmitt, Gerhard . . . . .	142
Response . . . . .	143
Schonauer, Willi . . . . .	144
Response . . . . .	145
Shen, M. K. . . . .	146
Response . . . . .	148
Vallance, D. M. . . . .	150
Response . . . . .	152
Wilson, John D. . . . .	153
Response . . . . .	156
Country Votes from WG5	
Canada . . . . .	158
Response . . . . .	161
Austria . . . . .	162
West Germany . . . . .	163
United Kingdom . . . . .	164
Response . . . . .	166

# BSR PUBLIC REVIEW TRANSMITTAL FORM

FOR  
SECRET  
ACTION

X3TC X3J3

X3 Project # 67-R Technical Committee Doc. # X3J3/S8.104

Formal title as well as "nicknames" which might be used:

Fortran

Fortran BX

Results of TC LB to Forward for Further Processing: 29/7/0  
(Attach details) Yes/No/Non Response

Document Content Checklist -- Have you included:

- Foreword? X
- Name, address and phone number of Technical Editor? X
- Technical Committee List & names of Individual Experts who have contributed? (This is the list which will appear in the approved standard) X
- Expository Remarks? X

-----  
Preferred quote for Public Review Press Release -- geared toward enticing User interest in document:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

If you have additional press contacts you would like notified of the availability of the document, please include their names and addresses below (use separate sheet if necessary):

_____	_____
_____	_____
_____	_____
_____	_____

Attachment: SD-3 which authorized this document

10/86

10

MEMO TO: X3

DATE: June 15, 1987

FROM: *Jeanne Adams*  
Jeanne Adams, X3J3 Chair

SUBJECT: Submission

**X3J3 LETTER BALLOT**

December 1, 1986-January 5, 1987

Draft Revision of X3.9-1978 (Fortran 8x)

X3J3 has processed the ballot commentary of the X3J3 letter ballot and the TC97/SC22/WG5 ballot. The negative and affirmative with comments ballots are enclosed. Responses voted by full committee are included with the negative ballots, as well as with the affirmative ballots that contain remarks of a general nature. The editorial comments included with ballots have been considered by X3J3 and either accepted or rejected; these comments have not been included.

X3J3 requests a compliance review by SPARC. After this review, X3J3 requests that a public review be initiated. At the same time, since the WG5 ballot was affirmative, X3J3 requests that the document be submitted to SC22 members for a review concurrent with this public review. This is also a request to enter the draft revision for Fortran into the ISO arena.

The technical editor is:

Lloyd Campbell  
618 Southgate Road  
Aberdeen, MD 21001  
Telephone (301) 272-3771

The role call vote at the X3J3 meeting in Seattle took place after the ballot commentary had been processed. One member who registered a negative vote on the letter ballot changed his vote to affirmative. Of the nine negative votes, three changed an affirmative vote to negative. One vote was registered by a new member appointed at the Seattle meeting, and the remaining were votes that confirmed a negative vote on the December letter ballot. One member who registered a negative vote on the letter ballot was absent. Statements explaining negative votes were taken at this meeting and are a matter of record. Copies are available on request.

The two negative votes on the TC97/SC22/WG5 ballot requested additional features in the language. This submission memo concludes with a table of voting summaries.

X3J3 has reached the best consensus possible at this point; the language features described in the draft standard are a recommendation for the revision of X3.9-1978. There is considerable diversity of opinion among members of the committee as evidenced by the ballot responses. For example, approximately half the negative votes want fewer features, and half want additional features in Fortran. Thus, we have cross currents of disagreement but these do not show a consistent direction.

The committee intends to continue work toward resolving many of these issues at future meetings. X3J3 feels that the public should have the opportunity to comment on its past work and its current status. I believe that a formal public review under X3 and in the ISO arena is important at this time and will be of considerable help to the committee as it works toward the goal of the best possible standard for Fortran.

VOTES TAKEN

SUMMARY OF X3J3 LETTER BALLOT

Vote	Number
Affirmative	29
Negative with Reasons	7
No Ballot Received	0

SUMMARY OF WG5 LETTER BALLOT

Vote	Number
Affirmative	25
Negative with Reasons	2

VOTE SUMMARY OF X3J3 ROLL CALL VOTE  
May 14, 1987 in Bellevue, Washington

Vote	Number
Affirmative	26
Negative	9
Absent	3

# PROJECT PROPOSAL

## FORTRAN

PROPOSAL FOR CONTINUATION OF THE  
X3 STANDARDS PROJECT ON FORTRAN

1.0 Identification

- 1.1 Title: Support of and revision to X3.9-1978 Programming Language FORTRAN
- 1.2 Proposer: Jeanne Adams, Chair, X3J3  
National Center for Atmospheric Research  
P.O. Box 3000  
Boulder, Colorado 80307
- 1.3 Date: April, 1978

2.0 Description

2.1 Purpose

The purpose of the standard is to promote portability of FORTRAN programs for use on a variety of data processing systems.

2.2 Nature of the Standard

The standard is a document that specifies the form and establishes the interpretation of programs expressed in the FORTRAN language. The document will be a revision of American National Standard Programming Language FORTRAN, X3.9-1978.

2.3 Scope

1. Provide a mechanism for the solicitation and review of all activities regarding American National Standard Programming Language FORTRAN.
2. Carry out procedures to maintain the continuous responsiveness of the standard to industry needs.
3. Act as a liaison group with other organizations interested in the standardization of FORTRAN.
4. Augment the American National Standards FORTRAN (X3.9) with a comprehensive data base facility using the CODASYL FORTRAN JOD wherever consistent with the general interface features being proposed and wherever appropriate for inclusion in the language.

2.4 Program of work

1. Provide support, interpretation, and maintenance of X3.9-1978, Programming Language FORTRAN.
2. Continue pursuing the work required for the international standardization of FORTRAN.
3. Establish the criteria to be applied in evaluation proposals for the revision of X3.9.
4. Prepare a draft proposed revision of X3.9 for public review and comment.
5. Maintain a close working relationship with other national and international groups interested in FORTRAN standardization such as the CODASYL FORTRAN group.

6. Maintain a close working relationship with groups active in the creation of collateral standards affecting FORTRAN.
7. Keep the public informed of the direction and activities of the committee in order to encourage feedback.

### 3.0 Expected Benefits

#### 3.1 Intrinsic

The major intrinsic benefit of a FORTRAN standard is to minimize or eliminate potential incompatibilities in different implementations of FORTRAN language processors.

#### 3.2 Interchange

The standard provides for a degree of portability of both programs and programmers between different computers and computer operating systems.

#### 3.3 Educational

The standard provides a language definition that can be used by both authors of FORTRAN documents and teachers of FORTRAN at all levels. Additionally, textbooks, reference manuals, and other documents describing FORTRAN will become more consistent in their terminology and description of the language, thus reducing retraining costs.

#### 3.4 Economics

Interchangeability of programs and programmers has proven to be of major economic advantage within the industry. The continued maintenance and revision of the standard protects the investment in already existing FORTRAN programs and FORTRAN language processors.

### 4.0 Feasibility of Development

#### 4.1 State of the Art

During development of a new revision to X3.9, it is likely that augmentations will mainly be drawn from functionality that exists in advanced implementations of existing processors. Therefore, development of a new revision of X3.9 is well within the state of the art. During the period of development of X3.9-1978 and during the public comment period on BSR X3.9, a very large body of public comment was received. Many proposed extensions were found to have merit but were not included either because of limited resources available to the committee or because their inclusion would have unduly delayed the issuance of the revision. It is, therefore, highly timely that those proposed extensions be examined for possible inclusion in a subsequent revision.



#### 4.2 Resources

X3J3 has been able to attract sufficient working membership to complete the first revision of the standard. New members have joined the committee since the adoption of the revised standard. There is every indication that the technical committee will have sufficient resources to complete the Program of Work

#### 4.3 Cost

The cost of developing X3.9 has, for the most part, been borne by the sponsors of the membership of X3J3. The committee members and their employers are aware of the expense. The committee intends to meet five times per year for four days. Expected average attendance is 30 persons.

### 5.0 Feasibility of Implementation

#### 5.1 Supplier Conformance Consideration

In developing the present FORTRAN standard a great deal of effort was expended in describing the language so that it could be interpreted unambiguously by FORTRAN implementors. A large body of public comment confirmed the belief that the existing form of the standard was readily understandable by suppliers. No incompatibilities with emerging technology are foreseen.

#### 5.2 User Operational Considerations

Once a language standard is in force and there are conforming programs, incompatible change, even a small change, impacts the user community. Many of the members of X3J3 are users and additionally, X3J3 has been in continuous contact with various user groups. Any incompatible change to the existing standard would be accepted only after very careful consideration.

#### 5.3 Cost Considerations

One of FORTRAN's most important characteristics is that efficient processors can be implemented at a reasonable cost. One of the most important goals during the next revision will be to retain this characteristic.

### 6.0 Maintenance

#### 6.1 Extent and Frequency of Anticipated Changes

X3J3 intends to provide interpretation and clarifications of X3.9-1978 as the need arises. The committee also intends to comply with the requirement that X3.9-1978 be reviewed within a five year period.

#### 6.2 Resources

The committee accepts its responsibility to maintain X3.9-1978 and to continue this activity along with its revision efforts.

#### 6.3 Cost

The cost of maintaining the standard is estimated at 10% of the five-year development cost.

7.0 Related Standards Activities

Related standards activities are conducted by other X3J3 committees, X3L5, the CODASYL FORTRAN DML Committee, the Purdue Workshop, ECMA TC8 and TC22, IFIP Working Group 2.5-Numerical Software, and ISO TC97/SC5.

8.0 Recommended Time Frame for Revision of X3.9

- |      |     |  |
|------|-----|--|
| 1978 | Jan | Establish procedures within X3J3 for processing proposals for revision of X3.9 and begin research on the identification of problem areas that will generate revisions to X3.9. |
| 1979 | Jan | Begin processing proposals for revision of X3.9.   |
| 1982 | Jan | Submit draft proposed revision of X3.9 to X3 for public review and comment.  |
| 1983 | Jan | Publish X3.9-1983.   |

# X3J3 HISTORY

# DEVELOPMENT PHASES X3J3 MEETINGS 1977-1986

1977

October ---> OBJECTIVES

1978

March ---> TUTORIALS

April X3.9-1978

1979

January ---> PROPOSALS

1980

August ---> S6 NEW FEATURES

1981

1982

January ---> S7 F77 + S6

1983

1984

May ---> S8 DRAFT

1985

1986

April ---> FIRST BALLOT

REVISION  
PROCESSING

1986

December ---> SECOND BALLOT

**ABSTRACT**  
**HISTORY OF X3J3 MEETINGS**  
**1977-1987**  
Jeanne Adams, Chair, X3J3

**1977**

1977 marked the conclusion of the public review of Fortran 77 and the conclusion of the responses by the Fortran Standards Committee. A subcommittee was assigned to study the scope and the plans for the future of Fortran. A vote was taken to remain as a 'Committee of the Whole'. The committee was resolved to work with international organizations and the Fortran Experts Group as the draft standard was developed.

**1978**

X3.9-1978 was approved by BSR. A project proposal was prepared for the next Fortran Standard. SC5 approved Fortran 77 for processing as an International Standard. User needs were surveyed. A Core-Plus-Modules approach to Fortran was adopted. A Tutorial Program was initiated to conclude in January 1979. An international meeting was scheduled. Issues and clarifications of Fortran 77 were processed.

**1979**

A Core Foundation Document was passed. Task groups were formed to prepare a CODASYL Data Base Standard for Fortran and a standard for Industrial Real Time Fortran. Tutorials continue and the first proposals are processed. The development cycle for Fortran 8x begins. A tutorial on Open Systems was presented. Discussions conclude that too many new features are proposed.

**1980**

Procedures for Task Groups in X3J3 were formulated. Further attempts were made to place boundaries on new features. A list of allowable technical material was approved. A cutoff date for new material was established. Plans were initiated for the Proposal Document (S6) which was completed and refined. ISO 1529 was accepted as an International Standard.

**1981**

S6 was formally adopted by X3J3. Plans for a "Fortran Family of Standards" were presented to X3J3 and SPARC. Financial problems threaten the committee and cost cutting methods are discussed. Meetings are limited to four in a year and the milestone charts advance the completion date to 1986. Refinements of major proposals in S6 are the basis of most of the technical proposals processed. A high priority item for X3J3 is the publication of a Fortran Information Bulletin on Clarifications of Fortran 77. A tutorial was held on keeping Fortran a 'naive and universal' language.

## 1982

The preparation of the draft standard was begun by combining S6, the proposal document, and the Fortran 77 standard. This document (S7) is to be processed by UNICOMP, in Los Alamos, New Mexico. Critical issues were identified for the preparation of S7. A Fortran Information Bulletin was planned summarizing the work toward the Fortran 8x standard. A first draft of S7 was completed.

## 1983

S7 was reviewed from the point of view of the "ANSI Style Manual". S7 was formally accepted and S6 declared a historical document. A request was sent to X3 to withdraw the data base project on CODASYL. ISO was notified that a draft standard was in the final writing stage. After work progressed on S7 for several meetings, the committee concluded that it needed a basic reorganization from the one based on Fortran 77. After study for several meetings, a reorganization of S7 was begun. This decision was based on the fact that S7 did not present a clear picture of data types (including programmer defined ones, data objects and the relationship of data to an executing program. In spite of the fact that X3J3 knew that this would significantly increase the time needed to complete Fortran 8x, the committee felt that S7 as it stood was unacceptable. Therefore the development of S8, the reorganization of S7, was undertaken.

## 1984

A Four Meeting Plan was initiated that controls the critical issues needing processing, and establishes longer term committee goals. Project 318 was withdrawn by X3. The Fortran Information Bulletin summarizing Fortran 8x was published. Responses were prepared for the negative votes of IBM and DEC on the Fortran Information Bulletin. Attempts were made to decrease the size of the language; at each meeting candidates for deletion from the new features list were considered. Those that would be acceptable for deletion would not impact the size by much. A completed version of S8 was distributed. Because there were several authors, it became apparent that much editorial work was needed. A Fortran Forums Program was initiated and a user questionnaire was circulated on the new features in Fortran 8x.

## 1985

S8 was accepted as a base document and S7 became a historical document. The meeting schedule was expanded back to five meetings a year. November 1985 was the goal for forwarding S8 to SPARC for compliance review and balloting on S8 within X3J3. The vote to forward failed, but a straw vote indicated that a vote on S8 at the January 1986 meeting would pass.

## 1986

In January, X3J3 voted to conduct a letter ballot during March. The document was "frozen" except for editorial fixes and issues to fill holes and repair inconsistencies.

The ballot failed (16-20-1-0). Some of the ballots suggested that the language was too large, others did not agree with the style and editorial presentation of the document. Requested deletions were all quite different and it was apparent that further work must be done to resolve issues on the language content. The next three meetings were spent discussing deletions to the language and a reduced language was eventually adopted.

The reduced language was presented to the WG5 meeting in Halifax, where other suggestions to improve the language content were presented. A December ballot was conducted by X3J3 and by WG5. Both ballots passed. There were many editorial comments presented with affirmative votes, as well as with negative votes.

### 1987

An extension to the February meeting was called for March in Albuquerque to complete processing ballot comments. After the meeting in Albuquerque, all ballot comments had been processed by the full committee. A corrected document was prepared as the draft Fortran standard. The following reviews are requested by X3J3:

1. A compliance review by SPARC
2. Public Review approval by X3
3. Submission to SC22 for concurrent public review with X3

The documents enclosed were prepared for the above review requests.

3

X3J3

# X3J3 BALLOT RESULTS

# X3J3 BALLOT RESPONSES DOCUMENT NUMBER 207

## Part II



TABLE OF X3J3 BALLOTS

NAME	VOTE	Role Call
1. Adamczyk, J. Stephen	Aff	Neg
2. Adams, Jeanne C.	Aff	Aff
3. Barber, Graham	Aff	Resigned
4. Bowe, Valerie	Neg	Neg
5. Brainerd, Walter	Aff	Aff
6. Burch, Carl D.	Aff	Aff
7. Campbell, Lloyd	Aff	Aff
8. Crowley, Ted	Aff	Absent
9. Allison, Robert	Aff	Neg
10. Ellis, T. Miles	Aff	Aff
11. Freeman, Murray	Aff	Absent
12. Gridley, Kurt	*	Aff, New Member
13. Harris, Kevin	Neg	Neg
14. Hendrickson, Richard	Neg	Aff
15. Hirschert, Kurt W.	Aff	Aff
16. Hoover, Tracy Ann	Aff	Aff
17. Johnson, Andrew	Aff	Aff
18. Lakhwara, Anil	Neg	Absent
19. Marshall, Neldon	Aff	Aff
20. Martin, Bruce	Aff	Aff
21. Martin, Jeanne	Aff	Aff
22. Marusak, Alex	Aff	Neg
23. Matheny, James	Aff	Aff
24. Metcalf, Michael	Aff	Aff
25. Millard, G. G.	Aff	Aff
26. Moss, Leonard	Neg	Neg
27. Philamore, David	*	Aff, New Member
28. Phillips, Ivor	Neg	Neg
29. Ragan, Richard	Aff	Aff
30. Reid, J. K.	Aff	Aff
31. Schenk, Werner	Aff	Aff
32. Schonfelder, J. L.	Aff	Aff
33. Smith, Brian	Aff	Aff
34. Swift, Richard C.	Aff	Aff
35. Thompson, Brian	Aff	Aff
36. Wagener, Jerrold	Aff	Aff
37. Wearing, Alison	*	Neg, New Member
38. IBM, Weaver, Richard	Neg	Neg
39. Wilson, Alan	Aff	Aff

# X3 Subgroup<sup>-17-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

*See Attached*

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Bcx 3000  
Boulder, CO 80307

NAME ROBERT CARL ALLISON  
(PLEASE PRINT)

SIGNATURE *Robert Carl Allison*

DATE: 12/24/86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

9.0

*26*

Harris is marginally in favor of submitting the draft to X3 for further processing. We have some serious misgivings about the draft, but are willing to see if a public review indicates that the public has similar misgivings. There are some general comments about a few features and we have attached some editorial comments.

SET RANGE is insufficiently defined to be easily understood (if at all). For example, if a module defines a variable A with attribute RANGE but not SAVE, and the variable is USED in procedure B which performs a SET RANGE, and then calls procedure C which also USES variable A, it seems pretty clear that the effective range from procedure B is in effect. And if procedure C also performs a SET RANGE and then exits it seems pretty clear that procedure B is affected by the effective range declared in C. But, if B is called from some other procedure which does not USE A, and B performs a SET RANGE and then exits, and then procedure C is called, does it receive the effective range defined in B? It would seem so, since according to the sentence on page C-11, lines 22-24, the definition only goes away when no one is using the module. This requires an elaborate implementation scheme, which is both slow and large.

The major problem in figuring out SET RANGE is reconciling the statements in the sections on scoping with the description of modules. The section notes for Section 10 give a backhand indication of how the definition status of variables in modules become undefined, but it has nothing to do with local or global scope. Yet, the definition of when effective range reverts back to declared range is in terms of when a variable that is local to a scope becomes undefined (See page 6-3, lines 26-28).

I would be happy to submit a proposal to clarify the intent of these features if only I could figure out what the intent was.

Also, the implications of allowing intrinsic functions with constant expressions as arguments to be constant expressions are far reaching. In general, this requires the compiler to now include the entire runtime library so that it can perform any of the intrinsic functions. As a specific example, cross-compilers are going to have a terrible time doing trigonometric functions to an accuracy which is greater than the host machine's. (Imagine the unusual case of a cross-compiler from a micro or PC to a mainframe. It must emulate floating-point sizes two to four times its floating point size). In terms of compiler size (it is becoming increasingly difficult to imagine how this current draft standard will fit on a micro) and compiler speed (software emulation of a trigonometric function at software emulated precisions), this seems to be an expensive addition for limited additional functionality.

To: X3J3  
From: Bob Allison  
Subject: Deleting the Case-Value Range of the form ":"  
Date December 24, 1986

Once again, as we did on the last ballot, we would like to state that allowing a case-value range of the form ":" as well as DEFAULT is excessive, and would like to propose removal of that selector:

Make the following changes to S8, February 1987:

Page 8-3, line 36, replace with the following two lines:

"or [ case-value ] : case value"

"or case value : [ case value ]"

Page 8-4, lines 18-20, delete.

9.2

28

## RESPONSE TO ROBERT ALLISON BALLOT

There were 12 editorial comments in your ballot, all of which have been addressed by full committee as changes to the draft, or considered and rejected. You requested comment 1) on the definition of SETRANGE and 2) on certain implications of intrinsic functions with constant expressions.

### 1) SET RANGE

You are correct. SET RANGE as originally specified in S8 required an elaborate and slow implementation scheme to reset the effective ranges to the declared ranges. S8 has been changed to cause the effective range to revert to undefined where it previously reverted to the declared range. This makes SET RANGE work much as COMMON works in FORTRAN 77 and permits efficient implementations without undue burden on the user or the implementor.

### 2) Intrinsic Functions with Constant Expressions

The X3J3 committee considered the concerns expressed in the above ballot comments but felt no changes were required for the following reasons.

FORTRAN 77 currently permits constant expressions to include all of the intrinsic arithmetic, character, and logical operators with constants or constant expressions as operands, except for the exponential operator \*\* which is limited to integer powers. Therefore, cross compilers even for FORTRAN 77 currently must address the issue that the operands may have a very different representation on the host than on the destination machine.

On the other hand, FORTRAN 77 does not require that such expressions be evaluated at compile-time but can be postponed to runtime. Whether it is an advantage or disadvantage to do so depends upon the machine and the properties of the object files and the linker/loader.

The issue, therefore, is not one of precedent but of the number of operations (including those embodied in intrinsic functions) that must be supported in what appears to many to be compile-time expressions. Several models of how this can be implemented are readily apparent. Which one is sensible will depend upon the machine and also on the tradeoffs between storage and execution and on whether the evaluation of such expressions is borne at compile-time, linker/loader-time, or execution-time. (One model that addresses the particular example given in the ballot comment RA 9.1 is one where the expressions are evaluated just prior to the execution of the main program and the values inserted into the appropriate places in the load image.)

With the determination of execution-time determined extents for arrays, the need for arbitrary constant expressions in specification contexts becomes apparent. However,

the use of the entire intrinsic function set appears at first to be excessive. When attempting to formulate a rationale for including some but not all of these functions, it became clear to the committee that the choices were somewhat arbitrary and were dependent on the applications considered. Only a few such functions seemed to be clearly not useful. But providing a small list of excluded functions seemed not to widen the set of feasible implementations and at the same time was user unfriendly.

### **Case Value Range**

Your proposal to delete the case-value range for ":" was accepted.

# X3 Subgroup -22- Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, -with reasons

Comments (Additional pages of comments may be attached.)

SEE ATTACHED DOCUMENT 103(\*)TMR-1

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME T.M.R. ELLIS  
(PLEASE PRINT)

SIGNATURE TMR Ellis

DATE: 16 December 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

10.0

31

To: Jeanne Adams. Chair - X3J3  
From: Miles Ellis  
Date: 15 December 1986  
Subject: Response to X3J3 Letter Ballot - December 1986

My vote is in favour of submitting X3J3/S8/103 to X3 for further processing as an American National Standard. However, I have two proposals that I would ideally wish to see incorporated in the document before such submission.

Proposal 1 is concerned with Processor Conformance and, in particular, with Resolution 10 from the WG5 meeting at Halifax, Nova Scotia, in August 1986 (although in a slightly modified form).

PROPOSAL #1

Page 1-2, lines 5.6: Delete the first sentence of the paragraph and insert the following additional paragraph:

A processor conforms to this standard if

- (1) It executes standard-conforming programs in a manner that fulfills the interpretations herein.
- (2) It contains the capability to detect and report the use within a submitted program of a form designated herein as deleted, obsolescent, or deprecated, insofar as such use can be detected by static inspection of the program.
- (3) It contains the capability to detect and report the use within a submitted program of an additional form or relationship permitted by the processor, as described below.

Page C-1 line 2: Before the first sentence of the Section 1 Notes insert the following additional paragraph:

This standard requires a standard-conforming processor to be capable of detecting and reporting the use of a permitted additional form or relationship, or the use of a form designated as deleted, obsolescent, or deprecated where these can be detected by static inspection. It is expected that the processor will be accompanied by documentation which defines the additional forms and relationships that it allows, and which describes the means of reporting the use of such permitted additional forms and relationships, or of deleted, obsolescent, or deprecated forms. In this context the use of a deleted form is, of course, a permitted additional form.

END PROPOSAL #1



**RESPONSE TO MILES ELLIS BALLOT**

There were two proposals with this ballot.

The first, on conformance, was modified in committee and accepted.  
The second was an editorial proposal that was handled in another way.

# X3 Subgroup<sup>-25-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

See attached X3J3/103(\*)MFF-1 and X3J3/103(\*)MFF-2

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME MURRAY F. FREEMAN  
(PLEASE PRINT)

SIGNATURE *Murray F. Freeman*

DATE: 30 December, 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

30 December 1986

103(\*)MFF-1

To: X3J3  
 From: M.Freeman  
 Subject: Comments on X3J3/S8.103 attached to letter ballot

**General.** The comments below are almost entirely in the form of suggested edits to the document. I believe them to be editorial in nature; in those cases where they appear to be substantive, I consider the document to have been incorrectly transcribed or edited. Most of the edits are to improve readability and accuracy.

**List of Edits.** The edits below are in document order, except in a few cases where several changes are interdependent (usually where some reordering is suggested); in such cases the "key" change is given first. The form of each edit is:

*nnn. [page/line] suggested edit <rationale for edit>*

where *nnn* is the edit number, and *page* and *line* are the S8.103 page and line numbers. If the edit involves more than one line on a single page, *line* is replaced by *flin-llin*, where *flin* and *llin* are the first and last line numbers involved. Similarly, if the edit "spans" a page boundary, *page/line* is replaced by *fpag/flin-lpag/llin*, indicating the pages and lines involved. For text additions involving whole sentences, the line number for the end of sentence to be followed is used, if the added text is to be inserted in the present paragraph. If the added text is to form a new paragraph or paragraphs, a plus sign (+) is suffixed to the line number of the last line of the paragraph to be followed. The edits suggested are:

1. [iii/18] delete "symbolic" <missed edit>
2. [iv/19] replace "all of the" with "the execution" <exclude WHERE construct>
3. [iv/29] replace "name-directed" with "namelist" <missed edit>
4. [v/6] delete "symbolic" <missed edit>
5. [v/12] replace "that time" with "April 1978" <"that" reference too far back>
6. [v/14-15] replace sentence "At the present ... members" with:  
 "As of December 1986, there were 37 principal members of X3J3."  
 <specific dates & the voting membership should be indicated>
7. [ix/(7.5.2)] add "-" between "Assignment" and "WHERE" <indexer bug ?>
8. [x/(9.8)] correct page no. to "9-23" <indexer bug ?>
9. [xiv/(F.2.3)] add "-" between "Assignment" and "FORALL" <indexer bug ?>
10. [1-1/20-22] replace "computers" (twice) with "computing systems" <consistency>

11.1

35

**RESPONSE TO MURRAY FREEMAN BALLOT**

The committee appreciates your detailed editorial examination of S8. Your editorial items with this ballot have been considered by X3J3 and either accepted or rejected. Table C.1 has been reformatted as you suggested.

# X3 Subgroup<sup>-28-</sup> Letter Ballot

Accredited Standards Committee  
) Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

See attached comments

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME Kevin W. Harris  
(PLEASE PRINT)

SIGNATURE *Kevin W. Harris*

DATE: December 31, 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

12.0 37

Response to Letter Ballot on submission of FORTRAN 8X  
to X3 for processing as an American National Standard  
Digital Equipment Corporation

Digital Equipment Corporation votes NO on this ballot.

1 TECHNICAL COMMENTS

While Digital believes that an updated Fortran standard is overdue, and that X3J3/S8.103 contains many worthwhile new capabilities, we believe there are still several major problems that need to be addressed before forwarding this document as a draft proposed American National Standard. We appreciate the actions the committee has taken in response to the previous letter ballot. We would appreciate similar attention to the problems described below.

We believe that publishing this document may jeopardize the past successes of Fortran in its attempt to make several major changes to the language. We have several specific areas of concern. In each of these cases, it is apparent that the committee has made an attempt to address an acknowledged weakness in the Fortran 77 standard. However, we believe that the solutions chosen have significant drawbacks. Our major concerns are in the following areas:

1. Users have expressed the need for improved data type support in several areas, particularly the need for new intrinsic types for supporting bit, pointer, and varying length string operations. This proposal attempts to provide these through a set of language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this attempt will be too inefficient.
2. The new source manipulation capabilities are more powerful than necessary, too complex, and untested.
3. The new features intended to enhance portability of numerical software are untested by practice and are not ready to replace the existing default REAL and DOUBLE PRECISION declarations. They are not clearly effective at attaining the desired portability because they do not account for major sources of variability such as round-off error and intrinsic function accuracy.
4. The features chosen for the obsolescent and deprecated features lists are not justifiable based on potential benefits or costs.

These problems and proposed changes are described in more detail below.

## .1 Language Extensibility

This proposal contains a large amount of text pertaining to the topic of the use of derived types and language extensibility features as providing an effective data abstraction mechanism. The overview states that: "derived data types provide an effective implementation mechanism for data abstractions." Derived types are introduced in §2.4.1.2, discussed at length in §4, §5, and §6, and other aspects are treated in §7.1.4, §9.4.2, §9.4.3.4.2, §11.3.3.3, §11.3.3.7, and §14.

Other data abstraction related topics are discussed in §7.1.3, §7.3 (Defined operators), §11.3.3.6 (Operator Extensions), §12.5.2 (Overloading operators), §12.5.5 (Overloading names), and §14.

Several modules have been written using the language extensibility features described in this proposal, for providing support for important data abstractions. An example of this is given in §C.11. These have included a package to provide an approximation of a bit data type, a pointer type, and a varying length character type. In all these cases, however, the fact that the implementation methods rely on traditional Fortran intrinsic types and operations guarantees that these abstractions will be awkward to use and slow to execute.

An approximation to a BIT type, for example, will take many times the storage needed by an intrinsic BIT type. And it will not be able to take advantage of most of the powerful logical operations available on most modern CPUs. This will result in an order of magnitude efficiency degradation. An approximation to a pointer type cannot use true machine addresses and therefore will end up being several times slower than similar operations provided with an intrinsic pointer type. An approximation to a varying string type cannot take advantage of the built in string manipulation capabilities of the hardware. This result will likely be several times slower than an intrinsic type.

Thus, the language extensibility features are an inadequate means of providing the data abstractions that users have been requesting because they are incapable of taking advantage of the available hardware capabilities.

We recommend that all the text in X3J3/S8.103 devoted to the use of derived types as a data abstraction mechanism be removed and replaced with portable intrinsic language support for bit strings, general pointers, and varying length character strings.

## 1.2 Source Manipulation

This proposal describes a very large set of new features in this area. These are generally identified as the MODULE/USE facility, described in §11.3, §12.1.2.2, §14, §B.3.1 and §C.11. We are concerned because the new packaging features embodied in the MODULE and USE statements are complex, new, untested, and differ so widely from existing practice. We believe that a simplification of these facilities is warranted, and that this can be done without significantly degrading their benefits.

Although the description of the MODULE/USE capability is short, it implies great user and implementation complexity. In particular, there is a clear need for a reasonable implementation to provide a form of "precompiled" modules. As implied by §C.11, these will need an inter-file pointer capability to support the name space management requirements implied by the ability to nest a USE statement inside a module. In order to support module procedures, such precompiled modules must also carry the compiled (object) form of the procedures inside them as well. This requirement is further complicated by the need to tailor the module procedures to the environment in which they are USED, e.g. which versions of user specified generic types should be used.

We believe that there will be two principal effects of this complexity. First it will delay or inhibit implementation of these features. Since the implementation of these features isn't described anywhere, it will certainly require a great deal of time to specify exactly how each implementation will work in addition to the pure development time. Second, the implementations will be error prone. Since this is uncharted territory, it is highly likely that hidden limitations and problems will plague the early implementations.

Based on these problems, we have several recommendations. The first is to standardize the spelling of a textual INCLUDE facility. This will be beneficial with either the current form of MODULE/USE or the following simplified form.

We have two suggestions concerning how to simplify the precompiled modules without harming the utility of MODULE/USE. These are to eliminate module procedures and disallow the USE statement inside a module.

We believe that these two changes will preserve the important advantages of MODULES described in §C.11 while dramatically simplifying the complexity of both understanding and implementing them. A great deal of complexity is eliminated because it would no longer be necessary to carry the object form for the module procedures inside the precompiled modules. Intermodule references are not needed when modules can't refer to other modules.

Although this change would force the use of external procedures instead of module procedures, this wouldn't degrade the checking or other benefits provided by module procedures. By inserting the appropriate procedure interface blocks in the modules, both the definition and the uses of the procedures would be explicit and fully checked for consistency. The definitions of the module procedures would still have full use of the global data defined by the module.

Allowing USE inside a module contributes greatly to the name space management problem and contributes very little to the actual utility. In practice, authors write self contained packages that do not depend on other packages. This maximizes the usability of their software, so they have little need for this capability.



### 1.3 Portability Of Numerical Software

The new PRECISION and EXPONENT RANGE attributes are intended to replace the traditional Fortran REAL and DOUBLE PRECISION declarations for floating point quantities. This change is reflected throughout the body of the proposal, in §4.3.1.2 (description of type parameters), §5.1 (declaration syntax), §7.1 (expression rules), §13.6 (related intrinsic functions), §B.3.2 (deprecation of DOUBLE PRECISION type), and §C.4 (notes).

Although S8.103 is very thorough at describing the changes needed to support declaring precision and range of floating point quantities, it ignores two major influences of how numerical software is actually used.

First, a major requirement for high precision is the round-off and other errors introduced by inaccuracies in the floating operations and intrinsic functions. For example, just because a given computation requires 5 digits of precision on vendor A's machine doesn't mean it won't require 10 on vendor B's machine. The cause could be as simple as an inaccurate EXP function or as complex as an unstable convergence algorithm.

Second, users are generally unaware of their precision requirements. Good error analysis is used with only a small fraction of the numerical software in service today. In practice, users generally increase the accuracy and manipulate the calculations until they obtain a satisfactory result. This practice will cause problems any time the requested and actual precisions differ. For example, when a program requests 7 digits of precision but 15 digits are actually used, the true precision requirements may be anywhere between 7 and 15 digits. When porting this program to an environment that can deliver a hardware 7 digit precision, the user will often find this inadequate.

An additional problem lies in the fact that using the PRECISION and EXPONENT RANGE attributes cannot guarantee use of the two underlying hardware types generally provided on any given vendor's CPU. This means that a user cannot reliably exploit the underlying numerical hardware using these Fortran 8x features.

For these reasons, the new numerical control features are still experimental and it is premature to propose them as replacements for traditional Fortran usage. We recommend that these new attributes be removed from the proposal pending actual trial usage.

In addition, we recommend that alternative methods be sought to remedy the original problem these features sought to address. Although it would be difficult to develop, we believe that a more straightforward approach to solving the underlying problem, such as requiring minimal precision and range in the storage representation, and minimal accuracy standards for operations and intrinsic functions, would appear to be less risky and more productive.

## 1.4 Deprecated Features

While the new deprecated features architecture allays many of Digital's concerns over preserving investment in existing software, there is still a problem to be addressed. This is the lack of any rationale for the choice of what is on the obsolescent and deprecated features lists.

Take the first item on the list, for example, Arithmetic IF. While this feature is redundant in Fortran 77, it is frequently present in old programs. Indeed, it is present in many widely used benchmark programs to this day. There is no particular reason for a vendor to remove support for Arithmetic IF from his compiler, especially when such support is required for benchmark and standards qualification. It is easy to implement when writing new compilers. In short, there is a significant cost for its removal and no apparent benefit. How is this cost justified?

The same is true for many of the items on the Obsolescent and Deprecated lists. We would put items 1, 3, 4, 5, and 6 of the Obsolescent list (§B.2) and items 2, 3, 4, 5, 6, 7, and 8 on the Deprecated list (§B.3.2) in this category. We recommend that only items that are clearly costly in some major way, and can be easily replaced be allowed on these lists. For example, use of floating point DO variables frequently leads to errors in counting iterations. Use of Assigned GOTO pays a high penalty in missed optimizations.

Section B.3.1 is devoted to the deprecation of storage association. The only thing forcing X3J3 to take this position is the numerical control features discussed above. Other than this, storage association lives compatibly with the other S8 features. We regard storage association as a useful tool to be enhanced, rather than a mistake to be removed. Thus, we recommend that in addition to removal of the numerical control features, and the storage association features be removed from the deprecated features list.

## 2 EDITORIAL COMMENTS

It appears that a rule needs to be added to section 2, 12, or 14 to allow internal procedures to access the local names of their host procedures. In particular, §2.2.1 specifies that a scoping unit of program unit or subprogram excludes the subprograms contained within it. Section 14.1.2 defines local entities as belonging exclusively to their containing scoping units. Internal procedures are essentially useless without such a rule.

Some problems still exist in using consistent terminology. For example, §14 contains the text "and constructs have a scope of a scoping unit." The term "construct" is not to be found in any of the various indexes or glossaries, or indeed anywhere else in the document that we can find. We suggest that an automatic cross reference is needed on the terminology used in the text, in addition to that used in the BNF.

## RESPONSE TO KEVIN HARRIS BALLOT

X3J3 has considered your ballot and appreciates your continued participation in our effort to complete the draft standard for Fortran. Your two editorial comments were handled in another way in other ballot comments. Your points on Language Extensibility, Source Manipulation, Portability of Numerical Software and Deprecated Features have been considered in full committee and rejected. The following notes summarize the committee position on your four technical issues.

### 1.1 Language Extensibility

The ballot objects to the "Language Extensibility" features proposed for Fortran 8x on the grounds that they are inadequate to provide facilities for bit data, pointers, and varying length characters.

The reason that these three intrinsic facilities are not in the current draft is that no proposal has been able to command and maintain majority support. That bit data and pointers should be included as intrinsic data types, if they are included at all, has not and never has been seriously in doubt. That a dynamically variable length character would be best done as a new form of aggregate is also fairly widely accepted. Although the semantic extension capabilities with only modest additions could adequately express this functionality, the deficiency with the language for varying character lies more with overly restrictive storage management facilities (one is not permitted to allocate a string) and deficiencies in the area of I/O (the lack of input where the record structure is data determined rather than format).

The "Language Extensibility" features, which include the facilities for:

1. Defining and declaring structured data objects,
2. Defining procedures to manipulate such objects,
3. Allowing such procedures to be invoked using a conventional functional notation,
4. Allowing a restricted class of such procedures to be invoked using normal infix operator notations,
5. Allowing the "generic" or "overloading" properties of intrinsic functions and operators to be extended to user defined procedures using either invocation syntax,

were introduced to satisfy strongly expressed user needs for facilities to handle heterogeneous aggregates of data and to allow programs working with such objects as physical three vectors, relativistic four vectors, rational numbers, tensors, etc. to be written in a notation analogous to that conventionally used in these various fields. The facilities introduced although probably capable of being improved (all human invention is less than perfect) do allow such functionality in a reasonably economical manner.

## 1.2 Source Manipulation

You make a comment about the complexity of modules and USE, and two recommendations. We respond to those three items individually.

1. **Complexity of Modules and USE.** Modules and USE are certainly more complex than a simple textual include. However, X3J3 believes that they provide important functionality in an appropriate way, and are worth the cost. They offer a method for packaging data definitions and subprograms that logically belong together, and they do so in a way that is clearer and less error-prone than a textual include.

As to these features being new and untested, they are new to Fortran, but similar features have existed in other languages for several years.

2. **Standardization of a Textual Include Facility.** The INCLUDE statement is indeed implemented in many existing FORTRAN compilers. However, "standard practice" is not so standard. The syntax of the statements varies, and of course the syntax of file names is processor-dependent. X3J3 has considered an INCLUDE statement, but has been unable to agree on a way of specifying it so that it has the portability required of a statement in standard Fortran. In addition, X3J3 feels that an INCLUDE statement is unnecessary, in that most actual uses of INCLUDE can be better written in terms of modules and USE.
3. **Simplification of Modules and USE.** It is X3J3's feeling that adoption of this suggestion would not reduce the complexity of modules and USE in any meaningful way, but could significantly reduce their functionality.

## 1.3 Portability of Numerical Software

The proposed extensions aim to provide linguistic portability in a context which assumes that different vendors will provide floating-point approximations of varying precision and range, and that certain unstated basic levels of quality in implementation will be met. They do not attempt to solve problems associated with algorithmic portability or numerical stability. A program implementing an algorithm that is inherently sensitive to the fine detail of numeric precision, exponent-range, round-off mechanism, intrinsic function accuracy, etc. is inherently non-portable. A processor which represents floating-point quantities to some precision, but then proceeds to perform arithmetic or evaluate intrinsic functions to significantly less than this precision is simply an inadequate processor. Neither of these problems are amenable to solutions through language design.

FORTRAN 77 has three major linguistic portability problems in the floating point area.

1. The language does not provide any standard portable way of exploiting all of the floating point approximation methods supported by any processor. Only REAL

and DOUBLE PRECISION, and COMPLEX are supported linguistically. Many processors support three, or more, different floating-point approximation methods. The language needs to support use of these methods for real, complex and any other floating point data type.

2. Since an application that one processor may run adequately using the approximation method implementing REAL, may on another processor require DOUBLE PRECISION, the move of a program from one processor to another frequently requires major and distributed changes to the source codes. Similarly, a program may run adequately on one problem using REAL, but on another with different data values it may need to be run using DOUBLE PRECISION, again requiring major source code modifications. The language needs to be able to specify the floating point approximation characteristics required in a way which is processor independent and can be changed in a way which is at least potentially localizable to a few places in any program unit.
3. FORTRAN 77 does not provide a facility for user defined generic procedures. FORTRAN 77 introduced a major portability enhancement when it defined "generic" intrinsic functions. This allowed a program to be largely changed from REAL to DOUBLE PRECISION without having to search for every reference to an intrinsic function so as to change its name. This facility is required also for user written procedures.

The extensions proposed for Fortran 8x in this area solve these three portability problems. They do not handle, or attempt to handle, the sort of portability problem which could arise if a program were moved between machines with similar approximation methods but where one performed its arithmetic rounded and the other truncated. An application sensitive to such differences may well have numerical problems that are entirely non-linguistic. Provided such a program had been sensibly written the Fortran 8x floating point facilities would make it easy to force the program to use a more precise approximation method but otherwise this is not the type of problem envisaged as solvable by the language extensions proposed.

The optional addition of the parameters PRECISION and EXPONENT\_RANGE allow the selection of the floating point approximation method to be done in terms of minimum approximation quality requirements. These parameter values are so constrained to make the selection of approximation method an entirely static choice. These new forms provide a processor independent method of selecting "single precision" on one machine and "double precision" on another. In effect a type attribute such as

```
REAL(PRECISION = 10, EXPONENT_RANGE = 50)
```

is an alternative spelling for REAL on large-word machines and for DOUBLE PRECISION on small-word machines.

Since this method of selecting floating point representation is not dependent on any unique spelling for each supported approximation method, it provides a portable

linguistic device for providing access to any number of methods. Two may be selected and used, where appropriate, on a machine which supports only two, but on a machine supporting three or more, three or more may be used. This effectively solves the first two linguistic portability problems. The remaining one of allowing user-defined generic procedures is handled by allowing dummy arguments to be declared to have this generic property.

#### 1.4 Deprecated Features

The choice of what is on the obsolescent, deprecated and deleted features lists reflects the committee's best judgment of expected use of these features. If this judgment is correct, this mechanism allows the features to be removed from the language. If this judgment is incorrect, the features can remain.

Deprecated features are listed in an appendix as possible candidates for the obsolescent category in a future draft standard. They are not marked in the text, and are fully supported features of Fortran 8x, residing compatibly with the new features. Decisions about these features will be made by a future standards committee.

Obsolescent features are marked in the main text, but are fully compatible with new features. They may be deleted by the next Fortran Standards Committee. The rationale for obsolescent features is that these features are available in a better way in FORTRAN 77 and are therefore redundant. Since these features are marked in a special type font in the draft for Fortran 8x, users will be made aware that this feature is redundant and encourage change to a better method. There is no easy way to remove features from a language standard. However, X3J3 believes that the architecture for Fortran will give users at least two standard revision cycles for removing a feature. Older languages, like Fortran, that have had a long and successful history, should not be left to expand beyond a reasonable number of features; nor should they be prevented from adding new modern features that might make the language too large. The method suggested allows Fortran to grow and change and remain within the bounds of size and complexity, if obsolete features are marked and therefore grow into disuse in the Fortran Community.

There are no Deleted features in Fortran 8x.

-38-

# X3 Subgroup Letter Ballot

Accredited Standards Committee  
Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

See Attached

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME RICHARD A. HENDRICKSON  
(PLEASE PRINT)

SIGNATURE Richard A. Hendrickson

DATE: 1/3/87

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers  
Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8338  
Fax: 202/538-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

13.0

47

Comments on the NO vote.

Dick Hendrickson

The major reason for my voting NO is the large number of technical errors and holes in the current document. I have serious reservations about some of the features in Fortran 8x and several suggestions for new features, however, I believe the document should be forwarded for public comment as soon as possible. With this in mind I will change my NO vote to YES if the bulk of the technical problems are eliminated, without any action necessarily being taken on the features. The number of flaws in the current document, in my judgment, make it unlikely that we will get any meaningful public comment. We need to make a major effort to clean up IDENTIFY, USE and modules, and clarify the BNF.

I have detailed comments on each chapter. I did not have time to review the Appendix or glossary. The comments for each chapter include page number, starting and ending line number (99 generally means "continued on next page") and a severity level.

- Edit: Basically a rewording of something which isn't clear.
- Hole: A missing or contradictory item.
- Minor: What I believe to be a mistake. The text is clear enough, I just believe we should do something else.
- Major: A significant mistake. I strongly believe we should do something else.

Throughout the comments I used the term "depricated" when I should have used "Obsolescent".

Before I would vote YES essentially all of the edit and hole comments and items 1,2,4,6,7 and 13 below must be resolved.

I also have several general comments which didn't seem to fit in to a line by line chapter list.

- 1) We should replace "assumed shape" with "assumed extent". Shape includes both rank and extent, however, only the extents are assumed. The same applies to "deferred shape". The current terms are correct, but confusing.

13.1

48



- 2) We need to clarify what arrays can be ranged. I believe any array that is not in a globally accessed range list should be rangeable in a local subroutine. The text is unclear on this.
  
- 3) I believe the requirement that derived types and shared entities be in a separate module which is explicitly USED by each subroutine is awkward. I'd like to explore an "executable module". A module with a main program in it. The main program and every subprogram would share the common definitions from the module specification part without explicit USE statements.
  
- 4) I strongly believe that we must limit the complexity of expressions that can be in a constant expression. The current rule in Chapter 7 is something like "... or an intrinsic function with constant arguments...". This requires the processor to have available at compile time the "more than 100" intrinsic functions. Actually, the number is much greater, since there is likely to be a different implementation for each supported type, for functions with and without DIM= or MASK= parameters, etc. This is very complicated and will never be used.  
  
I strongly believe that the restriction in CH 7 must be changed to something like "... the intrinsic functions MAX, MIN, ABS applied to scalar arguments, and the inquiry functions...". This is a reasonable restriction, compilers can extend the list if necessary.
  
- 5) I would like to replace "%" with "." as the structure qualifier. I believe a simple rule like "No structure component name can be the same as any operator name" is simple, sufficient, easy to check at compile time and never likely to be violated.
  
- 6) I could not find any explanation of what internal procedures inherit from their host.
  
- 7) I could not find any explanation of how an allocatable function allocates and deallocates its space.
  
- 8) I would really like to see vector valued section selectors returned to the standard. I have had questions from 2 customers about them since we removed them.
  
- 9) I would like to see a stronger conformance section. Like Pascal, require the processor to describe what errors it doesn't detect, and what happens.

10) I still think we need some type of BIT data type. How can a person use TRANSFER to read in a tape record, not produced by FORTRAN, which contains packed 12 bit photomultiplier output? I think BIT data and PACKED structures would allow reasonably portable access to non FORTRAN data.

11) I do not like the CONTAINS statement. Would it be possible to change it to a declarative and give it a list? This would seem to resolve the ambiguity. We would only require a contains in a module header and in a subprogram with no executable statements.

12) I think the IOLENGTH feature in INQUIRE is needlessly complicated for something that will be seldom used. Could we not invent an IOLENGTH function? The function would return, in processor dependent units, the record size needed for its argument. A person who has a list would compute  $IOLENGTH(a) + IOLENGTH(b)$ .

13) We must prohibit use of transformational functions where array element selection is taking place. This means in function references if there is a MASK= argument and in a WHERE assignment statement.

## RESPONSE TO RICHARD HENDRICKSON BALLOT

X3J3 appreciates your extensive and careful reading of the draft Fortran standard. The individual comments of both an editorial and substantive nature have been considered by full committee and either accepted or rejected.

Each of the general comments in the introduction to your ballot have been considered and either accepted or rejected, or withdrawn by you. The following notes address these general comments.

### 1. Assumed Shape and Assumed Extent

The X3J3 committee discussed a suitable alternative descriptive term for the concept of "assumed shape". After considerable discussion, the committee agreed that the term "assumed shape" was slightly misleading but that the suggested term "assumed extent" seemed more confusing as the phrase, taken literally, includes the FORTRAN 77 concept of "assumed size". A more appropriate term might be "assumed extents" but the confusion created by two very different concepts being distinguished by a singular versus plural term was unacceptable.

In support of the current term "assumed shape", the committee felt that a basic principle for the language was that once an array was declared, its rank was fixed. The definition of "assumed shape" currently makes this assumption -- "An assumed-shape array is a dummy argument array that takes its shape from the associated actual argument array". An added sentence to clarify the definition might be helpful, namely "An assumed-shape argument is required to be of the same rank as the associated actual argument array."

### 2. Ranged Arrays

Currently RANGE is an attribute of an array. Local arrays obtained via a USE statement may not have any attributes changed in the USEing subprogram. Thus, such an array may not be ranged if the module writer neglected to specify RANGE in the module. Although it might be nice to relax this restriction, it does not prevent you from doing what you want. If you IDENTIFY a local array onto the USEd array, you can then effectively RANGE the USEd array and the range information is local to the subprogram. X3J3 feels that since the basic functionality is provided, a change to relax the restriction is not required.

### 3,5,8,10,11.

After the initial ballot of X3J3, the committee reviewed the comments that indicated that the language was too large. A compromise was reached within the committee to reduce the size of the draft. Many new features were, with regret, removed in order to satisfy this general complaint. Your ballot asks for new features or a return of the features placed in Appendix F; an executable module and putting BIT data type and

vector valued section selectors back into the language. The reduced language that was approved by the committee should remain. Further work by the committee would be counter-productive and would represent "thrashing" over ideas that have had much committee time and would not necessarily change the votes on these issues.

The CONTAINS statement was reviewed again based on another ballot, and left as it stands. The "%" structure qualifier was reviewed again also. This feature was again left as in the draft. Further progress by X3J3 cannot be made in these areas where repeated discussions result in the feature remaining as it is currently defined.

#### 4. Complexity of Constant Expressions

The X3J3 committee considered the concerns expressed in the above ballot comments but felt no changes were required for the following reasons.

FORTRAN 77 currently permits constant expressions to include all of the intrinsic arithmetic, character, and logical operators with constants or constant expressions as operands, except for the exponential operator \*\* which is limited to integer powers. Therefore, cross compilers even for FORTRAN 77 currently must address the issue that operands may have a very different representation on the host than on the destination machine.

On the other hand, FORTRAN 77 does not require that such expressions be evaluated at compile-time but can be postponed to runtime. Whether it is an advantage or disadvantage to do so depends upon the machine and the properties of the object files and the linker/loader.

The issue, therefore, is not one of precedent but of the number of operations (including those embodied in intrinsic functions) that must be supported in what appears to many to be compile-time expressions. Several models of how this can be implemented are readily apparent. Which one is sensible will depend upon the machine and also on the tradeoffs between storage and execution and on whether the evaluation of such expressions is borne at compile-time, linker/loader-time, or execution-time. (One model that addresses the particular example given in the ballot comment RA 9.1 is one where the expressions are evaluated just prior to the execution of the main program and the values inserted into the appropriate places in the load image.)

With the determination of execution-time determined extents for arrays, the need for arbitrary constant expressions in specification contexts becomes apparent. However, the use of the entire intrinsic function set appears at first to be excessive. When attempting to formulate a rationale for including some but not all of these functions, it became clear to the committee that the choices were somewhat arbitrary and were dependent on the applications considered. Only a few such functions seemed to be clearly not useful. But providing a small list of excluded functions seemed not to widen the set of feasible implementations and at the same time was user unfriendly.

**7,13. Withdrawn Issues**

These issues were withdrawn by the commenter.

**9, 12. Resolved Issues**

These issues were resolved in a proposal contained in another ballot.

# X3 Subgroup Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986  
Project: 67  
Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

*(3 documents attached)*

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME Kurt W. Heichert  
*Kurt W. Heichert*  
(PLEASE PRINT)

SIGNATURE *Kurt W. Heichert*

DATE: 87/1/2

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

14.0 54

Subject: Comment on Balloting Procedures  
From: Kurt W. Hirschert

The ANSI procedures under which this work is being done appear to be based on the expectation that the technical committee preparing a standard will not forward it for public comment until the committee believes it has a document suitable to be the final standard. Under this expectation, the public comment period is really only a safeguard that the committee has done its job properly, and the effects of public comment on the document should normally be only minor technical and editorial clarifications.

This expectation is reasonable for most of the work done under these procedures. If the standard is merely a ratification of existing practice, with few significant technical decisions to be made by the committee, then public comment need only be concerned with whether the existing practice has been correctly described. If the standard is short enough that reviewing it is not a major task, it can be expected that people interested in commenting on the technical decisions made by the committee will be able to do so during the time the document is being prepared, so that no new viewpoints will be revealed during public comment. If the expected users of the standard are sufficiently homogeneous that all relevant viewpoints are adequately represented on the technical committee, one can again expect that the public comment period will reveal no new viewpoints.

This preparation of this document, however, involved the integration of features taken from a great many sources and thus involved a great number of technical decisions. The document itself is so large that even when people outside the committee have been willing to invest the time to review it, they generally have not been able to complete the review of one version of the document before a new version replaces it. There are relevant interest groups that are not represented on the committee, notably the suppliers and users of Fortran on personal computers, for whom the cost of participating on the committee is not economically justifiable. As a result, it is extremely unlikely that public comment on this document will not result in significant technical changes, no matter how long the committee works on the document beforehand.

Given these circumstances, a vote to forward this document for public review does not necessarily mean that one believes that this document should be the revised standard. It does mean that there is nothing in this document that would make it unreasonable for this document to be the revised standard.

I am voting to forward this document for public review in spite of some reservations I have about it. I am concerned that some of the new features (e.g., RANGE) may be a less than optimal choice for solving the problems they are intended to address, but I note that Fortran has survived the suboptimal decisions such as using column-major storage order rather than row-major order. I am concerned that some features (e.g., assumed precision) may have been formulated in ways that make possibly desirable extensions unnecessarily difficult, but again I note that Fortran has survived other roadblocks to extension such as the syntactic difficulties in using expressions to specify the initial values in a DATA statement. I am concerned that there are features (e.g., pointers) that ought to be made a part of the language at this time but that are not included in this document, but once again I note that Fortran has survived such omissions as the failure of Fortran 66 to include end of file testing (corrected in Fortran 77) and the failure of Fortran 77 to make end of file testing work on all sequential files produced by Fortran (corrected in this document). In other words, even if all of my concerns about this document should prove valid, there is no reason to believe that a failure to address them would irreparably harm the language.

All of these concerns are of a nature similar to those that I would expect to be expressed during public comment, even if these particular concerns were addressed beforehand, and some are such that they will be more easily evaluated after a wider sampling of public opinion has been obtained. Thus, considering these concerns as a part of the public review process seems to be the most expeditious way to proceed in the production of a revised standard.

Ω

**RESPONSE TO KURT HIRCHERT BALLOT**

Each of your specific editorial and substantive issues have been reviewed by the committee, and either accepted or rejected.



# X3 Subgroup -48- Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME Tracy Hoover  
(PLEASE PRINT)

SIGNATURE Tracy Ann Hoover

DATE: 23 December 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8898  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or group represented on the committee to the use of the American National Standard under consideration.

15.0  
57

To X303  
From: Tracy Hoover  
Date: 23 December 1986  
Subject: Edits to SR. 103

Proposal

Make the following edits to SR. 103:

1. p ii, line 29 - Replace "All of these deficiencies, and more" by "These and other deficiencies."
2. p v, line 12 - Replace "that time" by "the publication of Fortran 77."

**RESPONSE TO TRACY HOOVER BALLOT**

Your first editorial comment was accepted, your second editorial comment was handled in another ballot comment.

# X3 Subgroup Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986  
Project: 67  
Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

[Empty box for stamp or signature]

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

Mail to: Jeanne Adams, Chair, X3J3 NAME E. Andrew Johnson  
 NCAR - SCD (PLEASE PRINT)  
 PO Box 3000  
 Boulder, CO 80307 SIGNATURE E Andrew Johnson

DATE: January 2, 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/337-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

16.0 60

# Prime Layered Systems Software Memorandum

To: Jeanne Adams, Chair X3J3  
 From: E. Andrew Johnson  
 Co: Sheryl Horowitz  
 Subject: Comments accompanying YES vote on X3J3 letter ballot 103(\*)JCA-2  
 Date: January 2, 1987

## Editorial Changes

- p. 9-15, l. 15-16 Delete "; Let  $n$  be  $-$  specifier". The revision from 102(\*)PKT-2, which removed the references to the  $n$ th value, make this postulation irrelevant.
- p. 12-12, l. 10-15 The second sentence describes what happens on "Execution of the CONTAINS statement". The third sentence states that the CONTAINS statement is *not executable*. I suggest changing the words "Execution of the CONTAINS statement  $-$ " (line 11) to "Flow of control reaching the CONTAINS statement  $-$ ".

## Request for Clarification

p. 4-8

One would really like to be able to reference symbolic constants and type names within a type definition, but S8.103 does not permit this, since there is no mention of inheritance of names into a derived type definition from the containing scope.

The example in S8.103 is:

```

TYPE STRING (MAX_SIZE)
  INTEGER
  CHARACTER (LEN = MAX_SIZE) VALUE
END TYPE STRING

```

but the standard does not support the following:

```

TYPE ID_STRING
  TYPE (STRING(MAX_ID)) VALUE
END TYPE ID_STRING

```

because it describes this type definition as a distinct scoping unit with no apparent way to inherit the type name "string" and the constant name "max\_id" from anywhere (scoping units are specifically defined as nonoverlapping). If this restriction is an accidental fallout of the "scoping unit" cleanup, it should be addressed (a clarification of the meaning of "Scoping Unit" would suffice). If the restriction is intentional, a negative example (like the one above) should be in the document (a note would suffice).

16.1  
61

**RESPONSE TO ANDREW JOHNSON BALLOT**

Your first two editorial comments were handled in another ballot comment. Your third was discussed in full committee and the issue was resolved in another way.

# X3 Subgroup<sup>-54-</sup> Letter Ballot

Accredited Standards Committee  
Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

see attached sheets.

Mail to: Jeanne Adams, Chair, X3J3 NAME Anil Lakhwara -  
 NCAR - SCD (PLEASE PRINT)  
 PO Box 3000  
 Boulder, CO 80307 SIGNATURE *A Lakhwara*  
 DATE: 2 January 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8338  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

17.0 63

Response to Letter Ballot on submission of FORTRAN 8X  
to X3 for processing as an American National Standard

Anil Lakhwara

Peritus International Inc.

December 31, 1986

Question:

Do you approve the draft Fortran Revision (Fortran 8X) of X3.9-1978 for submission to X3 for further processing as an American National Standard ?

Answer:

Peritus International votes no on this ballot.

Reasons/comments

Peritus's major concerns with the current form of the draft FORTRAN revisions are:

1. Non conformance of S8.103 to X3J3 Charter.
2. Size and complexity of the language.
3. Implementability of the language.

Based on these concerns, we recommend that this document not be forwarded to X3. Given below is a brief discussion of these problems

1. Non conformance of S8.103 to X3J3 Charter.

X3J3's charter was to standardize FORTRAN based on currently implemented language extensions , with exceptions only when there is clear evidence of the need for change. In our opinion this requirement has not been met; instead, S8.103 is a new language with Fortran-77 as a small subset. S8 draft revision reflects little concern for the history or the virtues of the FORTRAN as we know it today. The relationship between the present standard (X3.9-1978) and its predecessor (X3.9-1966) is clearly seen; this cannot be said for the present standard and the proposed S8.103.

Fortran standard affects all areas of science and technology; X3J3 is responsible to both the Fortran community and to those affected by the results of Fortran applications, to



provide a quality, error-free and reliable standard. No Fortran implementations exist for the bulk of the proposed extensions. Thus forwarding S8 to X3 for further processing would violate the the very basis of the standardization process.

Some well accepted extensions to Fortran language have been excluded. Most notably the INCLUDE facility and and the BIT data type. S8 also proposes some new solution to problems that have already been addressed in existing compilers in various manners.

In our opinion, X3J3 should have spent bulk of effort on the standardization of the existing extensions and providing reasonable array processing extensions. The need for the array processing extensions existed even at the time of the public review of the current standard.

## 2. Size and complexity of the language.

The language proposed in S8.103 is very large and complex. Fortran-77 is just a small subset. The immense size and complexity of the proposed language and its radical departures from FORTRAN-77 leads to the perception of two distinct languages glued together under the name of FORTRAN-8X. We believe that this is neither needed or wanted by the Fortran community. The size and the complexity of S8 language presents high risk to:

- o Quality, completion, and acceptance of the standard
- o Consistent interpretation by implementers and validation organizations
- o Continued wide implementation and accessibility of Fortran, because of cost of producing compilers and producing viable implementations for small machines.
- o Reliability, efficiency and quality of implementation. For further discussion of this item see section on "Implementability of the language".

The immense size of the S8 language also makes it hard to master it for purposes of teaching, writing, reading and maintaining programs. Chapter 14 "SCOPE, ASSOCIATION, AND DEFINITION" of S8 is a good example of the complexity of S8 language

We acknowledge and appreciate the extraordinary effort that has gone into producing the S8.103. However, we recommend that the following extensions be taken out of the proposed standard and put in Appendix F of S8. Removal of these extensions will not only reduce the size and complexity of the proposed standard but it will also yield a language

needed and wanted by the Fortran community.

Extensions to be deleted from S8

- a. Derived Data Types.. This feature is definite requirement for languages used for non numeric computations. Thus it is not absolutely required in Fortran which essentially is the language for numeric computations. Even if it were to be included in future revisions of the FORTRAN, it should not be included in the currently proposed form. We, believe that derived data types should be introduced in future revisions(9X) of FORTRAN in conjunction with "pointers" and "Object oriented extensions".
- b. Modular definitions .. This perhaps is the feature which contributes most towards the complexity and size of the language. Although it does provide all the benefits summarized on Page ii of S8, but it also radically alters the compilation model of FORTRAN-77. We believe the separate compilation model of FORTRAN-77 is one of it's major strengths contributing towards it's simplicity. In conjunction with removing of this feature, we also recommend that "INCLUDE" facility be added to S8. Although the simple include facility does not replace all the functionality of the Modular definitions, but it does provide partial replacement and it does not change the "Separate compilation model of FORTRAN-77.
- c. New syntax of data declarations.. These extensions symbolize "new language" and show "how different the proposed Fortran is from FORTRAN-77". The specification of selectable precision and range floating point types is not the answer to problem of creation of portable programs sensitive to range and precision of floating point types. We believe the true portability of such programs can not be achieved in reality without the co-operating standards of floating point type machine representation standards. We therefore feel that allowing selectable range and precision attributes for floating point types is not worth the added complexity to the language.
- d. Internal and recursive Procedures

3. Implementability of the language.

The implementability of the language depends on the following items:

- Conceptual simplicity
- Size
- Compilation model .. independent vs dependent compilation

FORTRAN-77 has all the attributes that make it very easy to produce very efficient compilers at a reasonable cost. These very attributes of FORTRAN-77 contributed to its wide availability and therefore its success. Another contributing factor to the success of FORTRAN-77 was the degree to which it adopted the language extensions previously available on existing FORTRAN implementations.

The S8 language is conceptually complex, immensely large, and requires dependent compilation model. Thus ~~the~~ lacks all the conditions of success that FORTRAN-77 has. Based on our recent experience in producing a highly optimizing production grade FORTRAN compiler (FORTRAN-77 plus 8X Array processing extensions without IDENTIFY, RANGE and Assumed shaped arrays) we have arrived at the following conclusions regarding implementation of S8.103 language:

- i. Most likely it will be necessary to start from the scratch. Old existing compilers can not be modified easily to produce a good FORTRAN-8X compiler.
- ii. The number of lines of code in the new 8X compiler and the associated run time environment will be anywhere between 1.8-2.5 times that of FORTRAN -77.
- iii. The size of a good validation suite is expected to be at least 150 lines of code in addition to FORTRAN-77 validation suite.
- iv. Testing and validation activity alone will require at least a year.
- v. The compile time efficiency will be about .5 to .75 that FORTRAN -77
- vi. At least initially, the efficiency of the object code will be about .75 that of FORTRAN-77

Thus, it will be expensive and will take four to five years before good S8 language systems will be available to the FORTRAN community, Thus denying FORTRAN users timely access to the most need language enhancements

17.4 67

## RESPONSE TO ANIL LAKHWARA BALLOT

Your ballot contains three main points;

1. that the draft does not conform to the scope and program of work as stated in the project proposal,
2. that the language is too large and complex,
3. and that the language is not implementable.

### 1) Conformance to Project Proposal

The scope and program of work as stated in the project proposal has been followed in processing the draft standard. One of the statements is that X3J3 "carry out procedures to maintain the continuous responsiveness of the standard to industry needs". The new features, for example arrays and data structures, are in response to these needs. X3J3 has acted as a liaison group (as required) with the international group concerned with Fortran (Working Group 5), as well as maintained other liaisons with data base and graphics committees, and with the engineering and mathematical community.

Members of X3J3 have published technical reports on the interpretation of the current standard and provided maintenance of this standard, as charged. The criteria for evaluating proposals has been published in the minutes, we have reviewed and studied collateral standards, and we have had many forums to publicize our work in the past several years, keeping the user community informed of our activities. The draft document (S8) has been distributed widely several times a year over the past three years.

We (X3J3) believe that we have been in conformity with our charge, that we have kept the user community informed of our progress and been responsive to user, government and industry needs and that the draft standard is in compliance.

### 2) Size and Complexity of the Language

This comment contains phrases such as "immense size and complexity", "FORTRAN 77 is just a small subset", "radical departure from FORTRAN 77", and two distinct languages glued together". The comment then suggests that derived data types, modular definitions, specified precision, and internal and recursive procedures be removed from Fortran 8x.

Fortran 8x is a major extension of FORTRAN 77, but it is neither "immense" nor "radical". The extensions constitute approximately one-third of Fortran 8x, and great attention has been paid to making them as Fortran-like as possible. For example, the new loop control structure is an extension of the FORTRAN 77 DO-loop (though many believe a completely new and different loop structure would have been better).

Of the approximately one-third of Fortran 8x that is an extension over FORTRAN 77, about half is due to the array processing facilities and the numerical computation

enhancements. Both of these appear to be greatly desired by the scientific user community. They are designed as straightforward extensions of the corresponding base functionality in FORTRAN 77, and improve Fortran considerably in the numerical computation and array processing areas.

With regard to the specific suggestions for removal:

- a. **Derived Data Types.** Data structures are among the most widely desired features by the Fortran user community. The Fortran 8x derived data type is a very simple data structuring facility, similar to that in Pascal or C. Despite its name, it is not anything like the Ada derived data type facility.
- b. **Modular Definitions.** The MODULE/USE feature constitutes only a small part of Fortran 8x. It is superior to INCLUDE for defining entities (such as global data) that are to be shared by different program units, a functionality that is very important to Fortran. Though implementations may choose to use a dependent compilation model for processing modules, modules may be processed as "filtered INCLUDE".
- c. **New Syntax (for REAL).** The numerical community consensus is that the specified precision REAL facilities contained in Fortran 8x are most appropriate means of achieving portable, robust numerical computation capabilities. These facilities have been designed in close coordination with the numerical community, taking care to configure them as Fortran-like extensions to the FORTRAN 77 REAL and COMPLEX facilities.
- d. **Internal and Recursive procedures.** Internal and recursive (especially recursive) procedures have been implemented as extensions to FORTRAN 77. As implementations are simple and the functionality is available and wanted, it seems advantageous to standardize internal and recursive procedures in Fortran.

### 3. Implementability

The issue of implementability of Fortran 8x is still subjective since no implementations currently exist. With respect to your six points:

- i. It seems unlikely that a true "start from scratch" would be needed for a Fortran 8x compiler. All the FORTRAN 77 runtime library (mathematical and I/O) should be salvagable. New routines will be needed and some changes to existing ones will be required but certainly not a start from scratch. Most existing code generators should also be reusable. You are probably correct that Fortran 8x will require a new compiler front-end. However, even here, basic components such as lexical scanners, parser drivers, support routines etc. can be reused. Compiler front-end technology is well understood and is not conceptually complex.
- ii. Yes, a Fortran 8x compiler will be larger than a FORTRAN 77 compiler but X3J3 does not believe that the additional size will be out of proportion to the

capabilities offered.

- iii. The current validation suite is in excess of 100K lines of code. An increase of 150K lines of code seems reasonable given the additional features and the need to test interactions of new and old features.
- iv. Testing and validation may require a year, however, this is not an unusually long time for a compiler, even a FORTRAN 77 compiler. Typically, this effort overlaps the latter part of the development cycle and so its duration is not perceived to be a year.
- v. The only realistic basis for comparing FORTRAN 77 and Fortran 8x compile-time efficiency is in compiling existing FORTRAN 77 codes. On this basis, the committee feels that your estimates of 25-50% degradation are far too pessimistic. The general feeling of X3J3 is that compile-time efficiency on FORTRAN 77 codes will be minimally affected. If you look at Fortran 8x features, compile rates in lines per minute become meaningful. Some of the array constructs require many lines of FORTRAN 77 to express what you can write in a single line in Fortran 8x. In general, compile-time efficiency of new features cannot be compared with FORTRAN 77.
- vi. Just as for compile speed, we should compare FORTRAN 77 object code efficiency with Fortran 8x on an equal footing. Given that FORTRAN 77 is a subset of Fortran 8x, the committee does not expect implementations to suddenly produce worse code for the same source input. When voting on new features, the committee has regularly and carefully considered the issue of optimizability. The actual efficiency of code for new features will undoubtedly improve over time as technology progresses and implementors learn better ways to implement the new features.

The decision of when to bring out a new compiler is often more of a business decision rather than a question of implementability. The majority of FORTRAN 77 compilers came out three to six years after the standard was approved. There is always a period of several years migration from one standard to the next due to user inertia and compiler availability. The timeframe of four to five years for Fortran 8x seems quite reasonable looking at the history of FORTRAN 77.

In summary, X3J3 does not feel that Fortran 8x is unimplementable. Certainly the language is more complicated than FORTRAN 77, but not out of proportion to the capabilities offered.

# X3 Subgroup<sup>-62-</sup> Letter Ballot

Accredited Standards Committee  
Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

*SEE ATTACHED*

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME Bruce A. Martin  
(PLEASE PRINT)

SIGNATURE *Bruce A. Martin*

DATE: 1986 DEC. 31

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

19.0 71

103(\*)BAM-0  
ballot

GRUMMAN

M e m o r a n d u m

1986 DEC 1+

FROM: Bruce A. Martin  
TO: X3J3

SUBJECT: Letter Ballot on draft Proposed  
American National Standard  
Programming Language Fortran

REFERENCE: X3J3/S8.103 draft Fortran Revision

I am in favor of submitting the draft Fortran Revision X3J3/S8.103 to SPARC and X3, and then for Public Review soon after. I understand that X3J3 will process these Letter Ballot comments before sending it.

It is my opinion that this draft is in far better shape than the previous letter ballot draft, and that the language has improved since then. I believe that Fortran 8x, as defined by this draft, is now more or less acceptable -- although I wish the language could be made even smaller, more regular, and more extensible.

Nevertheless, there are some further improvements that should be made to the language, as well as numerous editorial fixes and readability enhancements to the document.

My comments for this letter ballot consist of discussions, complaints, laments, and rationales, PLUS a set of specific proposals (with text), all of which are attached.

19.1 . 72



## RESPONSE TO BRUCE MARTIN BALLOT

Your ballot consisted of many proposals, a few were accepted, and others were handled in another ballot. The remaining issues have already been discussed in the past several years and the full committee decided not to revise the draft. Some of these issues were part of the revision of the draft that resulted after the first ballot.

For example, extensive discussions were held on the nature of the "decremental" features. A change was made to include three categories of these features (deleted, obsolescent and deprecated) in order to soften the impact of changes for Fortran users. The current draft allows a staging process where users have at least 20 years to move to a better feature. The content of the obsolescent and deprecated features was discussed over a period of several meetings. Your ballot would add features to the obsolescent list, while other ballots would remove features from this list. We feel that the current draft should remain as defined.

Another example involves the CONTAINS statement. In a recent meeting, the issue of deleting CONTAINS altogether was revisited. The issue of renaming the keyword was also considered. X3J3 decided not to modify its position with regard to CONTAINS.

Also, changes were made to improve the description of fixed source form in Section 3. The committee did not agree with your suggested changes to the free source form.

Your ballot contains many excellent ideas that have been discussed over the past several years. Each of your issues was discussed again in a subgroup during ballot processing, and evaluated. Those issues that were approved were brought to the full committee. It is not possible to revisit issues in a never ending succession of meetings that endangers the possibility of ever getting a draft standard out to the public.

# X3 Subgroup Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, -with reasons

Comments (Additional pages of comments may be attached.)

COMMENTS ATTACHED

Mail to: Jeanne Adams, Chair, X3J3  
 NCAR - SCD  
 PO Box 3000  
 Boulder, CO 80307

NAME: ALEX MARYSAK  
(PLEASE PRINT)

SIGNATURE: *Alex Marysak*

DATE: JANUARY 2, 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

To: X3J2  
From: Alex Marusak  
Subject: Comments for Letter Ballot on X3J3/S8/103  
Date: December 31, 1986

I am voting "Yes" on submitting Fortran 8X to X3 for further processing as an American National Standard.

Item 7 -----

Page 12-12, lines 8-14, 12.5.2.7 CONTAINS Statement. I would much rather flag internal procedures with the word INTERNAL, as in:

INTERNAL REAL FUNCTION BLIVIT (A,B,C)

than precede such a procedure (or block of procedures) with the CONTAINS statement, especially since internal procedures cannot be nested.

If Fortran 8X is to include the CONTAINS statement, there should be examples of its usage.

Proposal: Remove CONTAINS and its associated baggage. Restore the declaration INTERNAL to designate internal procedures.

Item 8 -----

Page A-4, lines 10-13. By forcing standard modules to conform to Core Fortran, we are imposing stricter rules on supplementary standards than we impose on ourselves.

Item 9 -----

Page B-2, lines 5-8. Making the PAUSE statement obsolescent is silly.

Item 10 -----

Page C-14, lines 21-31. There has to be something wrong with the assertion that a standard-conforming program on one processor can become non-standard-conforming on another: "(On that processor, the program would also be viewed as not conforming to the standard because of the references to the nonstandard intrinsic procedure.)"...

Item 11 -----

Proposal: Move Appendix F back to the main body of the Standard.

I believe that the proposed Standard was not improved by moving a small list of items to an Appendix titled "REMOVED EXTENSIONS".

Item 12 -----

Proposal: 1) Move the Bit facilities of pages F-1 through F-12 back to the Standard. 2) Move the Variant Structure facilities of pages F-12 through F-14 back to the Standard. 3) Allow bit arrays in variant structures to be overlaid with other arbitrary data fields. 4) Relax the rules on definition/undefinition to allow simultaneous definition of variants in a variant structure, if one of the variants is a bit field.

Discussion: Many implementations of Fortran contain extensions to allow bit data and bit manipulations. Many of these same implementations allow simultaneous definition of equivalenced variables, even though this is not allowed in the current Standard. The new proposed Standard makes no allowance for these widespread practices.

This proposal would make available the power of bit manipulation facilities (including the replacement as appropriate of logical data with more compact bit data) and the added power of the variant structure. It would legalize a widespread extension of EQUIVALENCE, but confine its use to a specialized, modern data structure.

Item 13 -----

Proposal: Move the facility of Vector-Valued Subscripts, pages F-14 and F-15, back to the Standard.

Discussion: The gather/scatter mechanism in array processing is too important not to be included in a straightforward manner in the language.

III. CONCLUDING OBSERVATIONS

There is much to recommend in this proposed Standard. Array processing, derived types and derived operators, user-specified precision, and dynamic memory management are obvious examples. Some things are missing; bit facilities and pointers are obvious examples. The public will have its own favorites for addition and deletion.

Much has been said of the size of Fortran 8X; many believe it to be "too big." I am concerned not so much with the Standard's size as with its complexity. The document is complex and interrelated; many of the language features are complex and interdependent; and implementations may be complex and tedious to use.

A hallmark of Fortran has been separate and independent compilation. My biggest concern is that with the mandatory usage of interface blocks (to pass assumed-shape or allocatable arrays) and of modules (to define derived-type dummy arguments), independent compilation will be a thing of the past.

21.11

76

## RESPONSE TO ALEX MARUSAK BALLOT

Your editorial items have been discussed and accepted or rejected. Your substantive items 1-8 have been examined and either accepted or rejected by X3J3. Items 9-13 and your concluding observations are discussed below.

- **Item 9 Pause Statement.** There is no clear unique functionality provided by the PAUSE statement. Making it obsolescent is consistent with the definition of obsolescent (better ways exist in FORTRAN 77), and makes it possible for PAUSE to be removed from Fortran in the future. It does not mandate removal, however, if upon closer inspection removal seems inadvisable.
- **Item 10 Nonstandard Intrinsic Procedures.** This is a FORTRAN 77 provision, and therefore total conformance of Fortran 8x with FORTRAN 77 requires this statement.
- **Item 11 Appendix F.** The widely supported "post-ballot" draft, of which Appendix F was a major feature, was developed after the March 1986 negative letter ballot. The most common comment of those negative ballots was that certain features should be removed from the draft standard; Appendix F contains those features so agreed upon.
- **Item 12.1 BIT Data Type.** Moving the BIT data type to the appendix was part of the post-ballot draft. Complicating the entire BIT issue is that there appears to be no consensus on the bit-array versus bit-string functionalities, and that none of the BIT proposals provide for BIT stream I/O.
- **Item 12.2 Variant Structures.** Variant structures constitute an "equivalencing" mechanism, providing this functionality in the absence of the FORTRAN 77 EQUIVALENCE statement. Part of the post-ballot draft was to restore the EQUIVALENCE statement to unobsolescent form in the draft standard, making it feasible to simplify the standard language somewhat by moving variant structures to Appendix F.
- **Item 12.3 BIT Arrays in Variant Structures.** This is allowed in the BIT and variant structure facilities described in Appendix F.
- **Item 13 Vector-Valued Subscripts.** As part of the post-ballot draft, it was agreed to move vector-valued subscripts to Appendix F. One of the reasons for this decision was the problem of satisfactorily handling the many-to-one problem in assignments involving vector-valued array sections.

## Concluding Observations

Interface blocks, while in many cases most logically placed in modules, need not be so configured. The presence of an interface block in a procedure subprogram does not constitute "dependent compilation" any more than does, say, the specification `INTEGER X ;` in both cases, specification information is contained in the program using that information. When modules are used, specification information is located in a different program unit, which requires the availability of the module (or the equivalent information) during compilation of the using program unit.

Independent compilation remains an important feature of Fortran. Regardless of whether it is implemented as a "filtered INCLUDE" or by other methods, the Module/USE facility has no more impact upon the independent compilation of program units than does the "textual INCLUDE" feature commonly found in many compilers and preprocessors.

# X3 Subgroup Letter Ballot

Accredited Standards Committee  
Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986  
Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

(SEE ATTACHED COMMENTS)

Mail to: Jeanne Adams, Chair, X3J3  
 NCAR - SCD  
 PO Box 3000  
 Boulder, CO 80307

NAME LEONARD J. MOSS  
(PLEASE PRINT)

SIGNATURE Leonard J. Moss

DATE: 18 DECEMBER 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178
---

Tel: 202/737-8898 Fax: 202/638-4922
--

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

25.0 79

SLAC MEMORANDUM

December 17, 1986

To: X3J3  
From: Len Moss  
Sunnie Sund  
Subject: Comments accompanying X3J3 Letter Ballot of 21 November 1986

---

We have discussed the current draft of Fortran 8x (X3J3/S8/103) with a number of concerned Fortran users at SLAC. Based on these discussions, we are voting "NO" on the letter ballot dated 21 November 1986.

The primary reason for this vote is the large number of technical and editorial errors still remaining in the document. While it is true that the document has improved considerably since the previous letter ballot in March, we still find a great many mistakes. Most of these problems are relatively minor and would probably not obstruct a careful reviewer from understanding the draft standard; however, a substantial number remain which are potential sources of great confusion. In addition, there are a number of cases where the current text is quite clear but also wrong, that is, disagrees with what has been passed by the committee. For example, R220 says that a module subprogram may not contain even the one level of internal procedures allowed for external subprograms and main programs. This makes it impossible to simply embed an external subprogram in a module in order to make the interface explicit, and is not, I believe, what was intended when the nesting of internal procedures was prohibited as part of the compromise plan.

Most, if not all, of these problems are non-controversial, so if their total number were small we might consider voting YES with comments, in the expectation that they would all get fixed during letter ballot processing. However, the number is not small, and our rate of convergence on a readable document has been quite slow. We feel very strongly that X3J3 should not publish a document until the whole committee is convinced that it is understandable, and this document does not yet meet that criterion.

We are enclosing separate lists of problems and proposed fixes, including text where possible. This is not a complete list, however, because the errors are very numerous and the time we have available for reviewing the draft is limited.

In addition to the technical and editorial problems discussed above, we have several additional concerns about major features, or their lack, in the current draft. At recent meetings, it has become clear that X3J3 is not willing to consider such major modifications at this time, and so we will withdraw our previous objections to publishing a document without



these changes. However, these are still vital concerns for SLAC and, we believe, all of the high energy physics community. Therefore, we may eventually renew some or all of the following requests:

o POINTERS

This feature has been discussed at great length, so I will merely repeat that some sort of pointer or recursive data structure facility is extremely important for many applications in high energy physics and, I believe, many other areas.

o BIT DATA TYPE

SLAC was very disappointed to see what we believed to be a very good BIT data type removed from the standard.

o BIT CONSTANTS

The lack of a BIT data type now makes the need for additional facilities for manipulating bits even more urgent. Although many vendors have already adopted the bit functions described in ANSI/ISA-S61.1 (Industrial Real Time Fortran), relatively few have provided the hex and octal forms for constants in the related MIL-STD-1753. An alternative to providing such constants might be to provide intrinsic functions to convert between hex or octal character strings and integers (note that these must be intrinsic functions so that they can be used in PARAMETER statements to specify bit masks). Hex and octal format descriptors are also very desirable.

o BIT FUNCTIONS

In addition, we feel strongly that the bit-wise logical functions described in MIL-STD-1753 and ANSI/ISA-S61.1 (i.e., IAND, IOR, etc., including ISHFTC, IBITS, and MVBITS, which are only defined in the former document) should be incorporated directly into Fortran 8x.

o INTRINSIC OPERATOR OVERLOADING

We believe that allowing overloading of intrinsic operators presents the opportunity for the programmer to write code that is difficult to maintain and read. With the availability of user-defined operators, the programmer has functionality equivalent to intrinsic operator overloading while retaining readability and maintainability.

Lastly, we are concerned by the fact that there are a number of members who feel that there are still major substantive changes needed in the draft, but who are reserving these concerns for the time being in order to expedite public review. We ourselves have finally been persuaded to go along with this approach, but we are very uneasy with it. We feel it

is somewhat dishonest (X3J3 is, after all, actually voting to forward S8 as a finished standard), and believe that, in the long run, it will delay the adoption of the final standard, for the following reasons:

- o Once the document goes to X3, any further substantive changes require a 2/3 vote of the committee (X3/SD-2, Section 8.5, point (e)), with a documented attempt to resolve the negative votes.
- o Substantial changes will almost certainly require additional public comment periods.
- o Any really major changes may well hurt our credibility with the public, possibly reducing our chance of ever getting the standard approved.

## RESPONSE TO LEONARD MOSS BALLOT

Your ballot contains two principal concerns, and in addition expresses need for a number of new features. You say that 1) the document is not consistent with what we passed, and 2) that there is an expectation that new features may be added or deleted as a result of public review. A response to the request for each of the new features that you would like in the language follows addressing your general comments.

### Document Correctness

The document is now reasonably complete and reflects the actions of X3J3. Your careful review has contributed to making it correct.

### New Features during Public Review

The present document represents the position approved by the committee. We believe it is a viable standard as presented. Further work by the committee without statements from the user community in a formal public review process would not be productive. Many interested persons are waiting for the formal public review before carefully examining the draft. If the standard as it currently exists passes all further processing within ANSI and ISO, the committee feels that it would be an acceptable standard for Fortran. The 2/3 requirement for voting ensures that arbitrary changes would be difficult to make. A new standard is needed as soon as possible. The current ballot indicates that most of the members believe that this is so. We appreciate your editorial comments on the draft. All of them have been considered by the full committee and either accepted or rejected. Your specific items for addition to Fortran are addressed below.

### Request for Various New Features

X3J3 has spent considerable time discussing pointers and bits. Versions of a pointer facility have been discussed. However, none were adopted. The need for pointers as a memory management tool is met by automatic and allocatable arrays. Pointers for recursive data structures can often be simulated by using subscripts into an array of derived type. (This does require that a fixed (or at least allocated) array be defined and limits the number of structures available.) The bit facility was removed from the draft in an attempt to simplify the language and reduce its size and complexity.

Intrinsic operator overloading permits, but does not require, obscure coding. However, operators are currently "overloaded" in the sense that type coercions take place. Array references look like function references, character variable references or simple variable references, depending on the subscripts. Overloading of intrinsic operators allows a precedence to be associated with the operator. Without this precedence it would be necessary to use parentheses to force evaluation order. This would be different from ordinary Fortran notation and would seem to be as potentially error prone as "excessive" overloading could be.

# X3 Subgroup<sup>-75-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, -with reasons

Comments (Additional pages of comments may be attached.)

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME IVOR R. PHILIPS  
(PLEASE PRINT)

SIGNATURE I. R. Philips

DATE: 1/3/87

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

26.0  
84

I am voting no on the current version of the standard as defined in the February 1987 version of CS. The reasons are given, in detail, later. I consider the standard, in its present form, unlikely to be accepted by the public. Nevertheless, in order to make progress, it is time the public is permitted to make its comments on the proposed standard. I have, therefore, divided the changes I advocate into two classes: recommended and mandatory. I believe both classes of changes are necessary to make the standard acceptable to the public. However, if the changes that I have designated mandatory are made (they are given in detail below), I will change my no vote to a yes vote. This yes vote will, however, be a yes vote with comments. The comments will be a repeat of the reasons why I believe the proposed standard should be modified.

### Reasons for Changing the Proposed Standard

There are several reasons why I recommend changing the standard. They are:

- (1) The standard has deviated too far from FORTRAN 77.
- (2) Features have been added to "modernize" the language that now result in many duplicative and confusing features. This is demonstrated by the existence of 16 obsolescent and deprecated features. Because of this, the language will, at the Fortran 9X stage, face an unattractive choice: either the duplicated features will be removed, resulting in upward incompatibility; or they will remain, causing the language to remain clumsy indefinitely.
- (3) The language is considerably more complex than its predecessor. The language can no longer be readily "contained in the head". Because of this complexity, the language will change from one that is readily usable by nonprofessional programmers into one that is not. Nonprofessional users make up a significant part of the Fortran programmer community—it has been estimated that the typical aerospace engineer spends a quarter of the workday programming (in FORTRAN).
- (4) Because of the large increase in the size and complexity of the language, vendors are unlikely to implement the whole language in their initial releases. A quite probable scenario is that the standard will be implemented piecemeal, with different vendors implementing it, in differing orders, over a lengthy period of time. This will cause considerable portability problems.

### Recommended Changes

For the preceding reasons, I recommend that this committee (X3J3) implement two standards. This has been exhaustively discussed before. The first one would be upward compatible with FORTRAN 77, containing no obsolete, obsolescent, or deprecated features. I will refer to this language as Fortran A. The second one would remove, immediately, all the obsolescent and deprecated features of FORTRAN 77 and would incorporate most of the new features proposed for Fortran 8X. I say most, because, for example, some features are largely present to make existing FORTRAN 77 code execute efficiently. Thus the interface statement could be almost completely eliminated. I will refer to this language as Fortran B.

## The Contents of Fortran A

The principal criteria used for selecting features to be added to FORTRAN 77 were that they should blend harmoniously with the existing features of FORTRAN 77 and that they should be of benefit to most of the user community.

Specifically, this language would contain FORTRAN 77 plus the following features drawn from the February 1987 version of S8:

- Up to 31 character names including underscores
- The IMPLICIT NONE capability
- The MODULE and USE constructs
- Derived data types
- Non-overloaded derived-type-operators
- Automatic arrays
- Allocatable arrays (Such arrays could only be deallocated in the same program unit in which they were allocated. An unallocated allocatable array could not be passed as an actual argument. Allocated allocatable arrays could be saved, with deallocation taking precedence over the save.)
- The BIT data type—as modified in the “mandatory” section
- Internal procedures
- Recursive procedures
- Whole-Array operations
- Array-valued functions
- Elemental references—but only to intrinsic procedures
- Masked array assignments
- The CASE construct
- The DO CONSTRUCT with CYCLE and EXIT
- The intrinsic procedures with optional and keyword arguments—excluding those that make no sense for FORTRAN A: for example, PRESENT

- The ability to specify minimum precision requirements for REAL and COMPLEX data objects
- The new features of the Input/Output sections

### The Contents of Fortran B

Fortran B would be obtained from the February 1987 version of S8 by removing all the obsolescent and deprecated features. In addition it would be possible to (almost) totally eliminate the need for explicit interfaces by having simple rules about defining procedures prior to referencing them. There may be other features of a similar nature. While the two languages would differ, to facilitate conversion they should be kept as close as the guiding principles for each language permit.

### Mandatory Changes

The following set of changes generalize the DIT data type to be a bit string. Bit strings are considerably more user friendly to deal with than bit arrays of length one and provide more functionality of the type that people who work in the telemetry area inform me they need.

- 1) Replace lines 9-13, page F-1 with:

**F.1.1.1 Bit Data Type.** The bit type has a set of values composed of bit strings. A bit string is a sequence of bits, numbered from left to right 1, 2, 3, ... up to the number of bits in the bit string. The number of bits in the bit string is called the length of the bit string. The length is a type parameter and its value must be greater than or equal to zero. Each bit in the bit string has the value zero or one.

- 2) Add after line 23, page F-1:

R206a *bit-literal-constant* is B [ [ *bit-base* ] ] '*bit-string*'  
 or B ( / *bit-base* / ) '*bit-string*'

In the preceding syntax rule, the interior brackets are part of the syntax.

R206b *bit-base* is 2  
 or 4  
 or 8  
 or 16

R206b *bit-string* is [ *based-digit* ]...

R206c *based-digit* is 0  
 or 1  
 or 2  
 or 3  
 or 4  
 or 5  
 or 6  
 or 7

or 8  
or 9  
or A  
or B  
or C  
or D  
or E  
or F

If the optional [ *bit-base* ] or ( / *bit-base* / ) is missing, then *bit-base* is 2. A *based-digit* is a bit string whose definition is dependent on the *bit-base* according to the table F.1.1.1.1 (A missing entry means that particular *based-digit* cannot be used with the corresponding *bit-base*.)

Table F.1.1.1. Interpretation of Based Digits

<i>bit-base</i>	2	4	8	16
<i>based-digit</i>				
0	0	00	000	0000
1	1	01	001	0001
2		10	010	0010
3		11	011	0011
4			100	0100
5			101	0101
6			110	0110
7			111	0111
8				1000
9				1001
A				1010
B				1011
C				1100
D				1101
E				1110
F				1111

Note that *bit-base* must be a power of two. This restriction is necessary to keep the bit type as an abstract type without a predefined interpretation.

- 3) Delete the first two sentences in lines 1-2, page F-2.
- 4) Change line 10, page F-2 to:  
or BIT [ *bit-length-selector* ]
- 5) Replace lines 12-13, page F-2 with:

The BIT type specifier specifies that all objects whose names are declared in this statement are of intrinsic type bit (F.1.1.1). The bit length selector specifies the length of the bit objects. If this bit length selector is not specified, the length of the data object is 1.

R607a *bit-length-selector* is ([LEN =] *type-param-value* )

- 6) Add after line 17, page F-2:

F.1.1.3.1 Bit Substrings. A bit substring is a contiguous portion of a bit string. The following rules define the forms of a bit substring:



R607b *bit-substring* is *parent-bit-string* ( *substring-range* )  
 R607c *parent-bit-string* is *scalar-variable-name*  
 or *array-element*  
 or *scalar-structure-component*  
 or *scalar-constant*

Constraint: *parent-bit-string* must be of type bit.

The length of a bit substring is the number of bits in the substring and is  $\max(\text{ending-point} - \text{starting-point} + 1, 0)$  where *ending-point* and *starting-point* are the starting and ending points respectively of *substring-range*.

Let the bits in the parent bit string be numbered 1,2,3,...,n, where n is the length of the parent bit string. Then the bits in the substring are those from the parent string from the starting point and proceeding in sequence up to and including the ending point. Both the starting point and the ending point must be within the range 0,1,2,...,n unless the starting point exceeds the ending point, in which case the substring has length zero.

If the parent is a variable, the bit substring is also a variable. If the parent is an array section, the substring is an array of the same shape as the array section and each element is the designated substring of the corresponding element of the array section.

- 7) In line 13, page F-4, change "character" to "character or bit".
- 8) In line 25, page F-5, insert "//," ahead of ".BAND.".
- 9) In line 29-30, page F-5, delete " and the ... <>".
- 10) Insert after line 7, page F-6:

B                      B                      B

- 11) Insert after line 15, page F-6:

B                      B                      L

- 12) Replace line 31-37, page F-6 by:

F.1.1.4.10 Bit Intrinsic Operations. Let x1 and x2 be scalar bit operands with respective lengths n1 and n2. The categories of bit intrinsic operations are:

concatenation	//
unary masking	.BNOT.
binary masking	.BAND., .BOR., .BXOR.
relational	.EQ., .NE., .GT., .GE., .LE., .LT., == , <> , > , >= , <= , <

Let .op1. be a relational bit intrinsic operation and let .op2. be a binary masking bit intrinsic operation. Then

x1 .op1. x2	is logical,
x1 .op2. x2	is bit of length $\max(n1,n2)$ ,
x1 // x2	is bit of length $n1 + n2$ ,
.BNOT. x2	is bit of length n2.



-82-

or B[ [ *bit-base* ] ]w[.m]  
 or B( / *bit-base* / )w[.m]

- 21) Add after line 28, page P-9:

The inner brackets for the bit edit descriptor are part of the syntax.

- 22) Replace lines 29-36, page F-9 with:

**F.1.1.7.2. B Editing.** The Bw, Bw.m, B[k]w, B[k]w.m, B(/k/)w, and B(/k/)w.m edit descriptors indicate that the field to be edited occupies w positions. The specified input/output list item must be of type bit. The specifiers BZ and BN are in effect for bit editing.

On input, Bw.m, B[k]w.m, and B(/k/)w.m edit descriptors are treated the same as Bw, B[k]w, and B(/k/)w respectively.

Bw.m and Bw are treated the same as B[2]w.m and B[2]w respectively.

The value of k determines the editing according to Table F.1.1.1.

On input, Table F.1.1.1 is used to convert input *based-digits* to bit strings.

Define j so that  $k = 2^j$ . An output list item whose length is not divisible by j is extended on the left with zeros so that the new length is divisible by j. The output list item is then divided up into contiguous substrings of length j and each substring is edited according to Table F.1.1.1.

Output is extended by blank characters on the left if necessary to fill w positions. If B[k]w.m, B(/k/)w.m, or Bw.m is in effect, the output has at least m digits and, if necessary, has leading zeros. The value of m must not exceed the value of w. If m is zero or not present and if the output list item has zero length, then the output field is filled with blanks.

**F.1.1.7.3 List-Directed and Namelist Input/Output.** List-directed and namelist input/output is restricted to a bit base of 2. The format for such input/output is Bw where w is the length of the bit string.

- 23) Page F-9, line 36: insert, after this line:

Page 12-3, line 2: change "type" to "or bit types".

- 24) Replace lines 37-42, page F-9 with:

**F.1.1.8 Bit Functions.** The elemental functions LBIT and BITL convert between bit (of length one) and logical type. The elemental function BIT\_INDEX finds the location of a bit substring within a bit string. The elemental functions IBITLR, BITLR, IDITRL, BITRL convert between bit and integer counting from the left or the right.

The transformational function BITTRIML trims zero bits from the left of a bit variable. The inquiry functions MAXBITS, BIT\_LEN, and BIT\_SIZE return information about converting bit to integer, the length of a bit string, and the number of bits needed to TRANSFER a scalar quantity to bit.

- 25) Page F-9, line 36: insert, after this line:

Page 12-3, line 2: change "type" to "or bit types".

-83-

- 26) Insert above line 1, page F-10:

BIT_INDEX(B,SUBSTRING,BACK)	Starting position of a bit substring
Optional BACK	
BIT_LEN(B)	Length of a bit entity
BIT_SIZE(X)	Equivalent bit length of an entity
BIT_TRIML(B)	Remove leading zero bits

- 27) Lines 1-11, page F-10, change all "array" to "string"
- 28) Line 16, page F-10, change "bit" to "bit of length one"
- 29) Line 22, page F-10, change "array" to "string"
- 30) Line 23, page F-10, change "Transformational" to "Elemental"
- 31) Lines 28-29, page F-10, change "bit array ... elements" to "scalar bit with length SIZE"
- 32) Lines 30-35, page F-10, replace with:

**Result Value.** The result is a bit type containing the binary representation of the argument. The rightmost bit of the result will contain the least significant bit of the binary representation. Zero extension or truncation will take place at the left end of the result as necessary. IBITLR(BITLR(J)) must have the value J for all non-negative integers J. BITLR(IBITLR(B),BITLEN(B)) must have the value B for any bit string B for which BITLEN(B) ≤ MAXBITS(1).

- 33) Line 36, page F-10, change "[ ... ]" to "B'000101'"
- 34) Line 39, page F-10, change "array" to "string"
- 35) Line 40, page F-10, change "Transformational" to "Elemental"
- 36) Lines 5-6, page F-11, change "bit array ... elements" to "scalar bit with length SIZE".
- 37) Lines 7-12, page F-11, replace with:

**Result Value.** The result is a bit type containing the binary representation of the argument. The rightmost bit of the result will contain the most significant bit of the binary representation. Zero extension or truncation will take place at the right end of the result as necessary. IBITRL(BITRL(J)) must have the value J for all non-negative integers J. BITRL(IBITRL(B),BITLEN(B)) must have the value B for any bit string B for which BITLEN(B) ≤ MAXBITS(1).

- 38) Line 13, page F-11, change "[ ... ]" to "B'101000'"
- 39) Line 15, page F-11, change "array" to "string"
- 40) Line 16, page F-11, change "Transformational" to "Elemental"
- 41) Lines 17-18, Page F-11, replace with:

**Argument.** B must be of type bit satisfying BITLEN(B) ≤ MAXBITS(1).

- 42) Lines 20-24, page F-11, replace with:

**Result Value.** The result has value equal to the non-negative integer represented by the bits in B with the rightmost bit being the least significant bit of the result. IBITLR(BITLR(J)) must have the value J for all non-negative integers J. BITLR(IBITLR(B),BITLEN(B)) must have the value

-84-

B for for any bit string B for which  $\text{BITLEN}(B) \leq \text{MAXBITS}(1)$ .

- 43) Line 25, page F-11, change "[ ... ]" to "B'0101'"
- 44) Line 27, page F-11, change "array" to "string"
- 45) Line 28, page F-11, change "Transformational" to "Elemental"
- 46) Lines 29-30, Page F-11, replace with:

Argument. B must be of type bit satisfying  $\text{BITLEN}(B) \leq \text{MAXBITS}(1)$ .

- 47) Lines 32-36, page F-11, replace with:

**Result Value.** The result has value equal to the non-negative integer represented by the bits in B with the rightmost bit being the most significant bit of the result.  $\text{IBITRL}(\text{BITRL}(J))$  must have the value J for all non-negative integers J.  $\text{BITRL}(\text{IBITRL}(B), \text{BITLEN}(B))$  must have the value B for for any bit string B for which  $\text{BITLEN}(B) \leq \text{MAXBITS}(1)$ .

- 48) Line 27, page F-11, change "[ ... ]" to "B'1010'"
- 49) Line 1, page F-12, change "bit" to "bit of length one"
- 50) Line 7, page F-12, change "array" to "string"
- 51) Line 12, page F-12, change "array" to "string"
- 52) Line 14, page F-12, change "BIT" to "BIT of length one"
- 53) Append after line 13, page F-12:

#### F.1.1.8.8 BIT\_INDEX(B, SUBSTRING, BACK).

Optional Argument. BACK

Description. Returns the starting position of a bit substring within a bit string.

Kind. Elemental function.

Arguments.

B must be of type bit.

SUBSTRING must be of type bit.

BACK (optional) must be of type logical.

Result Type. Integer.

Result Value.

*Case (i):* If SUBSTRING occurs within B, the value returned is the minimum value of I such that  $B(I : I + \text{BIT\_LEN}(\text{SUBSTRING}) - 1) == \text{SUBSTRING}$ ; otherwise, zero is returned. Zero is returned if  $\text{BIT\_LEN}(B) < \text{BIT\_LEN}(\text{SUBSTRING})$  and one is returned if  $\text{BIT\_LEN}(\text{SUBSTRING}) = 0$ . The default value of BACK is .FALSE. and its inclusion is optional when processing starts with the first character of B.

*Case (ii):* If B is to be processed starting with the last bit, BACK must contain the

-85-

logical value .TRUE. If SUBSTRING occurs within B, the value returned is the maximum value of I such that  $B(I : I + \text{BIT\_LEN}(\text{SUBSTRING}) - 1) == \text{SUBSTRING}$ ; Otherwise, zero is returned. Zero is returned if  $\text{BIT\_LEN}(B) < \text{BIT\_LEN}(\text{SUBSTRING})$  and  $\text{BIT\_LEN}(B) + 1$  is returned if  $\text{BIT\_LEN}(\text{SUBSTRING})$  is zero.

Example.  $\text{BIT\_INDEX}(B'0010110', B'101')$  has value 3.

#### F.1.1.8.9 BIT\_LEN(B).

Description. Returns the length of a bit entity

Kind. Inquiry function.

Argument. B must be of type bit. It may be scalar or array valued.

Result Type and Shape. Integer scalar.

Result Value. The result has value equal to the number of bits in B if it is scalar or in a component of B if it is array valued.

Example. If B is declared by the statement

```
BIT(LEN=7), ARRAY(100) :: B
```

$\text{BIT\_LEN}(B)$  has value 7.

#### F1.1.8.10 BIT\_SIZE(X).

Description. Returns the minimum length of a bit entity that will always permit an exact transfer of elements of the type of X to bit using TRANSFER.

Kind. Inquiry function.

Argument. X may be of any implicit type or derived type with a determinant size (any non-parameterized derived type or any realized parameterized derived type. X may be scalar or array valued.

Result Type and Shape. Integer scalar.

Result Value. The result value is the smallest integer I for which  $\text{TRANSFER}(\text{TRANSFER}(Y), B), Y == Y$  for all values Y having the type of X where B is a bit entity of length I.

Example. The value of  $\text{BIT\_SIZE}(1)$  is processor dependent. On processors that have 32 bit integers, the value would be 32.

#### F.1.1.8.11 BIT\_TRIML(B).

Description. Returns the argument with leading zero bits removed.

Kind. Transformational function.

Argument. B must be of type bit and must be a simple variable or array element (not an array or array section).

Result Type and Type Parameters. Bit with a length that is the length of B less the number of leading zeros in B.

-86-

**Result Value.** The value of the result is the same as B except that all leading zero bits are removed. If B contains no nonzero characters, the result has zero length.

**Example.** BIT\_TRIML(B'001101') has value B'1101'.

The next set of changes is "human engineering". The overloading of operators and names, while of benefit to mathematicians and experienced programmers, permits the writing of very obscure code—of the type that can cause severe difficulties to inexperienced or nonprofessional programmers. Therefore, to alleviate this situation the following set of changes is proposed.

54) Page 2-3, line 4: add, after this line:  
or *overload-stmt*

55) Page 5-16, line 2: add, after this line:

**5.5 OVERLOAD Statement.** An OVERLOAD statement specifies that a group of defined operators (7.1.3) and procedures (12.5.5) may be used or referenced, respectively, in an overloaded manner. In order for a program unit to use an overloaded operator in an expression, the operator must appear in an OVERLOAD statement in that program unit. In order for a program unit to reference an overloaded procedure the procedure name must appear in an OVERLOAD statement in that program unit.

548	<i>overload-stmt</i>	is OVERLOAD	<i>overload-list</i>
549	<i>overload</i>	is	<i>overloaded-procedure</i> or <i>overloaded-operation</i>
550	<i>overloaded-procedure</i>	is	<i>name</i>
551	<i>overloaded-operation</i>	is	( <i>operand-type, defined-operator, operand-type</i> ) or ( <i>defined-operator, operand-type</i> )
552	<i>operand-type</i>	is	INTEGER or REAL or DOUBLE PRECISION or COMPLEX or CHARACTER or LOGICAL or <i>type-name</i>

## RESPONSE TO IVOR PHILIPS BALLOT

The main points in your ballot are 1) dividing the language into two separate languages, and 2) adding two new features to the language, as in your BIT String Data proposal and the OVERLOAD statement. These are addressed in the following notes. Your 11 editorial comments have been considered by the committee and either rejected or accepted.

### Divide Fortran into Two Separate Languages

X3J3 has considered at great length and in great depth, the issue of a "partitioned" Fortran. One form of partitioning considered (during 1985-1986) is that of two separate languages. Another form, that of a "core + modules" architecture, was considered earlier, adopted, and subsequently rejected. In both cases multiple versions of standard Fortran appeared to impede program portability--the very thing that standardization is supposed to enhance! In the case of two languages (separately standardized and hence officially unrelated), the question of which is the "real" Fortran appeared to be a further frustration to the user community. As a result of these factors, X3J3's carefully considered conclusion is that Fortran 8x should be a single language.

### Add BIT String Data

Bit level processing has been discussed by X3J3 for many years. There are at least two major options for providing bit processing: bit strings and bit arrays. To date, and after considerable discussion, there has not been a lasting consensus for the form of a bit string facility. The current bit array facility in Appendix F was adopted on the basis of maximum compatibility with the array facilities, and, conversely, operations on bit arrays provide much bit-string type capability.

Two additional factors appear to prevent complete consensus on any comprehensive bit processing facility in Fortran. First is that data representation at the bit level is inherently variable and hence corresponding software is nonportable and/or inefficient. Second is that none of the bit proposals presented thus far have been able to define bit stream I/O satisfactorily, which seems to be required in order to support true bit data processing. For these reasons, X3J3 has not considered the bit string facility (which is more complex than the bit array facility in Appendix F) to be advisable.

### Add an OVERLOAD Statement

Although an OVERLOAD statement could improve readability, it is not necessary to disambiguate any instance of operator or procedure overloading. As it is unnecessary, implementations could extend the language to relax the requirement, and this would impede portability. Therefore, your proposal was rejected.



# X3 Subgroup<sup>-88-</sup> Letter Ballot

Accredited Standards Committee  
X3 Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

# FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME Richard Ragan  
(PLEASE PRINT)

SIGNATURE *Richard R. Ragan*

DATE: 12/29/86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

27.0  
97

My ballot comments are divided into two parts. The first part identifies the location in the document and the comment. The second section of the comments is the proposed text required to resolve the comment.

## Comments

---

1. Page 1-3, line 28,29: Is it necessary to embed blanks in the constants here since the subject of insignificant blanks has not yet come up.
2. Page 1-3, line 38. Item (1) is not true. There are syntactic classes that end in -stmt that are not Fortran statements. Known problem classes are R212, R214, R221, R223, and R822.
3. Page 1-3, line 44: R220 is wrong
4. Page 1-5, line 12: core conforming is not yet defined. This needs a forward reference
5. Page 2-1, line 15: "internal-procedure-part" is not defined
6. Page 2-7, line 18: comparison for equality is no longer defined for derive-types.
7. Page 3-4, line 10: We should give some thought now to how a future standards committee can add .xxxx. operators without invalidating standard-conforming programs that have been written. The Fortran 77 standard handles this for new intrinsic functions by requiring the use of an EXTERNAL statement. We should be sure that we have left adequate means for the addition of future operators to the language.
8. Page 4-7, line 1: Did we consider allowing the RANGE attribute for components of a structure? It appears that it could be useful. I am not prepared to propose it without some subgroup discussion however.
9. Page 4-7, line 5: non-precision type-parameter expression needs a forward reference.
10. Page 4-9, line 6: The subject of array constructors does not seem to be a proper subtopic of §4.4.3 since most arrays are not of derived-type. Is a heading missing?
11. Page 4-9, line 15: The form

**RESPONSE TO RICHARD RAGAN BALLOT**

All your suggested changes to the document to improve readability and consistency have been considered by X3J3 and either accepted or rejected.

# X3 Subgroup<sup>-91-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986  
Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

*See attached pages*

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME J. K. REID  
(PLEASE PRINT)

SIGNATURE *J. K. Reid*

DATE: 31 Dec 86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

28.0

100

## RESPONSE TO JOHN REID BALLOT

All your editorial items have been considered and accepted or rejected by X3J3. Your glossary which was included as a ballot commentary has been accepted for inclusion as an appendix to the draft.

# X3 Subgroup<sup>-93-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

Mail to: Jeanne Adams, Chair, X3J3  
 NCAR - SCD  
 PO Box 3000  
 Boulder, CO 80307

NAME Valerie G. Bowe Lawrence Rolison  
(PLEASE PRINT)

SIGNATURE Valerie G. Bowe Lawrence Rolison

DATE: Dec. 31, 1986

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers  
 Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
 Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

Unisys has cast a NO vote after much deliberation because we feel that the aggregate addition of the "modern" programming language features has been a disservice to the language. We can not in good conscience vote YES just to let the public have a look at the proposed standard. A YES vote, far more importantly, would be a statement that we are satisfied with the proposed standard in its current form and with the language it describes. We are not.

We feel that the addition of the entire package of modern programming language features has warped the language nearly beyond recognition. It is our opinion that the committee has spent too much time in language design at the expense of standardizing current practice. We fear the proposed language will no longer serve the community for which it was originally designed. The promoters of the modern language additions, on the other hand, contend that the language already fails to serve that community and that if the committee does not make drastic changes to the language, users will abandon it in favor of a modern language such as Ada. Our reply is: Let them use Ada. To paraphrase Mark Twain, we believe the predicted demise of FORTRAN has been greatly exaggerated.

The hallmark of FORTRAN has been that it is a reasonably compact language that may be implemented via fairly small compilation/run-time systems, and that its generated code, particularly numeric code, runs extremely fast. We are concerned that the committee has not given sufficient consideration to the impact of the new 8x features on compilation speed, compiler size, and run-time execution speed. There were presentations and long discussions at the January, 1986, X3J3 meeting about the size of the language that eventually influenced the development of the compromise. However, the features that are the greatest impact on the compilation and execution environments are still in the language. Such features are as narrowly focused as the implementation of modules and as general to the language as the addition of a significant number of semantic rules.

Our language design/implementation concerns include, but are not limited to, the following:

- \* The concept of modules and the USE statement introduce dependent compilation into the language. Dependent compilation makes it difficult, if not impossible, to implement a one-pass compiler. We believe that modules may be simply a case of overkill and that standardization of the current common practice of using an INCLUDE statement would have been satisfactory.
- \* The 8x document describes a language that is quite complex. Other complex languages, like Ada for example, benefit from grammars that lend themselves to be parsed by some form of parser generator. Because of the context-sensitive and ambiguous syntactic constructs in 8x, parsing will be difficult. For example, 8x introduces the derived-type constructor which could have the same syntactic structure as a function reference or an array reference. The increased semantic checking to resolve this and other new constructs will place further demands on a compiler. Parsing is also made more complex by the addition of a second source form.
- \* In many ways 8x describes a modern language that attempts to provide a virtual machine to the programmer. Yet in many other ways, it is still tied to the hardware (and probably rightfully so if one of the goals of this standard is to maintain upward compatibility with FORTRAN 77). 8x can therefore never provide a complete virtual machine model. Complex languages like 8x have a better chance of acceptance and usage as a modern language if they are based on a consistent formal model. It may be the absence of this model that accounts for critics stating that the language has no focus.
- \* The readability (and therefore maintainability) of FORTRAN programs is degraded by some new features such as intrinsic

operator overloading and user defined assignment subroutines.

- \* Many array-valued expressions will require the use of temporary arrays to hold array values during the course of expression evaluation. Optimizing array-valued expressions to minimize the production of such temporaries places further demands on the compiler. Such optimizations may be crucial to certain architectures in order to avoid significant run-time performance degradation.
- \* Most implementations will likely use some form of array descriptor to implement array argument interfaces, allocatable arrays, and alias arrays. The concepts of building and manipulating these (possibly quite complex) array descriptors will degrade both compiler and run-time performance.
- \* Memory management will degrade run-time performance due to heap and stack management overhead.
- \* REAL(\*,\*) has been greatly simplified from its original concept but will still present complications to implementors and is likely to produce many surprises in the user community. We recognize that a compiler may only have to generate "a few" copies of a given routine but this still may cause severe problems if each copy is a large source module. Consider, for example, "a few" copies generated of a source module, say, 10,000 lines long. It is easy for a nonvendor committee member to say "Just tell your customer to break up the source module." but it is very difficult to actually deal with an influential customer in such a manner in today's highly competitive computing marketplace.
- \* Keyword argument mapping and optional argument support are other entirely new (and questionable) concepts in 8x that we believe will produce complexity in compilers beyond the usefulness (and likely actual usage) of such features.

In summary, we have found virtually no new feature in 8x that will enhance either compiler or run-time performance. In actuality, 8x implementations must as a necessity be larger and slower than 77 implementations. We are concerned that the committee has created a negative incentive for the average user to move to an 8x implementation. As such, the committee has done both the language and the user an injustice.

We find the following passages from two NO votes on the first letter ballot to still be pertinent and provide excellent summaries:

The number of changes between FORTRAN 77 and S8 is much greater than for any previous revision of FORTRAN... The motivation appears to be an attempt to actually redefine FORTRAN as a new, all-purpose programming language. This is neither needed, nor wanted, by [the vendor] and its user community.

Kevin W. Harris

It is certainly not obvious that the FORTRAN language must evolve into an Ada-like (i.e. "modern") language in order to survive. Ada was designed to meet a certain set of requirements which are not necessarily the same requirements facing FORTRAN users...

One of the strengths of FORTRAN is the very fact that it is a fairly low level language and not very far removed from the underlying machine architecture. To try and design another "all purpose" (Algol, PL/I, Ada) language within the framework of Fortran will lose the support and advantages that Fortran [sic] enjoys today.

R. W. Weaver



We realize that we can not turn back the hands of time and undo the work of the committee over the last 9 years. With this in mind, we are not going to present a list of demands or requirements that must be met in order for us to cast a YES vote. We disapprove of the language that has been designed by X3J3 but we feel it would be counterproductive at this late stage to demand that the committee abandon the work that has been done and begin anew. We applaud the chance that public exposure may provide valuable input to the committee, but we must vote NO to express our disapproval of the alterations made to the language.

In the spirit of constructive criticism and cooperation, we offer the attached list of editorial corrections, minor technical changes that we believe will enhance the language in its current state, and questions/comments/suggestions for clarification and consideration by the committee.

## RESPONSE TO BOWE/ROLISON BALLOT

X3J3 agrees that predictions of Fortran's demise are premature. Fortran continues to serve the needs of the numeric computing community better than any of its more "modern" competitors. However, X3J3 believes that this community would be still better served by a revision of Fortran that provides "modern" language features in addition to strong support for numerical computing. X3J3 has endeavored to minimize the negative impact of these features on characteristics such as execution speed. In addition, X3J3 feels that modest processor performance penalties can be accepted, especially if they are under the programmer's control, if it is the result of adding functionality.

X3J3 appreciates your efforts in reviewing S8, as evidenced by the many useful changes you have suggested. Each of these changes has been considered by X3J3 and either accepted or rejected. Your principal technical issues and questions are addressed below.

### One-pass Compilation

Whether or not Fortran 8x can be implemented in a one-pass compiler is not viewed as a serious issue, since commonly used optimization technology also requires more than one pass.

### Complexity

FORTTRAN 77 is also context-sensitive and thus not particularly amenable to the use of automatic parsing tools. Since Fortran 8x contains all of FORTRAN 77, it is necessarily also context-sensitive. X3J3 believes that the new features in Fortran 8x do not significantly complicate this problem.

### Virtual Machine Model

FORTTRAN 77, like FORTRAN 66 before it, is based on a relatively low level machine model. The new features in Fortran 8x have been based on a somewhat higher level machine model in order to extend the range of machines whose characteristics can be effectively exploited in a Fortran implementation. This inconsistency in machine model is an unavoidable consequence of an evolving language with an evolving machine model.

### Readability

X3J3 agrees that if an intrinsic operator or assignment is overloaded with a procedure that does not reasonably parallel its intrinsic meaning, readability and maintainability will be impaired, but believes that if such an overload is consistently defined, readability and maintainability will be enhanced.

## Temporary Arrays

X3J3 recognizes that the evaluation of array expressions will, in many cases, require the use of temporary arrays for the storage of intermediate results. The management of these temporary arrays is a new burden on the processor which may be especially significant if the processor attempts to minimize the cumulative size of these temporaries. On the other hand, if array expressions were constrained to eliminate the necessity for creating such temporaries, programs performing these computations would be forced to store intermediate results in explicitly declared arrays. The cumulative size of these explicit "temporary arrays" would be at least as great as that of the compiler-generated temporary arrays and would prove more difficult to eliminate or reduce through optimization. Management of these "temporary arrays" could be left to the explicit control of the program, but this would tend to preclude optimizations based on details of the evaluation strategy such as the size of the hardware vector register.

## Array Descriptors

Although the calculations necessary in isolation to access an array element through a descriptor are essentially identical to those necessary to access an element of a FORTRAN 77 array, the greater generality of descriptors provides the processor with less implicit information about the operands of these calculations, reducing opportunities for optimization and thus degrading processor performance. In recognition of this, X3J3 has gone to great lengths to ensure that a processor can distinguish between arrays which, in effect, must be implemented with descriptors and those which can be implemented as in FORTRAN 77. Thus, a processor can avoid degraded performance except where the generality of new features demands. This "degraded" performance is typically no worse than would have resulted from the use of "descriptor-free" features in a manner to simulate the same generality.

## Memory Management

The cost of managing stack storage is normally quite small, but if the implementor judges that even this cost is too great, stack storage can be used only for automatic variables and recursive procedures. Thus, the cost of this storage management would be borne only by programs that use features which require it. The cost of heap storage management can be similarly distributed. Thus, the only storage management costs imposed on all users are overhead cost such as the cost of initializing the storage management system.

## Real(\*,\*)

The use of REAL (\*,\*) arguments creates a kind of generic procedure intended to be reused with arguments of differing precision or exponent range. It seems unlikely that a 10,000 line source module fits in this category. No experienced programmer should be surprised if performance is degraded by the use of a feature that is more general than the problem demands.

## Keyword Argument Mapping

Keyword argument mapping is the result of three user requests/needs: order-independent actual argument lists, optional arguments, and improved procedure call readability. These requirements are all met with keyworded actual arguments. This feature is quite simple, and should therefore provide the user with the desired optional functionality without adding much complexity to the implementation of Fortran 8x.

## Summary

Clearly, Fortran 8x implementations will be "larger than" FORTRAN 77 implementations, as FORTRAN 77 is a subset of Fortran 8x. Presumably such larger implementations will have somewhat lower compile-time performance. Run-time performance of FORTRAN 77 programs need not be lower, however, and performance improvements of FORTRAN 77 programs (over corresponding FORTRAN 77 provisions) can certainly be expected in things like array processing (or vector and parallel architectures) and internal procedures (if expanded in-line). To suggest that users do not want Fortran 8x is to ignore

- a. the results of numerous user surveys and
- b. the fact that Fortran 8x is the result of what may be the most open language standards process ever.

X3J3 has tried very hard to obtain widespread user input and develop a Fortran 8x that most satisfies the needs of Fortran programmers. Though top performance is a very important part of those needs, higher level language, added functionality and improved portability are increasingly important factors. Increasing volumes of application backlogs, increasing maintenance costs, and greater portability demands are widely affecting the Fortran user community and are not alleviated by the continuing dramatic advances in hardware capabilities. These hardware advances (together with improving compiler techniques) will help to make Fortran 8x compiler performance acceptable to the user community. It is the features of Fortran 8x, however, that will affect the users' abilities to develop applications more quickly, reduce maintenance costs, and obtain improved portability with acceptable performance. Evolution of Fortran, though not without costs, is the most cost effective way to achieve these benefits, not only for the user, but almost certainly for the vendor as well. Fortran 8x represents such an evolution, shaped primarily by Fortran user input and needs.

# X3 Subgroup<sup>-100-</sup> Letter Ballot

Accredited Standards Committee  
X3. Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

# FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

For comments see attached sheets (103)-525-1

X3 SECRETARIAT  
86 DEC 24 AM 1:10

Mail to: Jeanne Adams, Chair, X3J3 NAME LAWRIE SCHONFELDER  
 NCAR - SCD (PLEASE PRINT)  
 PO Box 3000 SIGNATURE Laurie Schonfelder  
 Boulder, CO 80307 DATE: 10-DEC-86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

31.0  
109

To: X3J3  
From: Laurie Schonfelder  
Subject: Letter Ballot Comments  
References: S8.103 February 1987

**Introduction:** This set of comments which is attached to an affirmative ballot is divided into two parts. The first is a set of trivial editorial corrections. These are concentrated in the sections for which I have had subgroup responsibility. The second part includes some more substantive commentary regarding areas where I think there are still a number of defects in either the functionality or its description. In each case correcting the defect would make a substantive change to the language which in my opinion would improve it. However none of these defects do I consider of sufficient importance to warrant rejecting the sending of the document for public comment at this stage.

**Editorial Comments:** The general quality of the document has improved markedly over the past few versions. The editorial subgroup are to be congratulated on the excellent work they have done. There are nevertheless a few remaining corrections that I have noted.

**Substantive Deficiencies:** The language as now defined has a number of infelicities, irregularities, and deficiencies which if corrected would greatly improve it. None of these are sufficiently serious to cause me to submit a negative vote on sending the current document, or one corrected as a result of processing the ballot, for public review. However, I feel that these comments should be recorded and considered during the ballot processing.

The problems I see are:

- 1) Irregularities in both the current definition of structure constructors and intrinsic type conversion functions.
- 2) Inconsistent use of long or short names for keywords and intrinsic functions. The long names are in many cases far too long.
- 3) Inadequate description and highly irregular detailed functionality of the RANGE array sectioning facility.
- 4) Inadequate description and highly irregular wholly deficient functionality of the ALIAS/IDENTIFY facility.
- 5) Restrictions on the ALLOCATABLE attribute and dynamic storage management which remove a large part of its functionality.
- 6) The restriction which prohibits the selection of array components from arrays of structures.

Time pressure does not permit me to provide complete proposals with text to correct all these problems; particularly 4) and 5) which are really symptomatic of a more substantial deficiency, viz., the lack of proper pointers. Some of these issues rightly therefore must be left for consideration during the public comment period. I am including text however

**RESPONSE TO LAWRIE SCHONFELDER BALLOT**

All your items have been considered by X3J3 and either accepted or rejected. The substantive issues mentioned in your ballot have been considered, or have been rejected in the compromise plan developed after the first letter ballot.

# X3 Subgroup<sup>-103-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

# FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with-comment
- Negative, -with reasons

Comments (Additional pages of comments may be attached.)

Comments enclosed

Mail to: Jeanne Adams, Chair, X3J3  
 NCAR - SCD  
 PO Box 3000  
 Boulder, CO 80307

NAME BRIAN T. SMITH  
(PLEASE PRINT)

SIGNATURE Brian T. Smith

DATE: 12/23/86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.



Comments Accompanying The Letter Ballot on Document S8.103

Brian T. Smith

I am very much in favor of having the document S8.103 available for public review as soon as possible. The following comments represent editorial improvements to the document and its exposition. It is my opinion that the technical content of this document, particularly after the enclosed comments are addressed, is sufficiently clear and unambiguous to be reviewed by the public as a draft proposed standard. Indeed, it is most appropriate at this time to have this draft reviewed by the users of Fortran before any further technical changes are made.

In reviewing this version of S8, its accompanying glossary (S11.102) and addenda, I recommend that the following corrections be made to improve the presentation and readability of the document. I found the glossary a considerable help in establishing the terminology required to read the document and recommend that it be included at least as an appendix to the draft document. As we all know, the index is incomplete but I recommend that we work to improve it for the public review.

## RESPONSE TO BRIAN SMITH BALLOT

Your ballot included many editorial suggestions that will improve the readability and consistency of the document. The proposal on host association was approved by full committee for inclusion. All your comments were either accepted or considered and rejected.

# X3 Subgroup<sup>-106-</sup> Letter Ballot

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No.: 103 (\*)JCA-2

Date: Nov. 21, 1986

Project: 67

Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes: NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Affirmative
- Affirmative, with comment
- Negative, with reasons

Comments (Additional pages of comments may be attached.)

SEE ATTACHED COMMENTS

Mail to: Jeanne Adams, Chair, X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307

NAME RICHARD C. SWIFT  
(PLEASE PRINT)

SIGNATURE Richard C. Swift

DATE: January 5, 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

33.0  
115

**RESPONSE TO RICHARD SWIFT BALLOT**

Your editorial comments were considered by X3J3 and either accepted or rejected.

# X3 Subgroup -108- Letter Ballot

Accredited Standards Committee  
Information Processing Systems\*

102 (\*) JCA-7  
Doc. No.: 103 (\*) JCA-2 p. 3 of 3  
Date: Nov. 21, 1986 10/1/86  
Project: 67  
Ballot Period: Dec. 1 - Jan. 5, 1987

Subject: Fortran Revision of X3.9-1978

Ballot Closes NOON Jan. 5, 1987

Ref. Document: X3J3/S8/103

## FOR ACTION

### Statement:

X3J3 has voted to conduct a letter ballot of the membership to approve the draft Fortran Revision (Fortran 8x) of X3.9-1978. This X3J3 letter ballot answers the following question.

### Question:

Do you approve the draft Fortran Revision (Fortran 8x) of X3.9-1978 (Fortran 77) enclosed with this ballot for submission to X3 for further processing as an American National Standard?

- Yes
- Yes, with (editorial) comments supplied below
- No, with (Technical and editorial) comments supplied below

Comments (Additional pages of comments may be attached.)

Comments Attached

Mail to: Jeanne Adams, Chair, X3J3 NAME R. Weaver  
 NCAR - SCD (PLEASE PRINT)  
 PO Box 3000  
 Boulder, CO 80307 SIGNATURE [Signature]

DATE 12/30/86

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8535  
Fax: 202/635-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

36.0 IBM SJ 176  
117

IBM Corporation Santa Teresa Laboratory San Jose, Ca

Response to X3J3 Letter Ballot on Forwarding 8x to X3 for Further Processing. January 5, 1987

X3J3, attempting to extend Fortran with many new features while maintaining compatibility with Fortran-77, has produced a draft document describing a proposed 8x language. Language design is a major effort for any organization and is especially difficult for a standards organization. X3J3 is to be admired for its continued labor over the many years required for the difficult task of defining a large and complex programming language.

X3J3 now proposes that this language, 8x, be submitted to the X3 parent committee for further processing, with the goal of 8x eventually becoming a new standard, replacing Fortran-77.

The IBM response to this proposal is "no".

The difficulty with 8x lies not in the quality of the work that X3J3 has done (indeed we will here not make any comment on the quality of 8x as a language) but rather in the somewhat surprising observation that the result of X3J3's work is a new language and not a revision of Fortran.

Why isn't 8x a new Fortran? What is Fortran? Or better, "what are the attributes of Fortran that should remain constant across different versions of a Fortran standard"?

Each new 8x feature is, in some sense, "a good idea"; at some time each feature had sufficient support in X3J3 to be added to the draft. Notice, however, that the questions above, and the criteria we are about to develop, pertain to Fortran in total, rather than to individual features. The overriding concern, which we and every reviewer must address, is the total mass of the 8x language.

In reviewing the Fortran literature and in the experience of the many IBM'ers who produce, market, sell and use Fortran compilers, we have found that Fortran:

- \* is a small language, used by full-time programmers as well as being suitable for occasional use by professionals who are not necessarily computer programmers. Fortran is described as a language where, the occasional user can write programs without the extensive use of a reference manual.
- \* is efficient. Fortran statements have a straight-forward relationship to the machine. Fortran users understand what Fortran does and there is a simple run-time environment.

- \* permits inexpensive implementations (relative to many other languages).
- \* has an enormous inventory of application code
- \* is one of the most widely implemented programming languages (and thus one of the most, if not the most, portable programming languages)

This widespread implementation is, we believe, a result of both the application code inventory and the ease of implementing efficient compilers.

Further, prior Fortran standards have carefully preserved user investments in programmer training, as well as vendor investments in Fortran implementations.

D. Barron summed it up when he wrote that Fortran has survived like some hardy weed. In one short phrase, Fortran's successes and deficiencies are both acknowledged!

But what about 8x? Where does it stand with regard to the advantages that Fortran has now?

- \* small size? NO.

8x is a very large programming language, consider the following (see note 1):

	Pascal	Ada	8x
statements	19	48	84
syntax rules *1	150	179	280
vocabulary *2	46	84	144
intrinsic functions	29	49	97

- \*1 syntax rules (BNF productions) specify the sequences of tokens that programmers use in constructing a program
- \*2 Keywords + operators

- \* efficiency? NO (at least not soon).

While the efficiency of individuals features can be debated (and has been), that is not the problem. Implementors must contend with the total language. As described by Neil Lincoln (in "High Speed Computer and Algorithm Organization" Kuck, Lawrie and Sameh eds.):

"There was, in 1964 no great body of theoretical work to guide compiler developers in dealing with the multiplicity of CDC 6600

functional units and its many asymmetrical registers. After nearly TEN YEARS, a truly optimum match between language features, compiler, object code reliability, and machine organization was finally achieved. To expect less of an effort for similar results with new super-computer schemes is fatuous."

Ada may require somewhat less than ten years; there is no reason to believe that 8x will be substantially different from the CDC 6600 or Ada experiences.

\* permits inexpensive implementations? NO.

8x impacts implementation costs in several ways:

- the size of the language; there is simply lots more work to do.
- dependent compilation. The USE statement requires information from other (prior) compilations. This requires new library services (an "environment"), both to store and control this information.
- other operating system services are impacted: Link editing must now handle scopes for external names, for example.

\* large inventory of application code? YES, but...

8x does accept the existing Fortran 77 inventory, but that inventory comes with COMMON, EQUIVALENCE, ENTRY, ... all of which 8x says are obsolescent and can be deleted from future versions of the standard when their use becomes insignificant.

\* widespread implementation? NO.

We noted that Fortran 77 provided both the motivation (large application base) and the ease of implementation that resulted led to its almost universal implementation. There will not be the same incentives for 8x implementations.

\* preserve user investments in user training? NO.

8x provides Fortran 77 compatibility to support "dusty decks", programmers are encouraged to use only the new language. Of 72 statements in this language, 31 are new and many statements now in common use are "discouraged". (See note 2).

\* preserve vendor investments in reliable compilers? NO.



8x would have a major impact on many existing Fortran compilers. All parts of a compiler would be affected by the significantly larger language: dictionary phases are heavily impacted by qualified names, name scoping and new data types. Code generation is further impacted by structures, array language and array descriptors.

Without regard to the advantages or shortcomings of the 8x design it is clear that it is a large new language, and that, while it is based on Fortran, it has different attributes and must be intended for a different, or more specialized, user community.

The Fortran user community spans all segments of society - industry, government, education, .... The continued availability of a reliable, efficient Fortran is essential to many users. We do not believe that Fortran should be replaced by an unimplemented, untested language.

-----  
For IBM to vote "Yes", the 8x proposal must be reorganized into two languages.

One called Fortran, with the same attributes that Fortran 77 has today, would be processed as a new version of the Fortran standard (the document would be a revision of the current Fortran 77 standard). This language:

- \* Protects the users' investment in existing programs
- \* adds data structures
- \* adds array language as an option. Array language needs to be optional to permit an economic choice of implementation on scalar machines (especially scalar micro's).
- \* adds a bit string data type (missing from the 8x proposal, this requirement was considered for Fortran 77, dropped, provided for in the ISA S61.1 standard, and continues to be discussed in the literature, most recently in the Oct 86 CACM "Accessing Bit Fields in Fortran-77" by Manny Hayes)
- \* adds ALLOCATE/FREE statements
- \* adds environmental inquiry functions
- \* adds precision specification
- \* adds keyword and optional arguments to CALL statement

\* adds IMPLICIT NONE

\* adds new form of DO

The second language (name to be specified), consisting of the remainder of the 8x language, would be processed as a Technical Report. This would provide for some implementation experience, testing, and evaluation of the new language before being considered for standardization.

R. Weaver

Note 1: The use of syntactic counts to characterize language size is not new here. See, for example, M.B. Wells in "Frontiers of Supercomputing" Metropolis, Sharp, Worlton and Ames eds.:

"The primary fault of Ada is that it is big and complex. There are 62 key words and over 150 syntactic constructs. ... Actually, in spite of this, it has a certain cleanliness that makes it surprisingly easy to learn, at least up to some reasonable level of competence."

8x, a mixture of existing Fortran and new language, will have no such "cleanliness".

Note 2. 8x statements are:

Obsolescent:	(12) ASSIGN	Assigned	GOTO	Comp	GOTO
	Arith-IF	PAUSE		DIMENSION	
	COMMON	EQUIVALENCE	DOUBLE		
	BLOCKDATA	ENTRY		stmt-function	

From Fortran 77:	(41) Arith-assign	logical-assign	char-assign
	type-assign	GOTO	Logical-IF
	CALL	RETURN	CONTINUE
	STOP	DO	ELSE
	ELSEIF	END	ENDIF
	IFTHEN	READ	WRITE
	PRINT	BACKSPACE	REWIND
	ENDFILE	FORMAT	CLOSE
	INQUIRE	OPEN	EXTERNAL
	INTEGER	REAL	COMPLEX

36.5

122

LOGICAL	CHARACTER	IMPLICIT
INTRINSIC	PARAMETER	SAVE
DATA	FORMAT	FUNCTION
SUBROUTINE	PROGRAM	

New:

(31) EXIT	CYCLE	ENDDO
SELECTCASE	CASE	ENDSELECT
NAMelist	END SUB	EXPONENT-LETTER
END PROG	END FUN	END BLOCKDATA
CONTAINS	ALLOCATE	DEALLOCATE
IDENTIFY	WHERE	ELSEWHERE
ENDWHERE	type-dcl	derived-type
ENDTYPE	USE	MODULE
ENDMODULE	INTERFACE	ENDINTERFACE
OPTIONAL	INTENT	RANGE
SETRANGE		

36.6

123

## RESPONSE TO IBM BALLOT

The IBM ballot is divided into two sections. The first deals with problems with Fortran 8x taken as a whole. The second is a proposal to reorganize the language remedying the problems outlined in the first section. This response gives the committee's view of these concerns and why it believes the reorganization is unnecessary.

The IBM ballot first addresses the properties that Fortran has possessed that have made it successful. Among these are small size, efficiency, low implementation cost, widespread implementations, and a respect for user and vendor investments in code and training. It proceeds to build a case that Fortran 8x is deficient in these areas. X3J3 has the following view of these concerns:

### Language Size

While X3J3 does not believe that the "size" of a language can be adequately expressed by a count of statement types, syntax rules, vocabulary, and intrinsic functions, it is plain that Fortran 8x is considerably larger than FORTRAN 77. X3J3 believes that this increase in language size is justified on several grounds:

1. Each feature, when considered individually, has been judged by a majority of X3J3 during its deliberations to be worth the cost of implementing, training, and providing support for the feature by vendors and users.
2. It is desirable to limit the frequency of revisions to a widely used language like Fortran to allow a wide range of experience to accumulate before making changes. For this reason, it is important to introduce substantial new capabilities in each revision.
3. The ability of users and vendors to deal with complexity has increased since FORTRAN 77 was issued.
4. Evolution of hardware has created a need for language support.

### Efficiency

X3J3 has deliberated at length about the efficiency of each of the new features in this document, and the performance impact on FORTRAN 77 features. In each case, a workable implementation model has been discussed that would allow a reasonable implementation. X3J3 believes that the efficiency of existing FORTRAN 77 programs will be minimally affected. The efficiency of code for new Fortran 8x features will undoubtedly improve as technology progresses and implementors learn better ways to implement the new features.

### **Implementation Cost**

While X3J3 admits that Fortran 8x will be more costly to implement than FORTRAN 77, there are several justifications for these added costs. First, implementors are already spending considerable sums in extended FORTRAN 77 implementations. X3J3 feels that Fortran 8x provides a set of guidelines to channel this investment into areas that benefit users most directly. Second, implementation costs have declined since FORTRAN 77 was published due to the availability of faster processors, new program development tools and improved programming techniques. Finally, such implementation costs need not be borne all at once. They can be spread out over a period of time by releasing progressively more complete, robust, and efficient implementations.

### **Inventory of Application Code**

As all of FORTRAN 77 is contained in Fortran 8x, all investment in FORTRAN 77 training is therefore preserved in Fortran 8x. None of the features cited (COMMON, EQUIVALENCE, ENTRY) are obsolescent in Fortran 8x, and therefore could not possibly be considered for at least two more revisions of the standard beyond Fortran 8x--after the year 2010 at the earliest. Fortran should therefore strengthen its position as the language of choice for scientific applications, while preserving all investment in FORTRAN 77 code, thereby providing considerable motivation for widespread implementation.

### **Implementation Availability**

The same reasons for producing a Fortran 8x compiler exist as for FORTRAN 77. They include:

1. a large body of existing, upward-compatible programs,
2. a national, an international, and presumably a Federal standard resulting in requirements for Fortran processors, and
3. user demand for language features to exploit current hardware.

### **Preserving Training Investments**

X3J3 believes that training investments are preserved by offering total upward compatibility with the FORTRAN 77 language. Of course training costs will increase if any new features are added in Fortran 8x. However, the majority of the new features can be learned as needed.

### **Preserving Vendor Investments**

Unless extensions to FORTRAN 77 are minimal, they are likely to impact vendor investments in current compilers. Note that the majority of the items you cite as impacting existing compilers are also in your proposed Fortran language.

## Summary

X3J3 has previously considered the idea of splitting the standard into two languages and rejected it. X3J3 feels that creating two languages is not a good idea for the following reasons:

1. Two versions of Fortran would impede program portability, the very thing that standardization is supposed to enhance. In addition, your proposal to make the array processing extensions optional would further limit the portability of programs using those extensions.
2. It is not obvious that features in your proposed Fortran can be successfully separated from the MODULE/USE feature which is in your second Fortran.
3. The content of your proposed Fortran represents a substantial part of Fortran 8x and hence does not seem to have a significant impact on the concerns you raise relative to size, efficiency, and implementability.

# WG5 BALLOTS

TO: X3J3 and ISO/TC97/SC22/WG5 - Fortran  
FROM: Jeanne Adams, Chair, X3J3  
VIA: Jeanne Martin, Convener, WG5  
SUBJECT: X3J3 Responses to WG5 Ballot Comments

An informal ballot on the adoption of X3J3/S8.103 (February 1987) as the draft proposed new Fortran standard was sent to WG5 members. The ballot period extended through January 12, 1987. Twenty-seven ballots were received. The vote was:

Yes ..... 25      No ..... 2

Many of the comments returned with the ballots were suggestions for editorial changes. These were processed along with similar suggestions contained in the X3J3 ballot responses and either accepted or rejected. X3J3 prepared responses to the noneditorial comments. This document is a collection of those responses. The next page is a ballot tally. Affirmative votes without comments are not supplied, but are available on request.

Responding to country votes of WG5-Fortran conclude the ballot responses in Part II.



FORTRAN BALLOT SUMMARY FOR WORKING GROUP 5/SC22/TC97/ISO

Vote

Individuals

Adams, Jeanne	Yes
Buckley, Albert	Yes
Bourstin, C.	Yes
Delves, L. M.	Yes
Du Croz, J. J.	Yes
Hammarling, S. J.	Yes
Kan, Tadayoshi	Yes
Kneis, Wilfried	No
Luttermann, Hermann	Yes
Marshall, Neldon H.	Yes
Martin, Jeanne T.	Yes
Meek, Brian L.	No
Meier, Bruno	Yes
Milinazzo, F.	Yes
Munchhausen, Meinolf	Yes
Muxworthy, David T.	Yes
Plassner, Klaus	Yes
Pollicini, Aurelio	Yes
Rotthausen, K.	Yes
Schmitt, Gerhard	Yes
Schonauer, Willi	Yes
Shen, M. K.	Yes
Vallance, David M.	Yes
Wagener, Jerrold L.	Yes
Warren, Graham	Yes
Wilson, John D.	Yes
Wu, Qing-bao	Yes

Countries

Austria	Yes
Canada	Yes
Germany	Yes
United Kingdom	Yes

23753  
L300

SECRET  
JAN 21 3 03 AM '87

LLNL LVMR

LLNL LVMR

AFNOR Z 611974F

ZCZC

JFA TX 214 87.01.21

ATT : MRS JEANNE MARTIN

I APOLOGISE FOR THE DELAY OF AFNOR ANSWER RELATING TO DP FORTRAN-EX FRENCH VOTE IS AS FOLLOWING

YES WITH COMMENTS

1 - TO CHECK THAT THE IMPLEMENTATIONS COULD AVOID CONFLICTS BETWEEN NEW AND DEPRECATED FEATURES

2 - THE FORTRAN AFNOR GROUP IS SURPRISED THAT X3J3 HAS NOT TAKEN INTO ACCOUNT SOME OF THE RESOLUTIONS (NOTABLY POINTERS-SIGNIFICANT BLANKS) VOTED WITH A VERY LARGE MAJORITY AT THE HALIFAX MEETING.

RELATING TO PP WG 5 PARIS MEETING, I WILL COMMUNICATE YOU THE FINAL DATE AFTER DISCUSSION WITH THE AFNOR FORTRAN GROUP.

SINCERELY YOURS AND BETTER LATER THAN NEVER - BEST WISHES FOR 1987

BOURSTIN

AFNOR

LLNL LVMR

AFNOR Z 611974F

TO REPLY FROM TELEX I OR II (TWX) DIAL 100 FROM EASYLINK USE /WUW.  
EST 1127 JAN/21/1987

## RESPONSE TO BOURSTIN BALLOT

In your comments you indicate you are concerned about possible conflicts between the new and deprecated features. X3J3 has made every attempt to avoid any such conflicts and to permit the use of old and new features in the same program with the exception that a program unit must use a single source form throughout.

You also mention the Halifax Resolutions that recommend the addition of pointers and the reinstatement of significant blanks in the new source form.

Prior to the first S8 ballot, that failed in X3J3, the functionality of pointers for arrays was provided by the IDENTIFY statement. In an attempt to simplify and reduce the size of the language, the IDENTIFY statement was restricted to identification on a single host, thus removing the pointer capability. A similar functionality for scalar entities has never been described by X3J3. There is considerable pressure to forward S8 for review as soon as possible. Adding new functionality at this time would delay that process and would be contrary to the direction that was mandated by the first unsuccessful ballot.

Blanks were, at one time, recognized as significant characters in free source form only. The language must work equally well expressed in either source form, so it was not possible to make use of blanks to simplify the syntax. Many believe that existing programs should be processable forever without change, and thus it would never be possible to make use of blanks as meaningful characters. There is little point in placing new restrictions on the form of the language.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country S J HAMMARLING  
or  
Name NAG Limited, Oxford, UK.  
please print

Individual  
Signature S. J. Hammarling  
Date 18<sup>th</sup> December 1986

PS

132

Ms Jeanne T Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA



The Numerical Algorithms Group Ltd  
NAG Central Office  
Mayfield House  
256 Banbury Road  
Oxford OX2 7DE  
United Kingdom

tel (0865) 511245  
international + 44 865 511245  
telex 83354 NAG UK G

17 December 1986

Dear Ms Martin

WG5 Informal Letter Ballot

Please find enclosed the postal vote of my colleague Sven Hammarling to the WG5 informal letter ballot, together with a number of drafting amendments.

The NAG technical planning committee met in Moreton-in-Marsh, Gloucestershire on the 10th - 11th December 1986. Its membership includes well-known and knowledgeable experts in the area of numerical software.

Your colleagues have already acknowledged our concern at the delay in publishing the draft Fortran 8X standard. The planning committee urged a "yes with comments" response to the postal ballot. It continues to believe that the current draft proposes a language that will greatly benefit the Fortran community. We look forward to the opportunity to make a careful review of the final draft during the public review process.

With best wishes,

Yours sincerely

A handwritten signature in cursive script that reads 'Brian Ford'.

Dr Brian Ford

- cc: J.C. Adams, Chairperson, ANSI X3J3 Committee
- B. Meek, Chairman of BSI IST 5, Programming Languages Committee
- R.W. Meijer, CEC, Information Technology Task Force
- P. Messina, Chairman, Language Working Party, US Dept. Energy
- D.T. Muxworthy, CAST, Edinburgh
- J.K. Reid, AERE, Harwell
- J.L. Schonfelder, University of Liverpool
- B.T. Smith, Argonne National Laboratory
- J. Wilson, Chairman of BCS Fortran Special Interest Group

P 11

133

## RESPONSE TO HAMMERLING BALLOT

Your detailed editorial comments were very helpful to X3J3 in refining S8. They have been considered along with many others and either accepted or rejected.

There are alternatives for the left and right brackets, which are reserved for national character sets. Section 3.1.4 on special characters has been extended to indicate the alternate syntax.

You note that the model defined for integers does not include the full range of 2's complement values. That is true. It is merely a model and has the property that processors may have more numbers than the model permits, since it does not include the most negative integer (or real) number on 2's complement machines.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

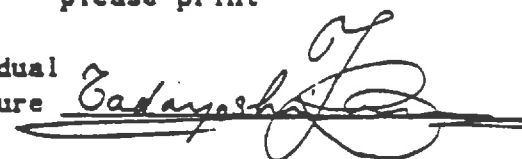
- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country Japan  
or  
Name Tadayoshi Kan  
please print

Individual  
Signature 

Date 1986-12-19

P12

135

Japanese Comments

We approve to process S8.103 for submission to SC22 provided that the following proposals should be considered.

- (1) Extension of the intrinsic character type

(see Attachment A) :

to insert [(KIND=n)] between the keyword CHARACTER and [length selector] in the syntax R502,

- (2) Additional intrinsic functions (see Attachment B) :

to add the important functions shown in the Attachment B to the intrinsic functions of S8.103.

We strongly propose either the above (1) and (2) are included in the standard or in Appendix F of S8.103.

Then, we would like to prepare the detail works caused by adopting our proposal, if necessary.



Attachment A

Extension of the Current Intrinsic Character Type

We expect that 8x will be widely used for new fields other than traditional numerical computation, because of having contemporary features like "derived data type." Especially, we strongly hope that such application systems as data bases and mechanical translators could be implemented by 8x. In order to do so, it is necessary that 8x has the capability to process various kinds of character codes, e.g. Kanji code, which are represented by multibytes. We think this capability can be realized by only extending the current character type to more flexible character types shown below.

Proposal:

(1) Type and Storage Units

Character Type and Storage Units	
Type	Number of Character Storage Units
Character Type 1 (1st kind)	1
Character Type $n$ ( $n$ -th kind)	Any

The 1st kind character type is the current type, and the  $n$ -th ( $n > 1$ ) kind is the proposed type which is provided by pre-compiling information as parameter setting.

(2) Type Declaration Statement

CHARACTER [(KIND =  $n$ )] [length selector]

If (KIND= $n$ ) is omitted or (KIND=1) is written, the statement specifies the current character type.

P 14

If  $(\text{KIND}=n)$  is written and  $n \neq 1$ , the statement specifies the processor setting character type corresponding to  $n$ , other than the current character type.

**Reason:**

- (1) We believe that new users who appreciate the introduction of derived data type will expect to process various character strings by 8x. From this viewpoint, it is a significant defect that S8 has only the current intrinsic character type.
- (2) It seems that the current intrinsic character type is intended to manipulate only one byte code, e.g. ASCII. However, ISO/TC 97/SC 2 is now discussing various multibyte codes which are requested by many member bodies of ISO/TC 97. Considering the increasing needs in the next decade, 8x shall have the capability to various kinds of character codes.
- (3) From the standpoint to implement such application systems as data bases and mechanical translators, it is necessary to deal with only character codes but also various kinds of codes in wide meaning. We can expect that programs and data are simplified and compactized by using such codes, e.g. a code of molecular structures or a code of dot printing patterns. From this point, we emphasize that the specification of the kinds of codes should be provided by pre-compiling information as parameter setting.
- (4) By only little modification of the current CHARACTER statement like our proposal, it is expected to produce extremely powerful and wide applicability of 8x.

Attachment B

1. Mathematical Functions

CBRT	cubic root
ERF	error function
ERFC	complement error function
GAMMA	gamma function
LGAMMA	logarithmic gamma function
CANG	amplitude of a complex number
LOG2	logarithmic function to the base 2
EXP2	exponential function to the base 2
ASINH	hyperbolic arcsin
ACOSH	hyperbolic arccos
ATANH	hyperbolic arctan

2. Constant functions

C_E	e
C_P1	$\pi$
C_EULER	Euler's constant

3. Table functions

T_THETA	theta function
T_B	Bernoulli number
T_EULER	Euler number

P 16

## RESPONSE TO TADAYOSHI KAN BALLOT

Your ballot contained requests for two extensions to the draft proposal. The first was your proposal to extend the CHARACTER data type. It was not adopted for the reasons given below. The second request, for additional intrinsic functions, was rejected as it would increase the size of the language. The first letter ballot failed in X3J3 because many X3J3 members felt the language was too large. Prior to the second ballot, a number of features were removed from the language. They are described in Appendix F for reference purposes only. It was further agreed that new features would not be added before the document is forwarded as a draft proposed standard.

X3J3 is sympathetic with the need to support very large character sets in Fortran; however, X3J3 has not yet explored how best to support this functionality. X3J3 would like to investigate this problem during the public review of S8, with your help, and address your ballot request more fully.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied


Comments: (Additional pages of comments may be attached.)

MISSING : EXCEPTION HANDLING, BIT DATA TYPE

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country  
or  
Name WILFRIED KREIS  
Please print

Individual  
Signature 

Date 12/15/86

P17

PLEASE NOTE NEW ADDRESS:  
KREUZWIESE 7  
D 6392 NEU-ANSBACH  
WEST GERMANY

141

## RESPONSE TO WILFRIED KNEIS BALLOT

X3J3 notes that the reasons for your negative ballot are that exception handling and bit data type are missing. X3J3 considered your requests but did not accept them. Both of these features were in the language at the time of the first letter ballot; however, that ballot failed in X3J3 because many X3J3 members felt the language was too large. Prior to the second ballot, several features including these two were removed from the language. They are described in Appendix F for reference purposes only. With these features removed, the second ballot was successful. More specific reasons for not reinstating these features are as follows.

### Exception Handling

The CONDITION/ENABLE facility of Fortran 8x has been one of the most controversial features of the draft standard. Among the reasons for this controversy are:

- a. concern about the possible effects on optimization,
- b. questions concerning propagation of conditions up the call chain in search of a handler, and
- c. the infeasibility of sensible resumption of execution in the case of some conditions.

The ENABLE block, which syntactically resembles a control construct, is intended to limit the impact on optimization to programmer-specified portions of the code; programmers must explicitly so identify each portion of code in which exception handling is activated. There seems to be a considerable amount of "discomfort" in X3J3 about the provisions in this exception handling facility for propagating conditions back up the call tree in search of a handler; the implications for procedure interfaces may not be entirely clear. Finally, there is considerable controversy over the "granularity" of resuming execution after handling a condition; the current facility has statement level granularity, with execution resuming at the next statement.

### BIT Data Type

Bit level processing has been discussed by X3J3 for many years. The current bit array facility in Appendix F was adopted, rather than a bit string facility. On the basis of maximum compatibility with the array facilities, and, conversely, operations on bit arrays, provide much bit string type capability. Nevertheless, there remains significant preference for a bit string, rather than a bit array, facility. Two additional factors appear to prevent complete consensus on any comprehensive bit processing facility in Fortran. First is that data representation at the bit level is inherently variable among implementations and hence corresponding software dependence is extremely nonportable. Second is that none of the bit proposals presented thus far have been able to satisfactorily define bit stream I/O, which seems to be required in order to support true bit data processing.

-134-

# WG5 INFORMAL LETTER BALLOT

for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

A Merry Christmas and a (better) standard New Year! see over!! →

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

or  
Name

B. L. MEEK (UK)

please print

Individual  
Signature

B. L. Meek

Date

19 December 1986

P31

143

I vote NO for two reasons, either of which would have been sufficient in itself. If both are met I shall be happy for it to be issued as a draft for public comment though there are many deficiencies (e.g. its inconsistency of presentation and its looseness of wording) which would subsequently need to be corrected before it could become a DIS.

Reasons for NO vote:

1. Failure to include the enhanced "Conformance of processors" section (including detection and reporting of extensions, and documentation) agreed at the Halifax WGS meeting. If there are any difficulties in phrasing, let these be allowed to come out at public comment stage, along with everything else. What deficiencies there are seem to me to stem from the vagueness of much of the rest of this section (e.g. "forms and relationships"), not from the Halifax wording.
2. Failure to reinstate Appendix F into the main body of the standard. I am willing to accept two classes of entry: "to be included unless strong reasoned argument is received against" and "to be excluded unless strong reasoned argument is received in favour." Not this.

P32

Brian Meek



## RESPONSE TO BRIAN MEEK BALLOT

You cite two reasons for your negative vote: the lack of processor conformance requirements and failure to reinstate Appendix F. Processor conformance requirements were added at the February 1987 X3J3 meeting. However, a substantial part of Appendix F has not been reinstated. It is the intent of X3J3 that Appendix F constitute precisely those features that are "to be excluded unless strong reasoned argument is received in favor".

The WG5 Halifax Resolutions specified reinstatement of several specific Appendix F features. Most of these (e.g., structure constructors, overloaded operators) have been reinstated. The WG5 Halifax Resolutions did not request reinstatement of a large majority of the Appendix F features. The current status of Fortran 8x is in very substantial conformance with the Halifax Resolutions.

-137-

# WG5 INFORMAL LETTER BALLOT

for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

- include the syntax and semantics of pointers at least in appendix F
- include a new appendix with syntax charts (like appendix F of ANSI X3.9-1978)

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country BRUNO Meier  
or  
Name SWITZERLAND  
please print

Individual  
Signature

Bmeier

Date

3/1/87

P33

146

## RESPONSE TO BRUNO MEIER BALLOT

Your ballot comments contain two suggestions for additions to S8: pointers and syntax charts. X3J3 considered your suggestions and decided not to adopt them.

You suggest the addition of pointers, at least in Appendix F. The first ballot failed in X3J3 primarily because many X3J3 members felt the proposed language was too large and complex. Prior to the second ballot, several features were removed from the language. One of the considerations used in selecting features to be removed was the necessity for promoting the efficient execution of Fortran programs. Thus features that might interfere with optimization were prime targets for removal. One of these was exception handling; another was the identification of arrays on multiple hosts. The latter feature provided the functionality of pointers for arrays. Since it was considered as merely a simplification of the IDENTIFY statement, it was not described as a feature in Appendix F. A similar functionality for scalar entities has never been described by X3J3, and would seem unlikely to gain favor at this time.

You suggest the addition of an Appendix containing syntax charts similar to Appendix F of ANSI X3.9-1978. When X3J3 adopted the form for the syntax rules used throughout the document to describe the language, it was felt that syntax charts would not be needed. The syntax rules define the language more precisely than the word descriptions used in ANSI X3.9-1978 and can be included in the body of the proposed standard, whereas the syntax charts were relegated to an Appendix. However, the syntax rules are extracted from the text of S8 to form a concise description of the language in Appendix D.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)  
Seven pages of comments are attached.

---

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country  
or

Name David Muxworthy, U.K.  
please print

Individual  
Signature

D. J. Muxworthy

Date December 19, 1986

P35

148

General Editorial Comments.

The document is found difficult to read by those new to it. It would benefit from editing:

- to have a short introductory paragraph at the start of each section
- to cut out unnecessary sentences and much tutorial matter which has no place in a standard
- to eliminate text which is duplicated in different sections
- to have a short introduction before a long set of syntax rules
- to give section numbers in the case of important forward references
- to reduce verbosity, thus exposing the meaning of the text.

I support the use of "Fortran" as a proper noun, not as an acronym.

Detailed Editorial Comments.

These are arranged in page order, not order of importance.

- 1A. p 1-1 l 7. This conflicts with p 1 l 6-7. Replace "is standard conforming" by "is intended to be standard conforming".
- 1B. In accordance with Halifax WGS resolution 10, and subsequent discussion in X3J3, make the following changes:

p 1-1 l 34. Add "generally" before "refer". This was in the corresponding text in ANS X3.9-1978 (p 1-2 l 9).

p 1-1 l 37. Add "usually" after "must". This was in the corresponding text in ANS X3.9-1978 (p 1-2 l 14).

p 1-2 l 5-6. Delete the sentence contained in these lines and replace by:

"A processor conforms to this standard if:

(a) it executes standard-conforming programs in a manner that fulfills the interpretations prescribed herein;

(b) it contains a capability to detect and report the use within a submitted program of permitted additional forms of Fortran statements and constructs (see below), or of forms designated herein as obsolete, obsolescent or deprecated;

(c) it is accompanied by documentation which defines the additional forms

RESPONSE TO DAVID MUXWORTHY BALLOT

Your detailed editorial comments were very helpful to X3J3 in refining S8. They have been considered along with many others and accepted or rejected.

You made some good suggestions for improving the readability of the document. However, a substantial rewrite would delay publication for public comment and that would be contrary to the first Halifax Resolution which WG5 regarded as of prime importance.

Message 2:

From Z3000TT@AWITUW01.BITNET Wed Jan 7 08:38:04 1987

Received: Wed, 7 Jan 87 08:37:48 pst from WISCVW.WISC.EDU by lll-crg.ARPA (4.12/  
)

id AA19723; Wed, 7 Jan 87 08:37:48 pst

Received: from (Z3000TT)AWITUW01.BITNET by WISCVW.WISC.EDU on 01/07/87  
at 10:37:17 CST

Message-Id: <870107144943.00002639.ABVU.TU@AWITUW01>

Date: Wed, 7 Jan 87 14:49:43 MEZ

From: Z3000TT@AWITUW01.BITNET (G SCHMITT TU Wien)

Subject: Vote on S8.108

To: martin@lll-crg.arpa

Status: R

Dear Jeanne,

too send you a quick vote on S8.103 I will vote before finishing  
the detailed reading of S8. Even if I am unhappy with some parts  
of S8, as I have already stated, I vote

Yes, with comments supplied

You will get more comments if I find time to type them in. This is  
my personal vote. There is a meeting of our "Language"-standards  
committee an friday. I expect the vote of AUSTRIA will also be "YES".  
You get an message immediately after the meeting.

My yes-vote is only because too speed up the process of public comment.  
Otherwise I would have voted "NO" especially because I like to see more  
deprecated features (storage assotiation, ...) and some features of  
Appendix F (Exeption Handling, ...) back in the text.

Concerning the informal X3J3 reports I get this time only the report  
of John Reid. I miss Len Moss's and Mike Metcalf's reports.  
At the moment I know smething about 15 W65 e-mail addresses. This is  
more than 25% of the addresses.

Sincerely, Gerhard

&

## RESPONSE TO GERHARD SCHMITT BALLOT

Your detailed editorial comments were very helpful to X3J3 in refining S8. They have been considered along with many others and accepted or rejected. Your other suggestions are contrary to the compromise of June 1986 that was essential for consensus and therefore were not accepted.

You say you would like to see more deprecated features, including those that rely on storage association. As you know, the first S8 ballot failed in X3J3. One of the key changes prior to the second ballot was to introduce obsolescent features that appear in S8 in a small type font. These are features that can be omitted from current Fortran programs. The deprecated features still exist, but they are not differentiated in S8 by a different type font. They are listed in Appendix B which also describes which new features can be used instead of the deprecated ones. In a subsequent revision of the language, some of the deprecated features could become obsolescent. Thus a path still exists for the migration of features out of the language, but it will depend on whether the features actually fall into widespread disuse.

Another key change prior to the second ballot was the removal of several features from the language in order to simplify and reduce the size of the language. One of the removed features was exception handling. With these features removed, the second ballot was successful.



-144-

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov..18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

*I deeply regret the removal of vector-valued subscripts for arrays. This brings back again many unnecessary do-loops and may prevent efficient vectorization of indirect addressing which is so important for engineering applications (finite element method). But I vote for a quick procedure to obtain the new standard SC22*

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country FRG  
or  
Name Willi Schönauer  
please print

Individual  
Signature *Willi Schönauer*

Date 12/8/86

153

## RESPONSE TO SCHONAUER BALLOT

Your ballot comments indicate you regret the removal of vector-valued subscripts from the proposed language. The first letter ballot failed in X3J3 because several members felt the language was too large and complex. Prior to the second ballot, the descriptions of some features (including vector-valued subscripts) were moved to an Appendix that is not part of the proposed standard. It was felt that vector-valued subscripts contributed to the complexity and irregularity of the language. Note that in the Appendix F description, there is a restriction that the left-hand side of an assignment statement must not include an array element more than once. This restriction was required to promote portability of Fortran programs since it was not in the best interests of optimization to standardize the order of array assignment. Without this restriction, efficient execution would interfere with portability, and portability is the main purpose of standardization. The rule introduces an irregularity since there is no similar restriction for the right-hand side of an assignment statement. With such features removed, the second ballot succeeded.

# WG5 INFORMAL LETTER BALLOT

**for individual WG5 members**

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

or  
Name M. K. Shen  
please print

Individual  
Signature M. K. Shen

Date 30.12.86

P59

155

Comments accompanying the informal letter ballot

For an average reader of the document in the forthcoming public review period the first sentence of Appendix F is unlikely to be convincing/sufficient. In order to reduce the language size effectively, one would argue, the whole of array operations and the corresponding intrinsic procedures should have belonged to Appendix F.

Since it appears questionable whether the majority of those WG5 members who have not attended X3J3 meetings possess sufficient information on the genuine arguments supporting the exclusion of the features presently in Appendix F from the standard proper, I suggest that WG5 request X3J3 to supply, for reference purposes, an abstract of concrete and convincing specific/non-specific arguments that have been successfully put forward in this connexion.

During the public review period WG5 members are quite likely to be questioned by interested persons of the international community concerning that matter. It would be very embarrassing to have to admit one's ignorance.

Munich, 30th December 1986



M. K. Shen

## RESPONSE TO J K SHEN BALLOT

You have requested that X3J3 supply concrete and convincing arguments to support the selection of features that were removed from the proposed revision of Fortran following the first letter ballot. That ballot, while successful in WG5, failed resoundingly in X3J3. The vote was 16-yes, 20-no. A two-thirds majority is needed to release the draft to X3 for public review. It is to the advantage of all concerned for the international and American National Fortran standard to be identical. The normal ANSI policy for dealing with a negative ballot is to examine the comments returned with the ballots to see how the proposed draft standard could be altered to gain the greatest number of yes votes.

The predominant ballot comment was that Fortran 8X was too large and complex and that some of the new features should be removed. Most commenters included a list of features recommended for removal. Rather than formally vote feature by feature, an attempt was made to incorporate those features most requested for removal into an overall "removal package", one that left the language not only smaller but also coherent as a whole. In addition, other features that were not removed were simplified. The removed features were placed in Appendix F so that they would be available for reference during the public review. The intention is that Appendix F will not appear in the final standard.

It was unfortunate that you could not attend the WG5 meeting in Halifax. Many of those who did attend felt, like you, that the removal of these features was unnecessary and undesirable, but they accepted it because they also felt the excisions would expedite the production of a new standard. They recommended the reinstatement of several features and most, but not all, of those features were reinstated. They felt it was important that Appendix F be included in the review document, and they urged that the document be made available for public review as soon as possible.

In general, the features removed were those that were seen to be most in conflict with the purposes of the standard: portability and efficient execution. It is difficult to accurately summarize the points in the comments calling for the removal of specific features, but the points include the following concerns:

- **Remove CONDITION/ENABLE.** There was concern about the possible negative effects on optimization. There were questions concerning the propagation of conditions up the call chain in search of a handler. Sensible resumption of execution in the case of some conditions was not feasible.
- **Remove BIT Data Type.** Bit strings are preferable to bit arrays. Bit processing is inherently nonportable. There was no bit stream I/O.
- **Revise Internal Procedures.** Internal procedures were simplified by prohibiting ENTRY statements in them and disallowing the nesting of internal procedures. Disallowing the passing of internal procedure names as arguments avoids implementation difficulties when the host is recursive.

- **Simplify MODULE/USE.** Simplifications include restricting USE statements to the using of modules (not also to control access to host entities), removing the ALL EXCEPT use option, and making module procedures like external procedures rather than like internal procedures.
- **Revise the Array Features.** Vector-values subscripts were removed because of the many-to-one problem. The FORALL statement was removed because the additional functionality was not considered to be sufficiently important. Several array intrinsic functions were removed because they were overly specialized. The IDENTIFY statement was limited to identifying on a single host to improve the efficiency of compiled code.
- **Remove Variant Structures.** The importance of the "equivalence functionality" of variant structures was reduced considerably when the EQUIVALENCE statement was removed from obsolescent status. There is continuing disagreement over how best to handle the "tagging" of variant components. With the advent of much larger memories, even on microcomputers, the space savings of variant components is of much less importance.
- **Make Blanks Insignificant in both Source Forms.** Since the language must work equally well expressed in either source form, it was not possible to make use of blanks in the free source form to simplify the syntax. Many believe that existing programs should be processable without change, and thus it would never be possible to make use of blanks as meaningful characters. There is little point in placing new restrictions on the format of the language.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

or Name D M VALLANCE  
please print

Individual Signature *D M Vallance*

Date 12/12/86

PG1

159

DMV/SMW/JMARTIN

18th December, 1986

Ms J T Martin,  
L-300,  
Lawrence Livermore National Laboratory,  
Livermore,  
CA 94550,  
USA

Dear Jeanne,

I enclose my completed informal Letter Ballot for WG5 members for your attention.

We have a strong interest in the progress of 8X here at Salford. Our discussions have led us to ask how much actual 8X code has been written, "compiled" and executed (interpreted?). (I know about Jerry Wagner's pre-processor). I have suggested to one of my colleagues that we might produce the syntax rules as a PROLOG program that would allow inconsistency checking amongst rules and would, in theory, allow programs to be written and "compiled". Perhaps we would even make it accessible over the networks to people. Do you think we might get some (relatively small) funding for such a project which could be completed in a fairly short timescale (say a few months)?

I have just (offer received today) been allocated some funding from UK Alvey Committee to work with NAG Ltd on Fortran 8X tools which, will, amongst other tools, provide a portable Fortran 8X parser. This is, however, a long term project and will not complete until c.1990.

Please let me have your thoughts on paragraph 2 if you have time.

Regards,



D M VALLANCE

p62

160



## RESPONSE TO VALLANCE BALLOT

You make suggestions for improving the presentation of the standard. Unfortunately, a substantial rewrite would delay the publication for public review and that would be contrary to the first Halifax Resolution, which WG5 regarded of prime importance.

X3J3 would like to present the details of the language as well as possible within the framework of a standard, but a standard is a legal and technical specification and not a descriptive or tutorial document. The reader, in a sense, needs to know everything at once. This leads to a large number of forward references.

The syntax rule meta language was chosen by a formal vote in X3J3. There is no ISO standard for meta languages; there was a draft proposal, but it was not adopted. In any case, X3J3 preferred a much simpler and easier to understand meta language.

At the most recent X3J3 meeting, conformance rules similar to the ones proposed in Halifax were adopted, and the glossary was incorporated as an additional appendix. Terms are defined precisely and one must be aware of the precise definitions to read the document correctly. This is particularly true for the terms you cite: "subprogram", "program unit", and "procedure".

The committee spent many meetings debating the term "module". Initially it was called a "bundle" with the assumption that at some future time the term would be changed. It is different from an Ada package, and "package" would not be appropriate. Since we had introduced the concept of "core + modules" and subsequently rejected it, we finally agreed to use the term "module". If there is confusion initially, the term may need to be qualified as in "Fortran module".

Internal procedures have been simplified considerably since the first ballot. At the most recent X3J3 meeting the default typing rules were modified to reflect most users' expectations. There have been clarifications of the use of non-Fortran characters in various contexts.

The comment on significant blanks was added to Appendix F to make it clear to reviewers of the document that they are no longer part of the new source form. A questionnaire was published in the Fortran Newsletter in 1979 and the responses so heavily favored making blanks significant that that was incorporated into the free form source. However, after the first ballot, the new source form was changed to be more compatible with the existing form. You are right that this item is independent of the rest of Appendix F, but X3J3 felt that an issue that had received this much publicity during the development period should be mentioned somewhere.

It is the intention that Appendix F not be part of the final standard, but there are many in X3J3 and WG5 who are not willing for the proposed draft to go out for public review without including in some form a description of the features that were removed as part of the compromise developed following the first ballot.

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

or  
Name JOHN D. WILSON  
please print

Individual  
Signature J. D. Wilson

Date 16 - Dec 1986

P72

102

Comments on X3J3/S8.103

My "yes" vote indicates I am in favour of submitting the Fortran 8X document for public comment. I do not think the draft standard, nor especially the document itself, is yet suitable as a new Fortran standard. However it is high time the general computing public was given the opportunity to give detailed appraisal of the proposal: any further delay is likely to cause the proposed standard to lose credibility. In my experience of presenting Fortran 8X to existing Fortran users, there was a fair amount of support for the draft proposals as outlined at the WG5 meeting in Geneva in April 1984, but there is growing impatience over the delay in completing the drafting process.

I shall postpone detailed and technical comments until the public review: the following are mainly general observations and impressions.

1. The document is a considerable improvement over earlier versions but still appears lacking in structure and consistency of style. Terms are not defined in a logical sequence but appear randomly throughout the text. There are still too many forward references and definitions which appear in more than one place. Every term should be defined completely and only once. The general text is too verbose, tautological and repetitive: it should be concise and precise. There is a logical structure to the document as a whole, but there is no consistent structure within a chapter.

2. The examples within the text are helpful although in some cases more explanation is needed (eg lines 30 - 40 on page 4-7). The Section Notes should be read with the text, which involves continual flipping back and forth while reading the document. I suggest the text and syntax definitions be printed on odd numbered pages only with the facing even numbered pages used for the Section Notes and examples.

3. I find the form of BNF used for the Syntax Rules extremely tedious and difficult to follow. A more diagrammatic form is needed: what was wrong with the "railroad" diagrams of FORTRAN 77 which proved very popular with users? I understand there were some discrepancies with the text but that is no reason to abandon it. Also, surely the word "help" in line 32 of page 1-2 should be deleted?

4. Appendix F should be deleted totally. I have changed my mind since the Halifax meeting; I now feel it only confuses the reader and adds unnecessary bulk.

5. I take issue with the second sentence of section 1.1 page 1-1. I do not feel this standard does anything to improve portability or reliability of Fortran programs! The statement on conformance (section 1.4) adds nothing to that in the Fortran 77 document. As a minimum, it is essential to have a much stronger statement on conformance along the lines of Halifax Resolution 10. Compiler writers must not be allowed to introduce extensions without at least giving the user the means to detect their use.

6. There seems to be an unnecessarily large number of similar terms used in Chapter 2: Program Unit, Subprogram, Procedure, some can be internal and external, some can't - all very confusing! Do we really need all these different terms, and even if we do how are they inter-related? A diagram is needed. From reading page 2-4 it is not easy to see what the difference is between a Procedure and a Subprogram: it is only upon reaching Chapter 12 that Procedures are explained more fully.

7. I don't like the term "module". It is frequently used elsewhere for other purposes - eg a group of subprograms. Also "modularity" is something different. The term "module" does not convey the sense of globality. I suggest changing the word used in this context to "globule".

8. A few minor errors:

- line 21 page 1-4 replace "protects" by "attempts to protect".
- lines 11-13 page 1-5 "core conforming" appears twice, once with a hyphen and once without. What is meant by "core"?
- line 1 page 2-8 replace "must" by "does".
- line 18 page 2-8. It is bad practice to use definitions like "A scalar is a datum that is not an array" especially when "array" has not yet been defined!
- line 40 page 4-4 gives 2.1 as an example of a signed constant. This is true according to rule R404 but why do we need two rules, R404 and R405, when the only difference is the presence of a sign which is optional anyway!
- line 25 page 5-2 replace "4.3.1.1" by "4.4.1.1"

J.D. Wilson  
December 17, 1986

74

## RESPONSE TO J D WILSON BALLOT

It is time that the Fortran draft revision to X3.9-1978 is submitted to X3 and SC22 for further processing. The draft that resulted from after the first ballot is a complete and well thought out draft. It is sometimes difficult to bring the national and international Fortran experts into agreement on some of the issues, as we discovered at the Halifax meeting in August of 1986. There was considerable support at Working Group 5 that all of Appendix F should have remained in the draft standard. However, it is important that due process is carried out in producing a standard and that an effort to achieve consensus is made. A careful effort has been made in the past year to do this. X3J3 has depended on the interest and support shown by the international community throughout the development of this standard.

You have seven observations and some minor editorial corrections. The corrections have been considered by full committee and either accepted or rejected. Comments on your seven points follow.

1. **Logical Structure of Sections.** The logical structure of the chapters emphasizes data types and the use of data in assignments, followed by executable constructs in Fortran, and I/O explained for various data objects. You may feel that it is not the best arrangement to describe program units at the end of the standard, rather than at the beginning when global concepts are under discussion. The draft standard is not a tutorial, but a specification, and the reader, in a sense, needs to know everything at once. This makes it necessary to have redundant definitions. The arrangement points out, we felt, the importance of data objects and their manipulation in today's Fortran.
2. **Examples and Section Notes.** There were a number of editorial comments that improve the examples in the text. The Section Notes are not allowed in the text of the standard according to the "ANSI Style Manual".
3. **Syntax Rules (BNF).** The railroad syntax did not define the language well enough. It was not possible to introduce important constraints and restrictions. The BNF follows commonly used rules for a more precise definition. The word "help" on page 1-2 is important because the text of the standard always takes precedence over the syntax rules if there is a discrepancy.
4. **Appendix F.** It is true that many people are ambivalent about including Appendix F. It is placed in the draft for review only. It will be removed from the final standard.
5. **Conformance.** At the 103rd meeting, we passed a stronger statement on conformance introduced by your colleague, Miles Ellis. This was in response to the resolutions at Halifax.

6. **Section 2 Terminology.** Program units are explained in detail in Sections 11 and 12. Again, I suggest that it is necessary to know everything at once in order to read a specification. Initial definitions for program units are necessary early in the text to explain the global concepts required to understand a Fortran program.
7. **Module.** The committee spent many meetings debating this term. Initially, it was called a "bundle" with the assumption that at some future time the term would be changed. It is different from an Ada package, and "package" would not be appropriate. Since we had introduced the concept of "core + modules" and rejected the initial definitions, we finally agreed to use the term "module".

# WG5 INFORMAL LETTER BALLOT

for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

*See the attached comments.*

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

or  
Name

CANADA

please print

Individual  
Signature

Graham Wilson

Date

7/JAN/86

PG7

167

IBM Canada Ltd.

895 Don Mills Road  
North York, Ontario  
M3C 1W3

For personal contact

GRAHAM WARREN  
DEPT. 72/123/895/TOR

TEL: (416)-448-2318

7/JAN/87

Dear Jeanne,

I enclose the W95 informal letter ballot and comments from the Canadian Fortran group. We support the submission of an edited version of the current S8 to SC22 for further processing, as we feel it is important to have a public review at this time. However, at present it is not clear that we would support the adoption of that document as the next Fortran standard.

As I mentioned on the phone, would you kindly arrange for the X3T3 material (minutes and pre-meeting distribution) sent to the Canadian group to be directed to me, at the above address, instead of to Bert Buckley. Bert is on sabbatical in France, and I am handling the distribution of that material to the members of the Canadian Fortran group.

Best wishes,

Graham Warren

P68

Cable: INBUSMACH  
Telex: 06-966574

168



Comments on S8 v103 from the Canadian Fortran Group

Technical Comments

1. We support the reduction in the size of the language which has been effected by the compromise solution (by simplifying some of the features or moving parts of the language to the "Removed Extensions" appendix), in particular:
  - removal of vector-valued subscripts
  - simplified REAL(\*,\*)
  - simplified IDENTIFY
  - simplified internal procedures
  - reduced number of array intrinsic functions
2. One removed feature which we would prefer to see reinstated in the language is the significant blank in free-form source. We feel that if this feature is ever to be introduced into Fortran, then now must be the time (Halifax WG5 resolution #14).
3. We would like to see a pointer facility in the proposed standard (Halifax WG5 resolution #11).

Editorial Comments

- |                       |  |
|-----------------------|--|
| page ii, line 45      | Change "to be removed" to "to be removed from future standards".   |
| page iii, line 31     | Change "(user defined)" to "(user-defined)".   |
| page iv, lines 15-16  | Change "describes the approximately one hundred intrinsic functions and two intrinsic subroutines" to "describes the 93 intrinsic functions and four intrinsic subroutines".   |
| page v, line 3        | Change "Section 14 (Entity Scope, Association, and Definition)" to "Section 14 (Scope, Association, and Definition)".  |
| page v, line 12       | Change "since that time" to "since April 1978".  |
| page 1-3, lines 38,41 | Change "-stmt" to "-stmt".   |
| page 1-5              | Why is the nature of deleted and obsolescent features defined in Chapter 1, yet the nature of deprecated features defined in Appendix B ? It would be better to have all 3 definitions together in one place.  |
| page 1-5, line 11-12  | The term "core conforming" is used without being previously defined. If it is not to be previously defined (in Section 1.4 "Conformance") there should at least be a reference here to the section of the document where it is defined (Appendix A, though even there the term is not explicitly defined). |
| page 2-8, line 28     | Change "a rank one array" to "a rank-one array".   |

P69

169

## RESPONSE TO CANADIAN FORTRAN GROUP

X3J3 appreciates your support for the compromise proposal developed after the first X3J3 ballot. Your detailed editorial comments were helpful in refining S8. They have been considered along with many others and either accepted or rejected.

You are undoubtedly correct in feeling that now is the time, if ever, when significant blanks should be introduced, as part of the free-form source. However, X3J3 feels that the insignificance of blanks is something that long-term users of Fortran like and want to retain. The issue is mentioned in Appendix F, and since there was considerable publicity when significant blanks were adopted in 1980, we expect to learn during the public review whether the decision to remove them is in accordance with today's users of the language.

Prior to the first S8 ballot, that failed in X3J3, the functionality of pointers for arrays was provided by the IDENTIFY statement. In an attempt to simplify and reduce the size of the language, the IDENTIFY statement was restricted to identification on a single host, thus removing the pointer capability. A similar functionality for scalar entities has never been described by X3J3. There is considerable pressure to forward the document for review as soon as possible. Adding new functionality at this time would delay that process and would be contrary to the direction that was mandated by the first unsuccessful ballot.

Message 5:  
From Z3000TT@AWITUW01.BITNET Fri Jan 9 09:03:30 1987  
Received: Fri, 9 Jan 87 09:03:10 pst from WISCVM.WISC.EDU by 111-crg.ARPA (4.12/  
)  
id AA20040; Fri, 9 Jan 87 09:03:10 pst  
Received: from (Z3000TT)AWITUW01.BITNET by WISCVM.WISC.EDU on 01/09/87  
at 11:02:47 CST  
Message-Id: <870109165234.000012F4.ABFC.TU@AWITUW01>  
Date: Fri, 9 Jan 87 16:52:34 MEZ  
From: Z3000TT@AWITUW01.BITNET (G SCHMITT TU Wien).  
Subject: :inocc,tt  
To: martin@111-crg.arpa

Dear Jeanne,

This is the official Austrian vote on S8.103, approved on our SC22 level.

Even if we are unhappy with some parts of S8 we vote

Yes, with comments supplied

Concerning the comments we refer to the comments made by Mr. Klaus Plasser  
and Gerhard Schmitt.

Kind regards, Gerhard  
Chairmann of ON/FNA001/A65 (correspond ISO/TC97/SC22) Programmiersprachen

# WG5 INFORMAL LETTER BALLOT

for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)

as defined by Halifax WG5 Resolutions

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country

~~Name~~ WEST - Germany  
please print

Individual  
Signature

Karl-Hinrich Reikhs

Date

1986-12-19

P 78

172

# WG5 INFORMAL LETTER BALLOT

## for individual WG5 members

Subject: Revision of ISO 1539-1980  
Reference Document: X3J3/S8.103

Date: Nov. 18, 1986  
Ballot Period: Dec. 1 - Jan. 5, 1987

Question: Do you approve the draft Fortran Revision (Fortran 8X) of ISO 1539-1980 (Fortran 77) enclosed with this ballot for submission to SC22 for further processing as an International Standard?

- Yes
- Yes, with comments supplied
- No, with comments supplied

Comments: (Additional pages of comments may be attached.)  
One page of comment is attached.

Mail to:

Jeanne T. Martin  
L-300  
Lawrence Livermore National Laboratory  
Livermore, CA 94550  
USA

Country  
or  
Name United Kingdom  
please print

Individual  
Signature *D. J. Maxwell*

Date December 19, 1986

P 79

173

WGS INFORMAL LETTER BALLOT. COMMENTS ACCOMPANYING THE U.K. VOTE.

The U.K. votes "yes" in the WGS informal letter ballot in order that the document may be issued for public comment. The vote does not indicate that the document is yet ready to be registered as a draft proposed standard.

The U.K. commends X3J3 for responding favourably to several of the WGS Halifax resolutions, but expresses regret that no action has been taken on other resolutions, including numbers 10 (conformance), 13 (deprecated features) and 14 (significant blanks); moreover the U.K. records its disapprobation that X3J3 took action directly contrary to resolution 22 (name-directed I/O).

The U.K. reaffirms its support for the WGS Halifax resolutions.

Further explanation of these points is to be found in responses from individual U.K. WGS members.

## RESPONSE TO UK FORTRAN GROUP

You will be pleased to learn that conformance rules, similar to those proposed in Halifax, have been adopted by X3J3 at its recent meeting.

Deprecated features still exist, but the earlier list has been divided into two classes: obsolescent and deprecated. Obsolescent features are those that can be omitted from current Fortran programs today, as better features are provided in FORTRAN 77. These appear in S8 in a smaller type font. The deprecated features are listed in Appendix B. They can be omitted from Fortran 8x programs because the new features are considered "better". In a subsequent revision of the language, some of the deprecated features could become obsolescent. Thus a path still exists for the migration of features out of the language, but it will depend on whether the features actually fall into widespread disuse.

X3J3 feels that the insignificance of blanks is something that long-term users of Fortran like and want to retain. The issue is mentioned in Appendix F, and since there was considerable publicity when significant blanks were adopted in 1980, we expect to learn during the public review whether the decision to remove them is in accordance with today's users of the language.

X3J3 feels that it is inappropriate to have a syntax for name-directed I/O that is quite different from existing implementations, and therefore replaced it with namelist.

176



american national standards committee  
X3-information systems

operating under the procedures of the  
American National Standards Institute

105 (X) JCA-4  
P. 1  
doc. no.

: X3K5/87-14

date

: May 11, 1987

project

:

ref. doc.

: (none)

reply to

: J. R. Wood

IBM Corporation

E37/656-3

P. O. Box 12195

Research Tri. Pk.,

NC 27709

(919) 254-0182

Mr. James H. Matheny, X3J3-VR  
Computer Sciences Corporation  
Infonet Division  
2100 E. Grant Avenue  
El Segundo, CA 90245

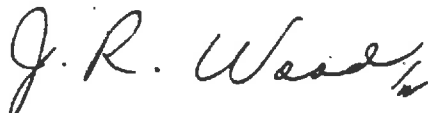
Dear Mr. Matheny,

I am the new Chairman of ASC X3K5 - Vocabulary for Information Systems. Mr. Bill Rinehuls, Chair of X3/SPARC, asked the previous K5 Chair to include COBOL and Fortran glossaries in the proposed American National Standard Dictionary for Information Systems (ANSDIS). We received a draft proposed Fortran glossary, dated May 7, 1986, by Mr. John Reid. This draft appears to need additional work before it can be published. For example, the glossary needs to be converted to ANSDIS style, there are questions regarding intended meanings, and there are some definitions that are circular: the definition uses the word that is being defined. Also, we believe that additional terms should be defined.

Changes proposed by X3K5 are explained under "General Comments." You will also find, under "Remarks", further elaborations on our rationale for changes and in some cases, questions that should be resolved before this glossary is published as an Appendix to ANSDIS.

One of my objectives during my tenure as Chair of X3K5 is to establish a more direct, interactive relationship with other TCs so as to better represent their specific terminologies in ANSDIS. The edited FORTRAN Glossary will be an Agenda item for the July 29 - 31 meeting of X3K5 at CBEMA. We welcome your participation and the participation of Mr. Reid, however, I do not have Mr. Reid's address, and he is not listed in the February '87 edition of X3/SD-6: Membership & Officers. If neither of you can attend, your comments will be appreciated.

Sincerely,



John R. Wood  
Chairman, ASC X3K5

cc: Jeanne Adams, Chair X3J3  
Catherine A. Kachurick, X3 Secretariat/CBEMA  
William C. Rinehuls, Chair X3/SPARC  
Josephine Walkowicz, former Chair X3K5  
X3K5 Distribution List

162nd Meeting of ASC X3K5  
Attachment 7.0(d)  
Doc. No: X3K5/87-13

## APPENDIX B: FORTRAN GLOSSARY

The terms in this appendix are defined according to their meaning in Fortran, as specified in *Programming Language Fortran, ANSI X3.9-1978*. These terms may have different meanings in other programming languages.

Italicized terms within a definition are defined either in this appendix or in the body of the dictionary. For Fortran, meanings given in this appendix take precedence over those given in the body of the dictionary.

### General Comments:

The source for this glossary is the *S8 Glossary of Technical Terms*, May 7, 1986, by John Reid (X3J3 document 100(13)JKR-2). This is a draft of a proposed appendix in the ANSI X3.9 revision. It has not yet been approved by X3J3 for inclusion in /S8. The glossary is therefore subject to further X3J3 review/comment and is subject to technical changes.

Stylistic changes have been made to the original Fortran glossary for consistency with the style of the proposed American National Standard Dictionary for Information Systems (ANSDIS). Technical Committee ASC X3K5 recommends that this style be adopted in future drafts of the Fortran standard glossary so that they can be copied directly into future editions of ANSDIS without stylistic modification. Questions and additional explanations of changes are given under "Remarks."

The intended audience of ANSDIS is the general information systems population, which includes students and occasional users of information systems. Prior knowledge of Fortran terminology and concepts cannot be assumed.

### Specific Comments:

- The original Fortran glossary does not cross-reference terms, whereas terms in ANSDIS are heavily cross-referenced. Cross-references have been added (see *access, area*, etc.).
- In ANSDIS definitions, every term that is defined elsewhere in the dictionary is italicized. Italics have been added in all such cases. Note that when an italicized term follows another italicized term, the terms may in some cases be defined separately while in other cases the combined-word term may be defined; for example, in *alias association*, the combined-word term is defined but in *alias array*, "alias" and "array" are defined separately.
- References to paragraphs, given in the original Fortran glossary, have been deleted as they are inappropriate (and meaningless) in the context of ANSDIS.

ppp 71

- References to other documents, such as "See BNF R1212" are treated as notes. In such cases, the complete title of the document should be given.
- Some text that is not essential to the definition of the term are treated as notes, for example see *access*.
- Some of the definitions given in the original Fortran glossary are circular; that is, the definition uses the term that is being defined. For example, see *conformance* and *connected* in the original glossary.
- ANSDIS style uses the opening phrase "Pertaining to ... " for definitions of modifiers, for example, see *conformable* and *connected*.
- The term being defined should not be repeated at the beginning of a definition; for example, in the definition of *entity* the phrase "The term entity is used for any of the following ..." should be deleted.
- Some terms are described but not formally defined. For example, in *association*, the reader is told when an association exists but is not told what an association is. In such cases, an alternative definition is proposed.
- Some terms, such as "construct", "lexical element", and "lexical token" are used in definitions but are not defined. Definitions should be added.
- In one instance, a Fortran definition conflicts with the definition given in the body of ANSDIS. See the Remarks under *processor*.
- See "Remarks" for questions that need to be answered or where additional information is needed.

RTP 71

**access.** The means by which a *scoping unit* accesses *entities* in a *module subprogram* or, in the case of an *internal procedure*, in its *host*. Such entities may be explicitly or implicitly accessible.

**Note:** In Fortran, access is provided by the USE statement.

**Remark:** The term "access" should not be used in the definition of access.

**alias.** A *data object* that has a *data type*, *type parameters*, and a *rank*. It may not be referenced or defined unless it is associated with a data object or *subobject* that may be referenced or defined. If it is an *array*, it does not have a *shape* unless it is alias-associated. See *parent of an alias*.

**alias association.** The association of an *alias* with a nonalias object, following a valid *execution* of an IDENTIFY statement.

**Remark:** "Object" is not defined. Is the term *data object*, which is defined, intended here?

**allocatable array.** A named *array* that has a *data type*, *type parameters*, and a *rank*, but that has a *shape* and may be referenced or defined only when it has space allocated for it.

**Remark:** *Whole array* is defined as "a named *array*." Should "named *array*" in this definition be changed to *whole array*?

**argument.** See *dummy argument*.

**argument keyword.** A *dummy argument* name. It may be used in a *procedure reference* before the equals *symbol* provided that the procedure has an explicit *procedure interface*.

**Note:** See BNF R1212.

**Remark:** What is the complete title of the referenced standard?

**array.** A *set* of *data*, all of the same *data type* and *type parameters*, whose individual elements are arranged in a rectangular pattern. It may be a named array, an *array section*, a *structure component*, an array-valued *function* result, or an array-valued *expression*. It has a *rank* of at least one. See *allocatable array*, *deferred-shape array*, *many-to-one alias array*, *sequence array*, *whole array*.

**Remark:** *Whole array* is defined as "a named *array*." Should "named *array*" in this definition be changed to *whole array*?

**array element.** The *scalar data* that make up an *array*.

**array section.** An *array subobject* designated by the *symbolic name* of an array with a section *subscript* list, optionally followed by a *substring* range.

**Note:** See BNF R612.

**Remark:** What is the complete title of the referenced standard?

**association.** Proposed definition:

A relationship between an *entity* and one or more *scoping units*. An association exists when an entity is identified by different names in the same *scoping unit* or

RTP 71

by the same name or by different names in different scoping units. See *alias association*, *storage association*.

Remark: The original text does not define "association". It only describes when it exists.

**attributes.** Properties of a *data object* that may be specified in a type declaration *statement*, namely *data type*, *type parameters*, *rank*, *shape*, whether *variable* or *constant*, initial value, accessibility (PUBLIC or PRIVATE), intent (IN, OUT, or INOUT), whether allocatable, whether *alias*, whether optional, whether to be saved, and whether ranged. See *value attribute*.

Note: See BNF R501.

Remark: What is the complete title of the referenced standard?

**belong.** Proposed definition:

A relationship between a *statement* and a construct. If an EXIT or CYCLE statement contains a construct name, it belongs to the DO construct using that name. Otherwise, it belongs to the innermost DO construct in which it appears.

Remark 1: The original text does not define "belong". It describes when it exists.

Remark 2: "Construct" is not defined.

**block.** A *sequence* of executable constructs embedded in another executable construct, bounded by *statements* that are particular to the construct, and treated as an integral unit.

Remark: "Construct" is not defined.

**component.** Proposed definition:

Any of the *set* of *entities* that make up a *data type*.

Remark 1: The original text defines "derived type", not "component." According to the definition of *entity* a component is an entity.

Remark 2: The original text says that a derived type is a set of components. Isn't any data type a set of components (the components being parameters and values)?

**condition.** A named circumstance in which it is inappropriate to continue the normal *execution sequence*.

Remark: Isn't this an "abnormal condition" or "error condition?" All conditions do not preclude normal execution.

**conformable.** Pertaining to *arrays* that have the same *shape*. A *scalar* is conformable with any array.

Remark: In ANSDIS style, definitions of modifiers take the form: "Pertaining to ..."

**conformance.** Pertaining to an *executable program* that uses only the forms and relationships described in *Programming Language Fortran, ANSI X3.9-1978* or to a *program unit* that can be included in an executable program in a manner that allows the executable program to comply with that standard. See *core conformance*.

Remark: The original definition is circular because it uses "conforms" and "conforming" in the definition of *conformance*.

**connected.** Pertaining to a unit that refers to a *file* which refers to the unit.

Remark: The original definition is circular because it uses the term *connected* twice in the definition of *connected*. Does "connection" infer a two-way reference between a unit and a file?

**constant.** A *symbolic constant* a *literal constant*, or a *subobject* of a constant containing no *expression* in its *designator* that is not a constant expression. See *literal constant*, *symbolic constant*. Contrast with *variable*.

**construct.** See *executable construct*.

**core.** The *set of language facilities* that are not identified as *deprecated features*. It is a complete language.

**core conformance.** Pertaining to a standard-conforming program that contains no *deprecated features*.

**data entity.** An *entity* that has or may have a *data* value.

**data object.** A named datum or a *set of data* of the same type and *type parameters* that has a *symbolic name* and may be referenced as a whole. It may be a named *variable* or a *symbolic constant*. See *definition of a data object*.

**data type.** A *set of data* values, together with a way to denote these values and a collection of *operations* that interrupt and manipulate the values. A type may be parameterized, in which case the set of data values depends on the values of the *parameters*.

**deferred-shape array.** An *allocatable array* or an *alias array*.

**definable.** A *variable* whose value may be changed as a whole. *Constants*, *allocatable arrays* that have not been allocated, *many-to-one alias arrays*, and *aliases* for which an *IDENTIFY statement* has not been executed are examples of *data objects* that are not definable. Many-to-one vector-valued *array sections* are examples of *data subobjects* that are not definable.

Remark: Should "vector" be defined in a Fortran sense, or does the definition given in the body of ANSDIS suffice?

vector

(ISO) A quantity usually characterized by an ordered set of *scalars*.

**defined assignment statement.** An *assignment statement* that is not an *intrinsic assignment statement* and is defined by a *subroutine subprogram* whose *interface* is explicit.

Remark 1: *Assignment statement* is defined in the body of ANSDIS. Does the term need a Fortran-specific definition?

assignment statement

An *instruction* used to express a *sequence of operations* or used to assign *operands* to specified variables, or *symbols*, or both.

Remark 2: *Interface* is defined in the body of ANSDIS. Does the term need a Fortran-specific definition?

**interface**

(1) (ISO) A shared boundary between two *functional units*, defined by functional characteristics, common physical interconnection characteristics, *signal* characteristics, and other characteristics, as appropriate. The concept involves the specification of the *connection* of two devices having different functions.

(2) A point of communication between two or more processes, persons, or other physical entities.

**defined operation.** An *operation* that is not an *intrinsic* operation and is defined by a *function subprogram* whose *interface* is explicit.

**definition of a data object.** The assignment of a valid value to a valid *variable* during *program execution*. Under certain other circumstances, the variable ceases to have a valid value and becomes undefined.

**definition of a procedure or type.** The definition of a *procedure* by a *subprogram* or the definition of a *derived type* by a *sequence of statements* comencing with a *TYPE statement* and terminating with an END TYPE statement.

Note: See BNF R417.

Remark: What is the complete title of BNF R417?

**deprecated feature.** An element of a *language* that is intended to be removed from the next version of the standard.

Remark: In ANSDIS style, the singular form of the term is usually defined.

**derived type.** A type whose *data* have *components of intrinsic* types and other derived types.

Remark: What are the "other derived types?" No others are defined.

**designator.** A *symbolic name*, followed by any number of *component* selectors, *array section* selectors, *array element* selectors, and *substring* selectors.

Remark: Should "selector" be defined?

**dummy argument.** An *entity* whose name appears in a dummy argument list in a FUNCTION or SUBROUTINE *statement*.

Remark: The definition is circular because "dummy argument" is used in the definition of "dummy argument."

**element.** See *array element*.

**elemental.** Pertaining to an *operation*, *function*, or assignment that is applied independently to the elements of an *array* or to corresponding elements of a set of *conformable* arrays and *scalars*.

**entity.** A *program unit*, a *procedure*, an *operator*, an *interface block*, a *common block*, an *input-output unit*, a *statement function*, a *type*, a *named variable*, an *expression*, a *component of a type*, a *symbolic constant*, a *statement label*, a



construct, an *exponent letter*, a range list, or a *condition*. See *data entity*, *global entity*, *local entity*, *statement entity*.

Remark: Should "common block" be defined? Also, "exponent" and "letter" are separately defined in the body of ANSDIS, but do these combined definitions convey the meaning intended in Fortran?

**executable construct.** DEFINITION NEEDED.

Remark: The reference to BNF R219 may be sufficient for the purposes of the Fortran Standard, but if the term is important it should be defined in ANSDIS Appendix B: Fortran Glossary. "Construct" should also be defined.

**executable program.** One *main program* and any number (including zero) of *subprograms*.

**executable statement.** An instruction to perform or control one or more computational actions. The executable *statements* are all those that make up the syntactic class of *executable constructs*.

**expression.** A construct formed from *operands*, *operators*, and parentheses that may be a *variable*, a *constant*, a *function* reference, or that may represent a computation.

Note: See BNF R715.

Remark 1: The original definition uses "expression" in the definition of "expression"

Remark 2: What is the complete title of BNF R715?

**extent.** The size of one dimension of an *array*.

**external procedure.** A nonintrinsic *procedure* that is defined by an *external program unit*. Contrast with *internal procedure*

**external program unit.** A *program unit* that is not contained within another program unit. Contrast with *internal program unit*.

**feature.** See *deprecated feature*.

**form.** Any of the three types of *records*: formatted, unformatted, and endfile.  
Remark: The original definition uses "form" in the definition of "form."

**function.** A *procedure* that is invoked in an *expression*.

**function subprogram.** A *subprogram* whose first *statement* is a FUNCTION statement.

Note: See BNF R204.

Remark: What is the complete title of BNF R204?

**global entity.** An *entity* identified by a lexical token whose scope is an *executable program*. It may be an *external program unit*, a *common block*, or an *external procedure*.

Remark: "Common block", "lexical token", and "scope" are not defined in a Fortran context, and they are not defined in the body of ANSDIS.

RTP 71

**handler.** A *sequence of statements* commencing with a HANDLE statement and ending with the statement before the next HANDLE or END ENABLE statement, whichever comes first. A handler is executed when a *condition* specified for it is signaled.

**host.** A *program unit* that immediately contains an *internal procedure*. In the case of nested internal procedures, the host is the program unit that immediately contains the internal procedure. For example, if A contains B and B contains C, A is the host of B and B is the host of C, but A is not the host of C.

Remark: What is the meaning of "immediately containing?" Does "immediate" mean "first" or "at the beginning of"?

**instance of a subprogram.** A *subprogram* created when a *function* or *subroutine* defined by a *procedure* subprogram is invoked.

**interface.** See *procedure interface*.

**interface of a procedure.** See *procedure interface*.

**internal procedure.** A *procedure* that is defined by an *internal program unit*.

**internal program unit.** A *program unit* that is contained within another program unit.

Remark: Is this also called a "nested program unit"?

**intrinsic.** Pertaining to types, *operators*, *procedures*, and *conditions*, defined in the ANSI Fortran Standard, that may be used in any *scoping unit* without further definition or specification.

Remark: The original definition uses "intrinsic" in the definition of "intrinsic".

**invoke.** (1) To *call* a *subroutine* by means of a CALL *statement* or by a defined assignment. (2) To call a *function* by a *reference* to it during the evaluation of an *expression*.

Remark: "Invoke" is a verb; in ANSDIS style, the infinitive form (to invoke) is defined. *Call* is defined in a general sense in the body of ANSDIS.

**keyword.** A *statement keyword* or an *argument keyword*.

**length.** The number of *characters* in a *character string*.

**lexical element.** DEFINITION NEEDED.

Remark: The term is used in the definition of *statement entity* but it is not defined.

**lexical token.** DEFINITION NEEDED.

Remark: The term is used in the definitions of *global entity*, *literal constant*, and *local entity* but it is not defined. How does a lexical element differ from a lexical token? Is the element a part of the token?

**literal constant.** A lexical token that directly represents a *scalar* value of *intrinsic* type.

Note: See BNF R305.

Remark 1: What is the complete title of BNF R305?

Remark 2: "Lexical token" is not defined.

**local entity.** An *entity* defined by a lexical token whose scope is a *scoping unit*.  
Remark: Neither "lexical token" or "scope" are defined.

**main program.** A *program unit* that is not a *subprogram*.

**Note:** See BNF R203.

Remark: What is the complete title of BNF R203?

**many-to-one alias array.** An *alias array* that has more than one of its elements associated with the same datum.

**many-to-one vector subscript.** A *vector subscript* that has two or more elements with the same value.

**module.** An *external program unit* that contains or accesses definitions to be accessed by other *program units*. See *standard module*.

**name.** See *symbolic name*.

**object.** See *data object*, *subobject*.

Remark: *Subobject* is defined, but "object" is not defined. For consistency and completeness, both should be defined. Is object (used in the definition of *structure*) always synonymous with data object in the context of Fortran?

**operation.** See *defined operation*.

**operator.** A specification of a particular computation involving one or two *operands*.

Remark: *Operand* is defined in the body of ANSDIS:  
(ISO) An entity on which an operation is performed.

**parameter.** See *type parameter*.

**parent of an alias.** The *data object* or *subobject* with which an *alias* appears in an IDENTIFY *statement*.

**present.** Pertaining to a *dummy argument* whose *procedure* has been invoked if there is a corresponding actual *argument* and the actual argument is a dummy argument that is present in the invoking procedure or is not a dummy argument of the invoking *program unit*.

Remark: Is this the only context in which the common English word "present" is used in Fortran?

**procedure.** A computation that may be invoked during *program execution*. It may be a *function* or a *subroutine*. A procedure *subprogram* may define more than one procedure if it contains ENTRY statements. See *definition of a procedure or type*, *external procedure*, *internal procedure*.

**procedure interface.** The characteristics of a *procedure*; the name of the procedure; the name of each *dummy argument*; if it is a *function*, the *operator*, if

any, by which it may be referenced, and, if it is a *subroutine*, whether it may be referenced by an *assignment statement*.

**processor.** The combination of a *computing system* and the mechanism by which *programs* are transformed for use on that computing system.

Remark 1: This definition conflicts with the more rigorous definitions of "processor" and "computing system" already contained in ANSDIS and approved by both X3 and the International Customs Council. In today's usage, a processor is often a microchip; it is not a computing system. Computing systems contain processors. Processors do not contain computing systems:

**processor**

(ISO) In a *computer*, a *functional unit* that *interprets* and *executes instructions*. A processor consists of at least an *instruction control unit* and an *arithmetic unit*.

**computer system**

A *functional unit*, consisting of one or more *computers* and associated *software*, that uses common *storage* for all or part of a *program* and also for all or part of the *data* necessary for the *execution* of the program; executes user-written or user-designated programs; performs user-designated data manipulation, including *arithmetic operations* and *logic operations*; and that can execute programs that modify themselves during their execution. A computer system may be a stand-alone unit or may consist of several interconnected units. Synonymous with ADP system, computing system.

Remark 2: These more rigorous definitions, called for and approved by the International Customs Council, must be retained in the body of ANSDIS. Since "processor" is a general DP term, not a Fortran term, the entry should be deleted from the Fortran Glossary.

**program.** See *executable program*, *main program*, *subprogram*.

**program unit.** The fundamental component of a *Fortran program*; a *sequence of statements* and comment lines. It may be a *main program* or a *subprogram*. See *external program unit*, *internal program unit*.

Remark: Is "component" intended here with the meaning defined in Fortran or is it intended in the general sense of the word?

**rank.** The number of dimensions of an *array*. It is zero for a *scalar*.

**reference.** The appearance of a *data object name* or *subobject designator* in a context requiring the value at that point during *execution*, or the appearance of a *procedure name*, its *operator symbol*, or the assignment symbol in a context requiring execution of the procedure at that point.

Note: The act of defining a variable is not regarded as a reference.

**scalar.** A single datum that is not *array-valued*.

**scope.** DEFINITION NEEDED. Remark: The term is used in several definitions but is not defined. Is "range" a synonym?

**scoping unit.** One of the following:

1. A *derived-type* definition,
2. A *procedure interface block*, excluding any procedure interface blocks contained within it, or
3. A *program unit*, excluding derived-type definitions, procedure interface blocks, and *program units* contained within it.

**section.** See *array section*.

Remark: Is every section an array section or should section be defined? See the use of "section" in the note under *substring*.

**sequence.** See *storage sequence*.

**sequence array.** An assumed-size *array* or an explicit-*shape* array without the RANGE *attribute* that is either a dummy array associated with a sequence array or is not a *dummy argument*.

**shape.** The *rank-one array* whose elements are the *extents* in each dimension.

**signal.** DEFINE OR DELETE.

Remark 1: The original text describes when a condition is signaled. It is not a definition of *signal*. Signal is already defined in the body of ANSDIS:

signal

(1) (ISO) A variation of a physical quantity, used to convey *data*.

(2) A time-dependent value attached to a physical phenomenon to convey *data*.

Remark 2: If these definitions suffice for Fortran, delete "signal" from the Fortran Glossary. If they do not suffice, define "signal" in in a Fortran-specific sense.

**size.** For an *array*, the total number of elements.

Remark: The term "size" applies to many things other than arrays. Why does the Fortran Glossary define it specifically for arrays?

**standard module.** A *module* standardized as a separate collateral standard. Its *interface* must conform with the *core*.

Remark: The meaning of "collateral" here is unclear. Webster's:

- 1.a. Accompanying as secondary or subordinate. 1.b. Indirect.
2. Parallel, coordinate, or corresponding in position, order, time, or significance.

If "secondary" or "subordinate" is intended, replace "collateral" with the appropriate word.

RTP 71

statement. See *defined assignment statement, executable statement.*

statement keyword. A *word* that is part of the *syntax* of a *statement* and that may be used to identify the statement.

statement entity. An *entity* identical from a lexical element

Remark: "Lexical element" is not defined. How does it differ from a "lexical token" (also not defined)?

storage association. An association of *storage sequences* in which a storage unit of one is the same as a storage unit of another.

Remark: What is a "storage unit" in the context of Fortran? Do we mean the *data* contained in the storage unit of one storage sequence is the same as the data contained in a storage unit of another sequence? "Storage unit" implies space in storage, not data necessarily.

storage sequence. A *sequence* of storage units.

Remark: Without a definition of "storage unit" this definition is not meaningful. If all of storage consists of storage units, then is all of storage a storage sequence? Is any arbitrary collection of storage units a storage sequence? Can storage sequences contain storage sequences?

storage unit. DEFINITION NEEDED.

stride. The increment specified by a section *subscript* triplet.

Remark: What is a "section subscript triplet"? Neither section nor triplet is defined. Does section imply *array section*, which is defined?

structure. An object of *derived type*.

Remark: "Object" is not defined. Is *data object*, which is defined, the intended term here?

subobject. Part of a *data object*. It may be an *array element*, an *array section*, a *structure component*, or a *substring*.

subprogram. A *function subprogram*, a *subroutine subprogram*, a *module subprogram*, or a *block data subprogram*. See *function subprogram, instance of a subprogram, subroutine subprogram*.

Note: See BNF R204, BNF R205, BNF R206, BNF R207.

Remark: What are the complete titles of the referenced documents?

subroutine. A *procedure* that is invoked by a *CALL statement* or an *assignment statement*.

subroutine subprogram. DEFINITION NEEDED.

The original Fortran Glossary gives only a reference to BNF R205. The term should be defined here.

subscript. An item of a list of subscripts that selects an element of a named *array* or an array-valued *structure component*. See *many-to-one vector subscript, vector subscript*.

Remark 1: The original text describes the selection of arrays or components by a list of subscripts but does not define the term subscript.

Remark 2: Does a subscript select or does the list of subscripts select? Needs further study.

**substring.** A contiguous portion of a *scalar character string*.

**Note:** An array section can include a substring-range; the result is called a section and not a substring.

Remark 1: The note appears to define *section* as: An *array section* that includes a substring-range.

Remark 2: "Substring-range" is not defined.

**symbolic constant.** A named *data object* whose value must not change during *execution* of an *executable program*.

**symbolic name.** An *alphanumeric* name for an *entity*.

**type.** See *data type, definition of a procedure or type, derived type*.

**type parameter.** A *parameter* of a parameterized *data type*.

**type parameter values.** The values of the *type parameters* of a *data entity* of parameterized *data type*.

**unit.** See *program unit, scoping unit*.

**value attribute.** An *attribute* that describes whether a *data object* is *constant* or *variable* and whether it has a defined initial value.

Remark 1: The original text is not a complete sentence.

Remark 2: Should this read: "... whether a *data object* or *subobject* ..."? See next entry.

**variable.** A *data object* or *subobject* that is not a *constant*. It may be a symbolic variable, an *array element*, an *array section*, a *structure component*, or a *substring*.

**vector subscript.** A section *subscript* that is a *rank-one integer expression*.

**whole array.** A named *array*.

IRTP 71





105 (\*) JCA-5

5

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Scientific Computing Division/Advanced Methods Section

P. O. Box 3000 • Boulder, Colorado • 80307

Telephone: (303) 497-1275 • FTS: 320-1275

May 20, 1987

Stanley Allen, Secretary SMC  
CBEMA, X3, Information Processing Systems  
311 First Street, N. W., Suite 500  
Washington, D. C., 20001-2178

Dear Mr. Allen;

Jeanne Martin, X3J3 Secretary, has resigned to spend more of her time as convener of Working Group 5/SC22. She has done an excellent job as secretary for five years and deserves to have less responsibility in this area. As you know, this is a very demanding officer position in the TCs.

I have appointed John Reid to take her place. Dr. Reid has much experience in recording the work of X3J3; he has prepared a meeting summary for distribution after each meeting and is actively involved in our work. His address is:

Dr. John Reid  
Computer Science and Systems Division  
Building 8.9  
Aere Harwell  
Didcot, Oxfordshire OX11 0ra  
England

Will you notify the SMC of this appointment?

Sincerely yours,

Jeanne Adams, Advanced Methods Group  
Chair, X3J3

cc. Patty Steiner

193



**Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\***

105 (\*) ICA-6

Doc. No.: X3/87-04-090-X,S,M,T  
Date: April 21, 1987  
Project:  
Ref. Doc.:  
Reply to:

6

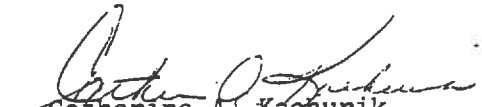
TO: X3, SPARC, IAC, SMC  
Officers, X3/TC's, SC's and SPARC/SG's

SUBJECT: Transmittal of the Revised Master Plan SD-1, May 1987, Revision 1

X3 approved the attached revision of the SD-1 Master Plan at their February 1987 meeting. The X3 Secretariat took the initial step to produce the SD-1 in the same format as the SD-2. i.e., that we would institute a change page procedure to the base document. Each page would be dated and identified by Revision number. Unfortunately, the document was so totally changed that we scraped that plan and we redid the whole thing. The next revision will follow our change page system. Please hold on to this SD-1 because it will be the last full copy you will receive.

Unlike that last version, this one has not gone to each name in our data base. Officers should reproduce the coupon below and we will fulfill the requests as they are received.

Sincerely,

  
Catherine A. Kachurik  
X3 Administrative Secretary

-----  
X3/SD-1 Request

Return to: Katrina Gray  
CBEMA/X3 Secretariat  
311 First Street, N.W.  
Suite 500  
Washington, DC 20001

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

TC/Status: \_\_\_\_\_

195



Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

**NEWS RELEASE**



For more information contact:  
Peter Bono, X3H3 Chairman  
203-464-9350

Date: May 11, 1987

\* \* \* NOW AVAILABLE AS A DRAFT STANDARD \* \* \*  
C LANGUAGE BINDING OF THE GRAPHICAL KERNEL SYSTEM (GKS)

X3 ANNOUNCES PUBLIC REVIEW AND COMMENT PERIOD ON  
DRAFT AMERICAN NATIONAL STANDARD X3.124.4-198x,  
C LANGUAGE BINDING OF GKS

Washington, D. C. -- X3, the Accredited Standards Committee on Information Processing Systems, announces a four-month public review and comment period on draft proposed American National Standard, X3.124.4-198x. The public review period extends from June 19, 1987 to October 19, 1987.

The American National Standard X3.124-1985, Graphical Kernel System is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages. The purpose of X3.124.4-198x is to define a standard binding for the C computer programming language.

It is expected that there will be a substantial positive impact on future software development due to the resulting portability and maintainability of applications code.

This draft standard is available for public review and comment for a four-month period ending October 19, 1987. Copies may be obtained from GLOBAL ENGINEERING DOCUMENTS, INC. by calling 800-854-7179 on the West Coast, or 800-248-0084 on the East Coast.

Single Copy Price: \$ 35.00  
\$ 45.50 (when ordered outside of the Continental U.S. +  
\$13.65 postage)  
# # # # #



**ANSI X3H2  
SQL2 CHANGE PROPOSAL**

Document Number: X3H2-87-123  
Date: May, 1987  
Author: Charles Ziering  
Subject: Date/Time Data Type  
References:  
1. X3H2-87-55 Examples of use of Date Time Data Types  
2. X3H2-87-56 Date Time Data Types: Proposal for SQL2  
3. X3H2-87-95 (ISO-ANSI working draft) Database Language SQL2

Summary

The separation of century and year in a date time value, with the option to omit the century, leads to a number of operational problems. Furthermore, the concept of a date that is not tied to a century is not especially useful. I propose that the designation of century distinct from "year within the century" be dropped from date time values, and that the year component be defined to include the century.

Discussion of Problems

1. 1900-02-29 is not a valid date since 1900 is not a leap year. But 2000-02-29 is valid since 2000 is a leap year. Thus you cannot check the validity of 00-02-29 [YY:DD] unless you know the century.
2. If you wish to add an interval 100 [DD:DD] to 00-01-01 [YY:DD], you cannot determine the correct answer without knowing the century.
3. Since we are all too rapidly approaching a century boundary, if users begin storing dates as [YY:DD], rather than [CC:DD], they will suddenly encounter problems when they try to extend them for comparisons or calculations after 2000-01-01.

EXTEND (promise-date, [CC:DD]) < CURRENT [CC:DD]

or

promise-date < CURRENT [YY:DD]

4. I can see no use for the concept of a date that is not associated with a century. Though "87-05-02" is a convenient shorthand for humans to use when recording dates, there is no misconception that it may mean 2087-05-02 versus 1987-05-02. It is up to an application to present and accept dates in a format most appropriate to a user, but it is the job of the database to store data in a consistently useful form.
5. If CC and YY are collapsed into the single field YY, the rules about subfields are much simpler.

Proposal Specifics

1. Remove all references to CC (century) and change all references to YY (year within century) to YYYY (year) in sections:
  - 4.2.4
  - 5.2
  - 5.4
  - 5.10
  - 7.3
2. Change 5.2 General Rule 3.b to add
  - "and YYYY, which is 4 digits in length"
3. Replace 5.3 Syntax Rule 8 Case a and b with
  - "X shall contain n <integer>s. The i-th <integer> corresponds to the i-th <datetime field>."
4. Remove 5.3 Syntax Rule 9 1st sentence. (Leave from "Case:" on)
5. Replace 5.3 Syntax Rule 11 with:
  - "Within the definition of a <datetime literal>, if YYYY is specified, then its <datetime value> shall contain at most 4 <digits>. All other <datetime value>s contain at most 2 <digit>s."
6. Replace 5.5 Syntax Rule 13 with:
  - "If INTERVAL is specified, then the precision of the <interval type> shall not include both the datetime fields MO and DD."
7. In 5.6 Syntax Rule 18, change "CC" to "YYYY" in the default precision.
8. Replace 5.6 General Rule 1 Case a and b with:
  - "X shall contain n values. The i-th value corresponds to the i-th datetime field."
9. Replace the lines for CC and YY in 5.6 General Rule 3 with:
  - YYYY: 1600 to 9999
10. Change "CC" in 5.10 Syntax Rule 2 to "YYYY".



105 (\*) JCA-9  
f. 1 of 2 (9)

Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109

May 6, 1987

Dr. Jeanne Adams, Chair X3J3  
Scientific Computing Division  
National Center for Atmospheric Research  
Box 3000  
Boulder CO 80307

Dear Dr. Adams:

In reading the responses to the ballot comments, I noticed the remarks (page 72, section Small Language, second paragraph) "...the number of intrinsic functions ... does affect the size of an 'implementation,' but not the size of the 'language.'" and (page 79, first paragraph) "The second request, for additional intrinsic functions, was rejected as it would increase the size of the language." Which is correct?

Sincerely,



W. Van Snyder  
Mail Stop 301-490

105 (X) JCA-9  
p. 2 of 2

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH  
Scientific Computing Division/Advanced Methods Section  
P. O. Box 3000 • Boulder, Colorado • 80307  
Telephone: (303) 497-1275 • FTS: 920-1275

May 27, 1987

W. Van Snyder  
Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109

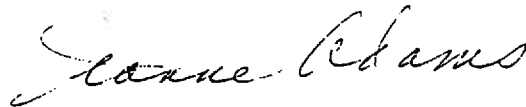
Dear Mr. Van Snyder;

The IBM response to ballot comments on page 72 of the pre-meeting distribution has been almost completely rewritten, before voted by X3J3. The apparent ambiguity has been removed. Page 79 of the pre-meeting still does remain in the formal responses to Working Group 5.

The final version of the response and ballot document will appear in the pre-meeting for the 105th meeting, and you can check up again by reading the X3J3 response to the IBM ballot commentary. I quote "While X3J3 does not believe that the "size" of a language can be adequately expressed by a count of statement types, syntax rules, vocabulary, and intrinsic functions, it is plain that Fortran 8x is considerably larger than FORTRAN 77." The response continues in defense of this position. Many of the comments were rewritten before X3J3 could vote to approve them. The material in the distribution was only a "talking" document, in anticipation of the final disposition of all of the responses in a formal vote by X3J3.

We appreciate your careful examination of these documents.

Sincerely yours,



Jeanne Adams, Advanced Methods Group  
Chair, X3J3

cc. X3J3 Distribution

202

105(\*) JCA -10

10

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Scientific Computing Division/Advanced Methods Section

P. O. Box 3000 • Boulder, Colorado • 80307

Telephone: (303) 497-1275 • FTS: 920-1275

May 28, 1987

R. H. Follett  
IBM Corporation  
649/RCTR/9E  
6705 Rockledge Dr.  
Bethesda, MD 20817

Dear Bob;

I would like to recommend Andrew Johnson for appointment to the SC22 TAG meeting in June.  
His address is:

Andrew Johnson  
MS 10C17-9  
Prime Computer Inc.  
500 Old Connecticut Path  
Farmingham, MA 01701

Andy has applied for International Representative, but we have not heard of an appointment as yet. Meantime, I have appointed Andy as acting International Representative.

Regards,



Jeanne Adams, Advanced Methods Group  
Chair, X3J3

cc. Andrew Johnson  
X3J3 Distribution ✓

203



105 (\*) JCA-11  
P. 1 of 1  
11

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Scientific Computing Division

P. O. Box 3000 • Boulder, Colorado • 80307

Telephone: (303) 497-1275 • FTS: 320-1275 • Telex: 45 694

May 28, 1987

William Rinehuls  
Chairman, SPARC  
8457 Rushing Creek Court  
Springfield, VA 22153

Dear Bill;

The final role call vote at the meeting in Seattle passed to forward the draft Fortran standard for further processing. Enclosed you will find a bound copy of the submittal package that I am sending Patti Steiner. 150 copies of the draft standard will be mailed to Patti in 10 days or so. They are not yet off the typesetter equipment.

I would like to comment on the two motions in your May 12 memo. First, I am delighted with the 2/3 vote rule for Milestone 11. I feel that if any comments are included with the package, all comments of a general nature should be included whether affirmative or negative. However, this should exclude any editorial comments, which in the case of Fortran could be so extensive as to "muddy" the waters in examining a draft standard. Perhaps this means that only negative comments should be included, but with Fortran, we received editorial comments on both affirmative and negative votes. In the enclosed document, I excluded editorial comments, but I included all comments of a general nature or ones that were requesting substantive changes to the draft.

Some members of X3J3 said that they would vote NO if their comments were not included. Early in the balloting I promised that all comments except editorial ones would be included in the submission package.

I also believe that the submission document should be a separately bound document. Some people will only be interested in the draft standard, especially if they follow the minutes and already have received all of the ballot information including X3J3 responses. In this case, they would not want to incur the page charge for the extra package. If you bound the two documents for Fortran together, it would be very large indeed, something over 400 pages. This might discourage some from obtaining a copy of the draft, possibly students or low budget persons. However, I do agree that during the review, all of the information should be available.

X3J3 does not meet until after your deadline, so this is my opinion as chair. I was sorry not to be able to deliver the annual report this year. If there is anything that I can do to facilitate the processing of the Fortran draft, please let me know.

Regards,

905



Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No.:

X3/87-05-127-X,M

Date:

May 27, 1987

Project:

Ref. Doc.:

Reply to:

105(X) JCA-12 (12)

Mr. E. Andrew Johnson  
Prime Computer, Inc.  
500 Old Connecticut Path  
Framingham, MA 01701

SUBJECT: Follow-up to SMC Letter Ballot 235, Approval of Appointment of  
International Representative, X3J3, Fortran

Dear Mr. Johnson:

I am pleased to inform you that the Secretariat Management Committee has appointed you as the International Representative. This appointment is effective immediately for a term of three years. The expiration date of your term will be May 20, 1990 and in February of 1990, we will issue a call for volunteers to the committee members so we can initiate a new letter ballot.

I certainly want to take this opportunity to extend my congratulations on your appointment and should you need any assistance from the Secretariat, please don't hesitate to let us know.

Sincerely,



Stanley Allen  
Manager, X3 Secretariat

cc: Patti Steiner, SD-6 input





BOEING COMPUTER SERVICES

13

P.O. Box 24346  
Seattle, Washington 98124-0346

A Division of The Boeing Company

June 8, 1987  
G-1581-HJQ259

Catherine A. Kachurik,  
ANSI X3 Administrative Secretary  
X3 Secretariat/CBEMA  
311 First Street, NW Suite 500  
Washington, DC 20001-2178

Subject: Fortran 8X Draft Standard

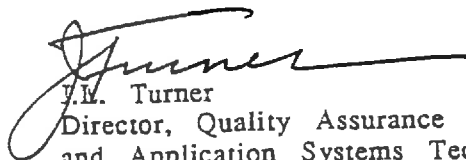
Dear Ms. Kachurik:

The Boeing Company has major concerns about the draft Fortran 8X standard recently passed by the X3J3 committee. If Fortran evolves according to the strategy proposed by X3J3, we feel the resulting adverse impacts on the Boeing software inventory will far outweigh advantages. Therefore, Boeing strongly urges the ANSI X3 committee to return the proposed standard to the X3J3 committee rather than releasing it for public comment. The following are major reasons for our concern:

- When obsolescent and deprecated features are deleted by future Fortran standard committees, we anticipate a major expense will be incurred to convert our extensive inventory of Fortran code.
- Casual Fortran users are expected to have considerable difficulty with a more complex language that has many redundant features.
- Major vendors have predicted poor performance for compilation and for the efficiency of generated code.
- We anticipate a long period of partial implementations when the language will not be uniformly available. Fortran portability will be severely impacted during this period.
- We anticipate a lack of compilers across our wide range of computers. Vendors have indicated that because of size and complexity this language would not be implemented on PCs.

If requested, we are prepared to submit a more detailed analysis. We need the benefits of modern programming languages. However, we feel the route proposed by the X3J3 committee will not be successful., Boeing is anxious to participate in defining guidelines to direct future standard development activities.

Sincerely,

  
 J.M. Turner  
 Director, Quality Assurance  
 and Application Systems Technology

cc: Jeanne C. Adams, X3J3 Chairman  
William C. Rinehuls, SPARC Chairman



105(X)JCA-14

CBE/MA 19

TO: Members, X3 Community  
(X3/TC, TG, and SG Officers -- Please retransmit to your members as soon as possible.)

FROM: Vico E. Henriques, President

SUBJECT: Transmittal of X3 and X3 Subgroup Service Fees for 1988-1989 Service Fee Period

As you will recall, the service fee period is established in two-year increments to allow you and your organizations some stability in budget planning. This year, budget development for the Secretariat has been a challenge with consideration given to those known factors, such as the increase in postage and unknown issues involved with JTC1 and X3's continuing evolution. The following decisions have been reached and are in effect for the next service fee period:

- 1) For X3, the service fees will remain the same. These figures, of course, reflect the 15% increase that X3 experienced last fee period. As a reminder, they are:

Producers	\$5750	
Government	5750	
Users, A	4600	(Users, Category A are "Fortune 500" Companies)
Organizations	3450	
Users, B	1150	(Users, Category B are below "Fortune 500")
Observers	1725	

The SMC recommendation to change Government fees to that equal of the Fortune 500 level was carefully considered. However, due to the rising costs and continuing need to spread the base as far as possible, it was felt that no fee could be lowered, especially in this year of raising Subgroup service fees.

- 2) Service fees for the Technical Committees will be raised as follows:

Current fees of \$175 for each Principal membership (which also includes one alternate), will be raised to \$200.

Current observer and second, third, fourth alternate fees of \$100 will be raised to \$150.

You will recall that at the beginning of last service fee period, we indicated that we might have to raise Subgroup fees in mid-period for 1987. Fortunately, we were able to prevent that. However, given the anticipated costs as reflected in the '88-'89 budgets, we can no longer hold the line at that level.

~~Transmittal of X3 and X3 Subgroup Service Fees for 1988-1989 Service Fee Period~~

211

- 3) Service fees for the members of the standing committees, SPARC and IAC, and Study Groups will be raised for the first time in history.

Current fees of \$50 for SPARC and IAC will be raised to \$100, with Study Group fees being raised to \$50.

- 4) In trying to address all facets of the next two years' work, should the JTC1 TAG be under the auspices of CBEMA, the fee charged to those not already X3 members would be \$500.

This number is firm for one year only. That is, it may be increased based on costs involved as experience dictates.

As always, a mechanism exists for consideration of requests for service fee waivers. Any organization or individual is free to apply for such a waiver immediately upon receipt of your service fee invoice.

I will be happy to discuss any comments you may have.

A handwritten signature in black ink, appearing to read "J. S. [unclear]". The signature is fluid and cursive, with a long horizontal stroke at the end.

1987-06-05

105 (\*) JCA-15

P. 1 of 4

15

Ms. Jeanne Adams  
Chairperson, ANSI X3J3  
NCAR - SCD  
PO Box 3000  
Boulder, CO 80307  
U. S. A

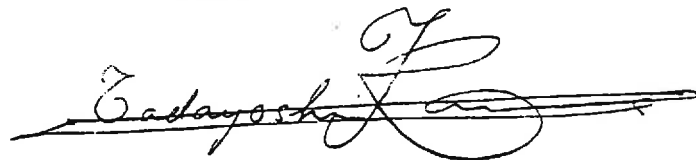
Dear Ms. Adams:

Thank you for your kind treatment for our experts who attended the 104th X3J3 meeting. I would like to make several experts attend the 105th (Liverpool) X3J3 meeting and make them explain Japanese proposal. Then I send you the followings:

- (1) Explanation of the Issues on Japanese Proposal discussed at the 104th X3J3 meeting, with Attachments
- (2) Attachment a): Note on the Extended FORTRAN77 which is capable to process Kanji data,
- (3) Attachment b): A Draft Proposal of JIS (Japanese Industrial Standard) Extended FORTRAN77 for processing Kanji data, (This document is described in Japanese, but an outline of the contents will be explained by our experts at the 105th (Liverpool) meeting),
- (4) Attachment c): Example of Japanese Character Code, JIS X0208,
- (5) Attachment d): Example of Chinese Character Code.

I expect the Liverpool meeting will be fruitful.

Sincerely yours



Tadayoshi Kan  
Chairman  
FORTRAN WG of Japanese  
National Committee for  
ISO/TC97/SC22

213

Explanation of the Issues on Japanese Proposal  
discussed at the 104 X3J3 meeting

1. Needs for various kinds of codes

(1) For natural languages :

We wish to manipulate not only Japanese Kanji ( see Attachment a), b), c) ), but also Chinese characters ( see Attachment d) ) and other hieroglyphic characters in oriental nations.

Remark : Although Japanese Kanji is based on the language of the old Chinese nation, Kan (漢) ; the current Chinese character is very different from Japanese Kanji.

(2) For scientific use :

We want to process wide-meaning and user-definable codes, like a certain binary code for complicated molecular structures of polycyclic hydrocarbons.

Such capability will promise a great evolution of Fortran.

2. Reason to be intrinsic, not derived

(1) Efficiency :

We wish to efficiently process codes for natural languages and wide-meaning codes described above, then the data type for such codes shall be intrinsic, not derived.

(2) Japan's experience of implementation :

In Japan, some main framers has already implemented the data type for Kanji and this feature has been supplied to users.

This implementation is successfully carried out as the intrinsic (if using 8X's concept) data type for FORTRAN 77.

### 3. Rationale for the form of modified CHARACTER Statement

(1) Syntax :

CHARACTER ( ( KIND = n ) ) [ length selector ]  
                  KIND part

If ( KIND = n ) is omitted or ( KIND = 1 ) is written, the statement specifies the current ( S8 ) character type.

Hereafter we denote [ ( KIND = n ) ] as the KIND part.

(2) Needs for the KIND part :

In order to treat various kinds of codes described in 1., we believe this form is the best, because

- i) it is consistent to the current CHARACTER statement,
- ii) the KIND part is adaptable for various kinds of codes.

Remark : Although the form of character statement for Kanji data type in JEIDA ( Japanese Electronic Industry Development Association ) Extended FORTRAN 77 is "NCHARACTER list", we don't support "NCHARACTER", because this keyword means "national" or "Nippon" and NCHARACTER statement can't declare more than one data types having multibyte code.

### 4. Portions to be modified are rather little

If Japanese proposal is adopted, several portions in the current 8X should be modified. However, such portions are rather little and Japan has the experience of modification for the Extended FORNTRAN 77.

We would like to participate in such work, if necessary.

Attachment a) :

Note on the Extended FORTRAN77 which  
is capable to process Kanji data

In Japan, some main framers have implemented FORTRAN77 with the feature which is capable to process Japanese Kanji data, and then the extended FORTRAN77 has been specified as JEIDA standard (not national standard). Then, a draft of JIS (national standard) for this FORTRAN was prepared in 1985 and now the draft of JIS Extended FORTRAN77 is pending for processing as JIS, because Japan hopes that the form of CHARACTER statement will become consistent to that of 8X.

Remark: In JEIDA Extended FORTRAN77 the form of character statement for Kanji is "NCHARACTER list", and this extension aims only to process Kanji.



Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

105 (X) JCA-16  
**NEWS RELEASE**

For more information contact:  
Ms. Elizabeth Rather  
Convener, X3J14  
(213) 372-8493

16

June 16, 1987

X3/IEEE SEEKING INTERESTED PARTIES TO WORK ON FORTH LANGUAGE

\* \* \* INTERESTED PARTIES URGED TO NOTIFY X3 OF INTENT TO PARTICIPATE \* \* \*

Washington, DC -- Accredited Standards Committee X3 on Information Processing Systems and the Computer Society of the IEEE, in joint cooperation announce the organizational meeting of the technical committee X3J14, Forth, to develop an American National Standard on Programming Language Forth. This Technical Committee will work to bring the various interests of the Forth Community together in one entity to further the standardization efforts.

There are two main dialects of Forth. One is based on Forth implementations marketed by FORTH, Inc. since 1973, and the other is based on a model distributed by the Forth Interest Group starting in 1978. Both dialects have converged under the influence of standards published by the Forth Standards Team, and at the application programmer level, are very similar. This similarity should serve as the foundation upon which ANS Forth can be built.

Forth is an extensible, interactive programming language being used increasingly in a wide variety of areas, especially machine control and artificial intelligence.

Because of the expanding use and increasing number of dialects, there is a need for a single clearly-defined standard. An American National Standard for Forth

217

will enable production of educational documents and manuals usable with any standard implementation.

Since X3J14 intends to complete the draft American National Standard by December 31, 1987, we are urging interested participants and users to get involved as soon as possible. X3J14 first meeting will be August 3-4, 1987, CBEMA Headquarters, Washington, DC. Those who are interested are invited to join in this project. Please contact:

Ms. Elizabeth Rather  
FORTH, Inc.  
Forth Standards Team  
111 N. Sepulveda Blvd.  
Suite 300  
Manhattan Beach, CA 90266  
(213) 372-8493

ISO/TC 97/21 N  
Date: 1987-05-26

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION  
ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/TC 97  
INFORMATION PROCESSING SYSTEMS  
Secretariat: USA (ANSI)

Ref.: X3J87-05-198  
Follow-up: X3J87-04-022

INFORMATION TO BE PROVIDED BY THE SUBMITTER:

TITLE: Summary of Voting on Resolution 9 of Document ISO/TC 97 N 1869, on the transfer of responsibility for the work on Computer Graphics from SC 21 to a new Subcommittee (24)

SOURCE: TC 97 Secretariat

PROJECT:

INFORMATION TO BE PROVIDED BY THE SECRETARIAT:

STATUS: The formation of SC 24 has received the substantial support of the 'P' members of TC 97. The effective date of the transformation from SC 21/WG 2 to SC 24 will be 1 October 1987.

REQUESTED ACTION: Member Bodies (P, O and I) of both TC 97 and SC 21 are requested to notify the TC 97 secretariat no later than 15 September 1987 if you wish to be registered as a member (P, O, I) of SC 24.

DISTRIBUTION: P, O, I Members (TC 97 and SC 21)  
Secretariat, ISO/TC 97/SC 21

Summary of Voting on Resolution 9 of Document TC 97 N 1869, on the transfer of the responsibility for the work on Computer Graphics from SC 21 to a new Subcommittee (SC 24)

105 (X) J-9-17

TC 97 'P' Members	Approve	Approve with Comments	Disapprove	Abstain	Comments
Austria	X				
Belgium	X				
Canada				X	
China	X				
Czechoslovakia	X				
Denmark	X				
Finland					
France	X				
Germany	X				
Hungary	X				
Iran	X				
Italy			X		see reverse
Japan	X				
Korea, Republic of					
Netherlands	X				
Norway	X				
Switzerland	X				
Turkey					
United Kingdom			X		see ATT 1
USA	X				
USSR	X				

17

Comments accompanying letter ballots.

Italy

ISO/TC 97/SC21/WG 2 has just begun to approach the model's themes from the OSI point of view. It seems to us that this approach has to be further pursued. The transfer of responsibility to a new Subcommittee may result in a sharp modification of this approach.

Action by Secretariat

The letter ballot on the formation of a new subcommittee on Computer Graphics has received the substantial support of the 'p' members of ISO/TC 97. Therefore, the TC 97 Secretariat will contact the members (P,O and L) of both TC 97 and SC 21 to determine which of them wish to be registered as members of SC 24. The effective date of the transformation from ISO/TC97/SC21/WG 2 to ISO/TC97/SC 21 will be 1 October 1987.

The concerns raised in the ballot comments will need to be addressed at the first Plenary meeting of SC 24 in order to ensure that effective coordination and liaison with the work on OSI and across the Systems Grouping (including SC 22) is maintained.

Further, the TC 97 Secretariat has had discussions with the Convener of SC 21/WG 2 to ensure that the work on graphics will not be delayed or disrupted due to the formation of SC 24. All available output documents from the SC21/WG 2 meeting held in Valbonne, 18-27 May 1987, will be forwarded to the SC 21 Secretariat for action where appropriate.



Att. I to  
ISO/TC 97 N 1922  
1987-04-06

UK COMMENT ACCOMPANYING NOTE OF DISAPPROVAL ON BALLOT ACCOMPANYING ISO/TC 97 N 1869, FORMATION OF A NEW SUBCOMMITTEE ON COMPUTER GRAPHICS

1. The UK acknowledges the importance of Computer Graphics standardization in its own right.

In particular, the UK accepts that the subject of Computer Graphics is sufficiently large that a viable Subcommittee could be formed with Computer Graphics as its only subject.

2. However, the UK disapproves the formation of a Subcommittee on Computer Graphics at this time.

The UK recommends that any rearrangement of the workload of ISO/TC 97/SC 21 should be done only in the context of a further top-down review of all the work for which ISO/TC 97 is currently responsible.

The UK believes that there is a need to recognize that the standardization activities across the Systems Grouping (with SC 22) are closely interrelated and need to be seen, planned and managed together. Further fragmentation of the activity among independent subcommittees without adequate coordination mechanisms will make the development of coherent standards across the Systems Grouping very difficult, if not impossible.

3. The UK notes that effective liaison from ISO/TC 97/SC 21/WG 2 is starting to bear fruit. The 1984 reorganization of ISO/TC 97 was designed (inter alia) to improve liaison; the UK believes that there is a danger of these liaisons being disrupted by further reorganization.

4. In particular, the progression of work in the area of Open Distributed Processing depends on very close liaison in all the existing areas of activity within ISO/TC 97/SC 21, including the present ISO/TC 97/SC 21/WG 2, Computer Graphics.

5. The UK considers that the formation of a new Subcommittee for Computer Graphics could have the same disruptive effect on the progression of technical work as was experienced during the reorganization of ISO/TC 97 during 1984 and culminating with the formation of ISO/TC 97/SC 21 in 1985.

6. The UK further considers that the administrative load placed by the very existence of a Subcommittee on Computer Graphics on experts in the field would further disrupt the progression of work. The UK considers that such administrative load may be expected to affect experts at the national as well as at the international levels.

220

18

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Scientific Computing Division/Advanced Methods Section

P. O. Box 3000 • Boulder, Colorado • 80307

Telephone: (303) 497-1275 • FTS: 320-1275 • Telex: 45 694

June 25, 1987

Don Deutsch, X3H2  
General Electric Company  
Geisco  
278 Franklin Road, Suite 300  
Brentwood, TN 37027

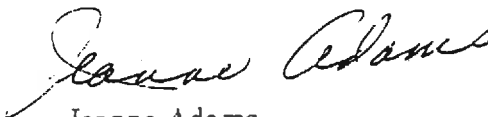
Dear Don;

Enclosed you will find the X3J3 ballots on the draft proposed Database Language Embedded SQL. The ballot closed on June 20. There were some NO votes and some comments. There are 38 members of X3J3. 20 is one more than half the membership, so the vote qualifies with 22 voting. I feel that I will receive more votes in the near future, and will send these on to you. Treat these as the initial tally only. Many people are on vacation at this time, and we should wait for more ballots.

Yes-19  
No-3  
Abstain-4  
Not Voting-12

If I can help in any other way, please let me know. I expected to hear from Murray Freeman, since he promised to prepare a paper and ballot on this. I will mail it to you when it comes. Murray as you know has agreed to continue as liaison to your committee. He has promised to present a paper at the fall meeting in Ft. Lauderdale. Of course, you and any others are also most welcome.

Regards,



Jeanne Adams  
Advanced Methods Group, NCAR  
Chair, X3J3

cc. X3J3 Pre-meeting Distribution

221

105 (\*) JCA-18  
p. 2 of 18

X3J3 BALLOT RESULTS ON SQL  
PRELIMINARY RESULTS, JUNE 25, 1987

Adamsyzk, J. S.	No Vote Cast
Adams, Jeanne	Yes
Allison, Robert	Yes
Bowe, Valerie	Yes
Brainerd, Walt	No Vote Cast
Burch, Carl	Yes
Campbell, Lloyd	Yes
Crowley, Ted	No Vote Cast
Ellis, T. M. R.	Yes
Freeman, Murray	No Vote Cast
Gridley, Curt	Yes
Harris, K	No Vote Cast
Hendrickson, Richard	Yes
Hirchert, Kurt	No
Hoover, Tracy	Abstain
Johnson, Andy	No Vote Cast
Lakhwara, Anil	No Vote Cast
Marshall, Neldon	Abstain
Martin, Bruce	No Vote Cast
Martin, Jeanne	Yes
Marusak, Alex	Yes
Matheny, James	Yes
Metcalf, Mike	No Vote Cast
Millard, G. E.	Yes
Moss, Leonard	Yes
Phillips, Ivor	No Vote Cast
Phillimore, David	Yes
Ragan, Richard	No
Reid, John	No
Schenk, Werner	Yes
Schonfelder, J. L.	Yes
Smith, Brian	Abstain
Swift, R. C.	No Vote Cast
Thompson, Brian	Yes
Wagener, Jerry	Yes
Wearing, Alison	Abstain
Weaver, R.	Yes
Wilson, Alan	No Vote Cast

222

# X3 Subgroup Letter Ballot

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

## FOR ACTION

Ref. Document: X3H2-87-32  
X3H2-87-79

**Statement:**

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

Yes

No, with reasons

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Valerie G. Bowe  
(PLEASE PRINT)

SIGNATURE Valerie G. Bowe

DATE: 6/17/87

*\*Operating under the procedures of The American National Standards Institute.*

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8886  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

223

Unisys votes yes on the X3H2 document, Database Language Embedded SQL, with one reservation.

We are concerned by the statement on page 24 of the document that "Blanks are significant in <embedded SQL statement>s" of <embedded SQL FORTRAN program>s. This is a major inconsistency within the FORTRAN language. A FORTRAN programmer, who uses the insignificant blanks feature of FORTRAN, would find it more than a little troublesome to remember that blanks are now sometimes significant. Also, FORTRAN implementors will now have the burden of writing processors which must deal with both insignificant and significant blanks.



# X3 Subgroup Letter Ballot

~~20~~  
22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

# FOR ACTION

Ref. Document: X3H2-87-32  
X3H2-87-79

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

       Yes

No, with reasons (SEE attachment)

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Kurt W. Hinchert  
(PLEASE PRINT)  
SIGNATURE Kurt W. Hinchert  
DATE: 87/6/18

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

025

For the most part, my review of this document was limited to those sections pertaining to Fortran, as I do not have the expertise to adequately review the sections pertaining to the other languages.

My negative vote stems from the fact that the description in this document is insufficiently precise to serve as a specification for a product. In particular, I found the following "holes":

- It is unclear whether an embedded SQL statement is to be allowed as the object of a logical IF statement. If the translation of the SQL statement would be multiple Fortran statements, it would also be necessary to convert the logical IF into a block IF. Also recognizing an embedded SQL statement, in which blanks are significant, as the terminal part of a Fortran statement, in which blanks are not significant, may be too much of a burden on the translation processor.
- I believe it is necessary to specify that an embedded statement not be the terminal statement of a DO loop. If the translation of an embedded SQL statement would be multiple statements, there is no placement of the label on the translation that would be correct both for transfers of control and termination of the loop.
- Fortran does not have a concept of declaring a variable. Instead, names are specified to have particular attributes, such as type. The attribute of being a (scalar) variable is deduced from the absence of specifications or usage contrary to that hypothesis. Thus, it is necessary to specify what specifications or usage of a name are inconsistent with its being specified in an embedded SQL declaration. For example, the document should probably prohibit appearance in other type specifications, appearance as an array declarator (in a DIMENSION or COMMON statement), appearance in an EXTERNAL statement, appearance in a PARAMETER statement, definition as a statement function, and reference as a function. On the other hand, it could permit appearance in a COMMON statement, appearance in a SAVE statement (including the SAVE of a common block in which it appears), appearance in a DATA statement, and appearance in the dummy argument list of the host program unit.
- The document uses the term "compilation unit". This term is not defined here or in the Fortran standard and is ambiguous in current usage. Do you mean the minimum unit that can be separately compiled (i.e., a program unit) or that part of a program which is compiled at one time (possibly several program units)? This question has relevance in determining the effective range of an embedded exception declaration.
- The statements concerning which embedded SQL statements can have statement numbers make no sense. Note that in the Fortran standard all statements can have statement numbers. However, the statement numbers on some statements may not be referenced.

In addition to the above questions, I have a number of concerns that would not be sufficient to cause me to vote negatively on this document, but which I would like to see addressed:

- I object to the procedures that lead to X3J3 members being forced to vote as individuals on this document. At no time were we ever asked whether we, as individuals, were willing to accept this burden. Moreover, the question we have been asked is unreasonable. Few, if any, of us have the expertise to vote on this

document as a whole. The best we can be expected to do is review its implications on Fortran.

- The editorial approach in this document is poorly suited for its intended audience. Most users of embedded SQL and implementors of the translators for embedded SQL are likely to be interested only in a single host language. This intermingling of rules for 6 different languages only serves to confuse things. Ideally, the rules for the different languages should be described in separate documents or at least in separate sections of this document. As an aid to those few people who are interested in multiple host languages, identical language should be used to describe concepts and rules common to all of the languages and a mechanism such as revision bars should be used to distinguish text specific to a particular language from text common to all languages.

At the very least, syntax which is different in the different host language should be provided separately for each language in question rather than providing a description of the union of all such syntaxes!

- I have misgivings about the use of the word EXEC as the initial keyword on all embedded SQL statements. This is preempting a "useful" keyword and may eventually interfere with extension of the host language standards or extension of this standard to a host language already using the keyword (or reserved word) EXEC. In addition, EXEC is a misleading keyword when applied to declarations. I suggest the SQL EXEC would be preferable to EXEC SQL for executable statements and the SQL BEGIN DECLARE SECTION would be preferable to EXEC SQL BEGIN DECLARE SECTION for declarations. Similarly, SQL WHENEVER may be preferable to EXEC SQL WHENEVER.
- Although it may be implicit in your description of equivalent modules and host language programs, I would appreciate an explicit statement that embedded SQL declare sections may appear only where the equivalent host language declarations may appear and that executable embedded SQL statements may appear only where executable host language statements may appear.
- Failure to use appropriate host language terminology and syntax can be misleading. For example, the syntax for a Fortran statement label (not statement number) is similar to, but not identical to, the syntax for an unsigned integer constant, so that an embedded exception declaration really ought to contain the former rather than the latter.
- In clause 8, case d, item 3, reference is made to a data type CHARACTER(L). Note that Fortran has only CHARACTER\*(L) or CHARACTER\*l (where l is an unsigned integer constant).
- The statement that any valid Fortran name is a <FORTRAN host identifier> could be misinterpreted. Note that since blanks are not significant in Fortran, a name such as N CASES is valid in Fortran. I assume that a name with embedded blanks would not be valid in SQL.
- It is unfortunate that this standard does not support Fortran arrays or the Fortran types LOGICAL and COMPLEX.
- The allowed options on embedded exception declarations are inadequate. In Fortran, I can imagine wishing to use an assigned GOTO or computed GOTO. In

Ada, I would strongly desire the ability to use the language's exception handling mechanisms. It might be desirable to add a third form that would bracket arbitrary host language statements as means of providing a general mechanism.

- Although the current Fortran standard allows only 6 character variable names, there is no immediate ambiguity, but if longer variables names are permitted (as in the proposed revision of the Fortran standard and many implementations of the existing standard) the possibility exists of ordinary Fortran statements beginning with EXEC SQL. (Since we were not provided with the syntax of SQL statements, I can't discount the possibility that there may be statements that would be both valid embedded SQL statements and valid Fortran statements.) It seems incredibly short-sighted not to address this issue.
- I hope that every C reviewer has pointed out the error in saying that SQL REAL should correspond to C double. Since you say elsewhere that SQL REAL corresponds to C float, it is unlikely that anyone would be misled, but an error as serious as this really ought to be fixed before this document is released to the public.

# X3 Subgroup Letter Ballot

~~20~~

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

Ref. Document: X3H2-87-32  
X3H2-87-79

# FOR ACTION

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

       Yes

       No, with reasons

*I wish to be recorded as NOT VOTING because I don't feel I am qualified to submit a ballot of 'yes' or 'no'.*

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Tracy Hoover  
(PLEASE PRINT)

SIGNATURE Tracy Ann Hoover

DATE: 22 June 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

229

# X3 Subgroup Letter Ballot

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

Ref. Document: X3H2-87-32  
X3H2-87-79

# FOR ACTION

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

       Yes

       No, with reasons

*NOT VOTING* Although I am in general in favor of this standard, I do not feel qualified to evaluate it sufficiently for an intelligent vote.

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Neldon H. Marshall  
(PLEASE PRINT)

SIGNATURE Neldon H. Marshall

DATE: May 14, 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

730

# X3 Subgroup Letter Ballot

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

## FOR ACTION

Ref. Document: X3H2-87-32  
X3H2-87-79

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

Yes  
 No, with reasons

*Q. What is the reason for the limited # of data types?*

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME DAVID PHILLIMORE  
(PLEASE PRINT)  
SIGNATURE David Phillimore  
DATE: 12th June 1987

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

# X3 Subgroup Letter Ballot

~~20~~

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

Ref. Document: X3H2-87-32  
X3H2-87-79

# FOR ACTION

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

Yes

No, with reasons

- 1) END-EXEC is not consistent with FORTRAN keywords (hyphen not allowed).
- 2) Blanks are not significant in FORTRAN but are significant in embedded SQL. This will confuse FORTRAN users.

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Richard Ragan  
(PLEASE PRINT)

SIGNATURE Richard R. Ragan

DATE: 6/12/87

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

232



# X3 Subgroup Letter Ballot

(22)

22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

Ballot Closes: NOON

X3J3 is coordinating liaison to X3H2

Ref. Document: X3H2-87-32  
X3H2-87-79

# FOR ACTION

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

       Yes

  X   No, with reasons *attached*

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME John Reid  
(PLEASE PRINT)

SIGNATURE *John Reid*

DATE: 11 June 1987.

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178
---

Tel: 202/737-8888 Fax: 202/638-4922
--

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

An embedded SQL FORTRAN program consists of a mixture of Fortran statements and embedded SQL statements, which I take to include <embedded SQL begin declare> and <embedded SQL end declare>. It is essential that the SQL processor can distinguish between Fortran statements and embedded SQL statements. This is achieved by starting each embedded SQL statement with EXEC SQL. This is adequate for Fortran 77 because no keyword commences like this and identifiers may have at most 6 characters.

The reason for my NO vote is that I cannot be sure that this will be adequate for Fortran 8x. The current draft maintains the interpretation of blanks as insignificant and allows identifiers of up to 31 characters. The simplest fix is to state in rule 1 on page 24 that if a statement can be interpreted as either a Fortran statement or an embedded SQL statement, it is interpreted as an embedded SQL statement.

I also have the following comments:

1. In the last sentence of page 4, you need to state that each <embedded SQL begin declare> and each <embedded SQL end declare> is removed.
2. On page 15, line 4, replace 'this document' by 'Database Language SQL'.
3. On page 15, line 11, replace 'program' by 'program unit'.
4. On page 24, line 6, replace 'Syntax Rules' by 'Syntax Rules of this Section'.
5. On page 24, syntax rule 2, replace each 'statement number' by 'statement label', which is the correct Fortran term.
6. Page 24, syntax rule 2: The BNF does not appear to allow for an <embedded SQL statement> to have a label.
7. On page 25, line 1, replace 'a host variable' by 'one or more host variables'.

# X3 Subgroup Letter Ballot

~~20~~  
22

Accredited Standards Committee  
X3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

## FOR ACTION

Ref. Document: X3H2-87-32  
X3H2-87-79

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

       Yes

       No, with reasons

  ✓   NOT VOTING, see ~~attached page~~ reverse side

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME Brian T. Smith  
(PLEASE PRINT)  
SIGNATURE Brian T. Smith  
DATE: 5/13/87

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

X3 Secretariat: Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001-2178

Tel: 202/737-8888  
Fax: 202/638-4922

American National Standards are developed by the voluntary participation of all parties and with the intention and expectation that the standards will be suitable for wide application. Since their use is likewise voluntary, an affirmative vote does not commit an organization or a group represented on the committee to the use of the American National Standard under consideration.

Comment: I am ~~not~~ unqualified to comment on this draft standard X3H2-87-79, Database Language Embedded SQL, March ~~1987~~ 1987, and do not have the time to become sufficiently knowledgeable of the Database SQL [reference 6]. My vote of 'Yes' or 'No' would be ~~completely~~ misleading, and therefore I refuse to vote.

I have however ~~noted 2~~ <sup>two</sup> ~~2~~ comments to make:

1. Page 16, line 3, change 'when' to ~~when~~ 'when'.
2. Page 24, points 3 and 4 are contradictory.

A Fortran variable name is allowed to be X Y where there is a blank between X and Y (as blanks are insignificant in X 3.9-1978) yet according to point 3, blanks are significant in SQL so X Y is not the variable XY. How do you resolve this contradiction?

Don E Smith  
5/14/87

# X3 Subgroup Letter Ballot



22

Accredited Standards Committee  
3, Information Processing Systems\*

Doc. No. X3J3-87-104/JCA-15

Date: May 1, 1987

Project:

Ballot Period: May 10, - June 20, 1987

Subject: Database Language Embedded SQL

X3J3 is coordinating liaison to X3H2

Ballot Closes: NOON

## FOR ACTION

Ref. Document: X3H2-87-32  
X3H2-87-79

Statement:

X3H2 has requested that X3J3 ballot on the above referenced document as coordinating liaison to X3H2.

Question:

Do you approve X3H2-87-79 (draft proposed) Database Language Embedded SQL, March 1987, for submission to X3 for further processing as an American National Standard?

Yes

No, with reasons

NOT VOTING with reasons (see attached)

Mail to: Jeanne Adams  
NCAR/SCD  
PO Box 3000  
Boulder, CO 80307-3000

NAME ALISON WEARING  
(PLEASE PRINT)

SIGNATURE Alison Wearing

DATE: 9-6-87

\*Operating under the procedures of The American National Standards Institute.

NOTE: If you find that you cannot vote YES and wish to be recorded as NOT VOTING or voting NO, please state this and explain the reasons for your position in the space above or on a separate sheet.

Reasons For Not Voting

I joined the ANSI X3J3 committee at the May meeting, the meeting at which the Letter Ballot was distributed. At this meeting, and its previous one which I attended as an observer the member who acts as the liaison officer between X3J3 and X3H2 was not present. I therefore consider that I cannot cast a vote on the limited information available to me.

I trust that this will not count as part of my permitted Not Voting allowance due to the exceptional circumstances.

*Alison Wearing*

Alison Wearing

9th June 1987

From : Robert Allison and Carl Burch  
To : X3J3  
Subj : Remove the EXPONENT LETTER Statement

**Background :**

At the Mount Kisco Implementors' Forum, both Stu Feldman and Frank Leahy commented derisively on the EXPONENT LETTER statement. Their comments were roughly (very roughly) that it was ridiculous to use a separate statement for such a little dab of functionality. Carl Burch tried once before to delete the statement (101(\*) CDB-4) but failed to consider the requirement for specifiable precision in DATA statements and other lists of constants. We now believe that the exponent letter definition itself is required, but a dedicated statement for that purpose is still excessive.

**Proposal 1.**

Delete the EXPONENT LETTER statement in favor of including it in the REAL statement's *entity-decl-list*. In other words, if a single-character alphabetic literal appears in the list of entities being declared, it is an exponent letter signifying the precision and exponent range attributes specified.

**Proposal 2.**

Delete the EXPONENT LETTER statement in favor of including it in the REAL statement's *precision-selector*. That is, if a single-character alphabetic literal appears as the last argument of the *precision-selector*, possibly preceded by "EXPONENT\_CHAR =", it is an exponent letter signifying the precision and exponent range attributes specified.

The following text changes are from X3J3/S8.104 (June 1987).

**Text Changes to Remove the EXPONENT LETTER statement :**

- Page 2-2 :  
Delete or *exponent-letter-stmt* from R221.
- Page 4-4 :  
Delete R409 (lines 3-4).  
Replace "EXPONENT LETTER" with "REAL" in lines 7-8.  
Replace "an EXPONENT LETTER" with "a REAL" in lines 11-12 and 22-23.  
Replace "The EXPONENT LETTER statement" with "*a defined-exponent-letter*" in line 24.  
Replace line 27 with the new syntax passed from Proposal 1 or 2.
- Page C-2 :  
Replace "EXPONENT LETTER" with "REAL" in line 21.

**Text Changes for Proposal 1 :**

- Page 4-4 :  
Replace line 27 with :  
REAL (10, 50) 'L'

Page 5-1 :

Add to R506 after line 40 (to *entity-decl*) :  
or *'defined-exponent-letter'*  
or *"defined-exponent-letter"*

Add as a constraint to R506 :

Constraint : A *defined-exponent-letter* may appear only if the *type-spec* is REAL and a *precision-selector* also appears.

Page 5-3 :

Add paragraph after line 22 : If a *defined-exponent-letter* appears in the *entity-decl-list*, it specifies that a *real-literal-constant* in which the *defined-exponent-letter* appears will have the precision and exponent range attributes specified in the *precision-selector*.

Note : the current /S8 text never actually establishes the linkage attempted in the sentence above.

Page 5-4 :

Add after line 3 :

REAL (16, 300) 'L'

End Text Changes for Proposal 1.

Text Changes for Proposal 2 :

Page 4-4 :

Replace line 27 with :

REAL (10, 50, EXPONENT\_LETTER='L')

Page 5-2 :

Change R507, lines 39-44 to:

*precision-selector* is ( *selector-value-list* )  
*selector-value* is [ PRECISION= ] *type-param-value*  
or [ EXPONENT\_RANGE= ] *type-param-value*  
or [ EXPONENT\_LETTER= ] *char-literal-constant*

Constraint: A *selector-value-list* may contain at most one of each of the parameters.

Constraint: If the optional characters PRECISION= are omitted from the precision type parameter, the precision type parameter must be the first item in the *selector-value-list*.

Constraint: If the optional characters EXPONENT\_RANGE= are omitted from the exponent range type parameter, the exponent range type parameter must be the second item in the *selector-value-list* and the first item must be the precision type parameter without the optional characters PRECISION=.

Constraint: If the optional characters EXPONENT\_LETTER= are omitted from the exponent letter type parameter, the exponent letter type parameter must be the third item in the *selector-value-list* and the first two items must appear without the optional characters PRECISION= and EXPONENT\_RANGE=.

Constraint: The *char-literal-constant* in the exponent letter type parameter must be one character in length, and must be an alphabetic character, excluding "D", "E", and "H". If a processor is capable of representing characters in both upper and lower case, the value specified is without regard for case.

End Text Changes for Proposal 2.



105(\*) CDB-2  
July 1, 1987

From : Carl Burch

To : X3J3

Subj : Some Notes on the Implementation of an Intrinsic BIT Module

At the March meeting I was so bold as to oppose Jeanne Adam's statement that BIT data type could be provided by the Module facility. I was pontificating on the principle that abstract data types (as implemented by a Module) serve admirably to provide data types and operations above the level of those provided intrinsically, but not to simulate more fundamental types. I was immediately demolished by Dick Hendrickson, who pointed out that I had missed the key word : intrinsic. Page 1-5 of S8/104 defines an intrinsic module as "a module definition included with this standard." This means two things to me :

- (1) It is part of the standard and must be implemented on all standard-conforming processors, and
- (2) The compiler writers can cheat.

By "cheat", I mean that we can introduce knowledge of the intrinsic module into the compiler, in somewhat the same way we do generic intrinsic functions in Fortran 77. I call it cheating because generics are not codeable in Fortran 77, but we provide them in the language anyway. A more blatant example is the generation of inline code for simple functions like ABS - the user's source code includes the syntax of a function call, but no call is actually done at run time. In general, once something is made intrinsic to the language, the compiler writers are free to "do what I mean" up to the limits specified by the language.

What does all this have to do with BIT data type? It opens the opportunity for implementation of a standard BIT manipulation facility, without the excess baggage involved with a fully general data type. For example, I never heard anyone say there was any use for formatted (or list-directed, or NAMELIST) I/O of BIT items. Even though its utility is questionable, it's perfectly clear that all intrinsic data types must be allowed for all kinds of I/O. Hence, the users would all have to pay for the implementation of the B edit descriptor, have their programs slowed by the paging overhead (on virtual memory systems) of the library code to interpret it, and wait longer for delivery of their vendor's F8x compiler, all for a capability that even avid proponents of a bit facility agree is only proposed for "orthogonality."

I believe that all such expensive and undesired side-effects can be avoided in the specification of an intrinsic module. Designing such a module would also serve to focus the debate between proponents of a BIT-string design and the current Appendix F (array-oriented) design.

Notes on a possible implementation of intrinsic module ANSI-BITS (based on the current Appendix F design) :

- (1) The statement USE ANSI-BITS would not invoke the general USE semantics. It would be interpreted as if it were a (gasp!) compiler directive enabling the use of the module's definitions and interfaces, as "intrinsic" to the compiler as the ABS function.
- (2) BIT data type itself would be "declared" as a derived data type with one private component, which is allowed only two values. Other than having a declaration for formal purposes, the compiler ignores it and actually uses real bits.
- (3) On machines like the Cyber 205 which use BIT vectors for masking, the compiler would recognize statements like

WHERE ( LBIT (bvector))

and use the bit vector directly (i.e., ignore the LBIT function call).

- (4) The other functions (MAXBITS, LBIT, BITL, IBITLR, BITLR, IBITRL, BITRL) and operators (.BNOT., .BAND., .BOR., .BXOR.) could be easily implemented as inline code. In fact, they would be very similar to the MIL-STD 1753 functions already implemented by many processors.
- (5) Unformatted I/O of bit vectors would still end up being somewhat tricky, but since it would be intrinsic it could use the I/O library directly. Again, "cheating" helps to do things not allowed in standard Fortran (even 8x). Actually, all that would be necessary would be to allow unlimited numbers of arguments such as are allowed in the F77 MAX function. This would be difficult to describe in a Module definition, however.

Another alternative would be to use a bit-stream model and have an interface that allowed one bit vector per call. Frankly, I don't know enough about the requirements in this area to even suggest which approach would be more valuable, Fortran's traditional record-oriented model or the bit-stream model. Note, however, that for either model (with the exception of unlimited argument lists) the intrinsic module approach allows both simple description in the standard and efficient implementation. "Simple description" in that I see nothing in the concept of an intrinsic module implying that more than the interface need be specified in the Standard - the implementation can be left as an exercise to the student, er, compiler writer.

From : Carl Burch

To : X3J3

Subj : Some Pseudorandom Thoughts on a Pointer Facility

First of all, I'd like to extend my apologies to all those that have paid far more dues on this problem than I. Though I have some misgivings about pointers for Fortran, I'd like to offer some ideas in the hope that they are not all completely naive. An earlier version of this paper was on the table in Seattle, where it received a few comments, most notably from Kurt Hirchert - to whom most of the improvements are due.

My experience using pointers in C and Pascal leads me to believe that they are used for three primary functions :

- (1) As a base for dynamic allocation, for indefinite-sized data structures like linked lists.
- (2) Dynamic aliasing, for performance in algorithms like sorts that swap pointers to large structures instead of moving the data records themselves.
- (3) building recursive data structures such as binary trees.

I've often heard it said at FIDS that we already have the largest part of a pointer facility - ALLOCATE for dynamic allocation, IDENTIFY for dynamic aliasing, recursive procedures for the manipulation of recursive structures. I suspect that this is true. With ALLOCATE, we already have introduced the dangling pointer problem into Fortran 8x - either directly (using an unallocated or deallocated array) or indirectly (by IDENTIFYing an alias array onto an allocatable array, then deallocating it). Limiting IDENTIFY as we have only mitigates the danger to the casual programmer somewhat. I'm reminded of the thief that opined he'd as rather be hung for a sheep as a lamb.

My own doubts about pointers in Fortran date back to my difficulties in learning Pascal. At the time, I was a reasonably experienced Fortran 66 programmer, but had only handled "pointers" in assembly language. I fell in the classic trap of confusing the pointer with the object to which it points (this memo has dangling pointers, so I'm trying to avoid dangling participles - or is it prepositions that dangle?), not to mention forgetting to check for NIL and being confused by objects that have no names (in and of themselves).

I can't say as I've done any of those things for some years now, but I still tend to divide the process of learning to program into three plateaus :

- (1) Standard iterative programming (e.g., Fortran 77).
- (2) Pointers, dynamic allocation and limited use of recursion (Pascal and C).
- (3) Routine recursion (LISP).

I omit newer disciplines such as object-oriented programming due to my own ignorance and their relative unproven status in production environments.

A lot of the problem with pointers is that the syntax for assigning the pointer was very similar to the syntax for assigning the pointer's object - they were both assignment statements, and the user is responsible for providing the ^ or -> that makes the difference. Perhaps the greater syntactic distance between the assignment statement and ALLOCATE/IDENTIFY will serve to clarify the difference in the minds of Fortran 8x programmers making the leap from level one to two.

Another problem was that pointers served to mix two fairly different concepts (dynamic allocation and aliasing) into one overly powerful and dangerous implementation. This is to say nothing of such further uses as LRLTRAN's pointers into code space and the like (probably the most dangerous form of ASSIGNED GOTO ever invented). Separating the two concepts into two syntactic classes seems promising - splitting both allocation from aliasing in ALLOCATE and IDENTIFY,

and pointer assignment (in ALLOCATE/IDENTIFY) from dereferencing (any other use of the name, and most especially assignment statements).

It seems to me that /S8.104 need be changed very little to give all the three capabilities listed above, without an explicit pointer data type. I'd like to suggest the consideration of a hidden pointer facility, keeping the greater syntactic safety of the current design but adding the additional capabilities listed above that /S8 currently lacks. I hold these to be :

- the capability to define recursive data structures.
- permitting data objects to have both the ALLOCATABLE and ALIAS attributes, or define ALIAS to also imply ALLOCATABLE.
- allow subsequent IDENTIFY statements to associate a given alias object with multiple hosts.
- the ability to allocate scalars.

As I recall, WG5 asked for the capability to allocate scalar objects last summer, but lost out in the wake of the Compromise. Of course, their real objective was to get a leg up on a pointer facility - maybe the time is now to revisit this question and settle it one way or the other.

This proposal represents a relatively "safe" pointer facility, safer even than Pascal in that "pointers" clearly must be unallocated (Pascal NIL) when created, eliminating at least the uninitialized-pointer class of problems. It is also safe as pointer arithmetic is not allowed - you can't add 7 to a pointer, which promotes portability in that you don't have to answer the question "7 what?".

A binary tree insertion module fragment would look something like this (please excuse all errors - we need to build a syntax checker for the examples) :

```
TYPE NODE
```

```
  INTEGER KEY ! data this node contains, especially the key value.
```

```
  TYPE (NODE), ALLOCATABLE, ALIAS :: L_CHILD, R_CHILD
```

```
    ! would have to have ALIAS attribute to rebalance tree.
```

```
END TYPE NODE
```

```
TYPE (NODE), ALLOCATABLE, ALIAS, SAVE :: ROOT
```

```
SUBROUTINE INSERT (ROOT, VALUE, EXISTED)
```

```
  ! <<< PARAMETERS >>>
```

```
  TYPE (NODE), ALLOCATABLE, ALIAS :: ROOT ! root of (sub)tree to search
```

```
  INTEGER VALUE ! key to match on
```

```
  LOGICAL EXISTED ! was item found in tree?
```

```
  ! <<< LOCAL VARIABLES >>>
```

```
  TYPE (NODE), ALIAS :: P ! current node - note that ALLOCATABLE is
```

```
    ! not necessary. Query - is it still
```

```
    ! type-compatible with ROOT?
```

```
  ! <<< BEGIN EXECUTABLE CODE >>>
```

244

```
IF (.NOT. ALLOCATED (ROOT) ) THEN ! subtree is empty, insert at root.
  ALLOCATE (ROOT)
  ROOT%KEY = VALUE
  EXISTED = .FALSE.
  RETURN
ENDIF

IDENTIFY (P = ROOT)

DO ! iterate down tree

  IF (P%KEY == VALUE) THEN ! found in tree
    EXISTED = .TRUE.
    RETURN
  ENDIF

  IF (P%KEY < VALUE) THEN ! try left child

    IF ( ALLOCATED (P%L_CHILD) ) THEN
      IDENTIFY (P = L_CHILD)
    ELSE ! not allocated, insert it here
      ALLOCATE (P%L_CHILD)
      P%L_CHILD%KEY = VALUE
      EXISTED = .FALSE.
      RETURN
    ENDIF ! try of left child

  ELSE ! key > value

    IF ( ALLOCATED (P%R_CHILD) ) THEN
      IDENTIFY (P = R_CHILD)
    ELSE ! not allocated, insert it here
      ALLOCATE (P%R_CHILD)
      P%R_CHILD%KEY = VALUE
      EXISTED = .FALSE.
      RETURN
    ENDIF ! try of right child

  ENDIF ! key > value

END DO

END SUBROUTINE INSERT
```

246

105(16) CDB-4  
July 2, 1987

From : Carl Burch

To : X3J3

Subj : Unbuffered Bit I/O

I was asked to prepare a discussion proposal for Subgroup 16 along the lines of the FORTRAN IV extension usually referred to as BUFFER IN/BUFFER OUT, but less the asynchrony. While the requirement is somewhat vague to me, the main points seem to be :

- the ability to read bit-string magnetic tapes, and (secondarily) other bit-represented data streams (A-D converters, datacomm packets, etc).
- the suppression of "green words" in the implementation.
- unbuffered transfers, primarily for performance.

I see two general approaches to this kind of functionality. The first would define a third class of file format (in addition to FORM='FORMATTED' and 'UNFORMATTED' in the OPEN), which I'm calling 'UNBUFFERED' for the interim (Note : using the same first letter is probably not a good idea - I know of at least three implementations that only look at enough letters to disambiguate the alternatives). The transfers would be done by the regular READ and WRITE statements, but with an iolist restricted to one contiguous block of memory (said in standardese, of course). The ACCESS= specifier would probably have to be 'SEQUENTIAL'.

The second option would be to define a set of intrinsic subroutines that would limit the operations to be done by their syntax (i.e., only allowing one variable argument for a buffer would handle the contiguous requirement except in the case of sections with stride greater than one). These would look something like :

```
CALL BUFFER_OPEN (UNIT=unit, FILE=file-name-expr)
CALL BUFFER_OUT (UNIT=unit, BUFFER= buffer_var)
CALL BUFFER_IN (UNIT=unit, BUFFER= buffer_var, LENGTH=length)
```

The buffer in each case would be required to be a *variable*, except that non-contiguous sections and aliases must be barred. The LENGTH argument is as in the CDC definition, which returns the number of words (processor-defined units, probably) actually transferred to the buffer. If the "record" read is too long for the buffer, the excess is ignored. If the data is shorter than the buffer, the unused portion is unmodified.

In both the above options, other use of the unit would probably have to be barred (it is difficult to BACKSPACE an A/D converter, and it definitely will not do direct access). This is necessary to ensure against implementations that add green words in order to support BACKSPACE and the like.

This proposal is strictly for discussion at this point - we need to define the problem much more rigorously before going ahead with concrete text.

A excerpt from the CDC FTN4 manual follows for background.

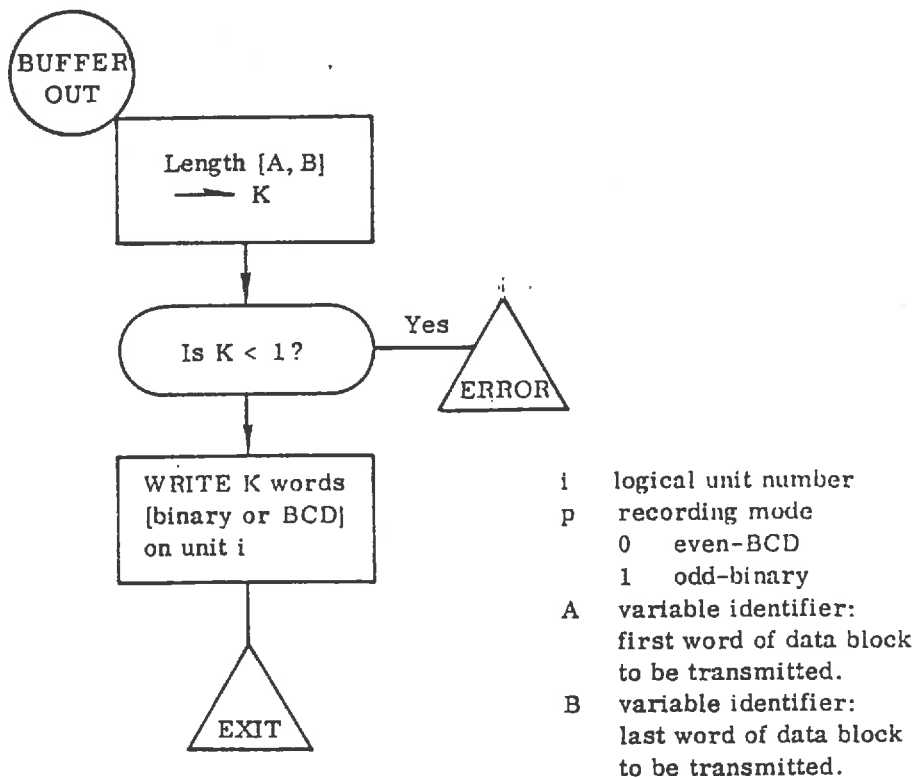
**10.5  
BUFFER  
STATEMENTS**

The primary differences between buffer I/O and read/write I/O statements are given below:

1. The mode of transmission (BCD or binary) is tacitly implied by the form of the read/write control statement. In a buffer control statement, parity must be specified by a parity indicator.
2. The read/write control statements are associated with a list and, in BCD transmission, with a FORMAT statement. The buffer control statements are not associated with a list; data transmission is to or from one area in storage.
3. A buffer control statement initiates data transmission, and then returns control to the program, permitting the program to perform other tasks while data transmission is in progress. Before buffered data is used, the status of the buffer operation should be checked. A read/write control statement completes the operation before returning control to the program.

In the descriptions that follow, these definitions apply.

- u logical unit number
- p parity key. May be specified by a simple variable (not subscripted). 0 for even parity (coded characters); non-zero for odd parity.



248



In the BUFFER statements the address of B must be greater than that of A. A unit referenced in a BUFFER statement may not be referenced in other I/O statements.

**BUFFER IN (u, p) (A, B)**

Information is transmitted from unit u in mode p to storage locations A through B. The use of this statement is described in detail under BUFFEI in Appendix I.

**BUFFER OUT (u, p) (A, B)**

Information is transmitted from storage locations A through B and one logical record is written on unit u in mode p containing all the words from A to B inclusive. The use of this statement is described in detail under BUFFEO in Appendix I.

Examples:

1) COMMON/BUFF/DATA(10), CAL(50)

PAR=0

BUFFER IN(9, PAR) (DATA(1), CAL(50))

BCD information is input from unit 9 to the labeled common area BUFF beginning at DATA(1), the first word of the block, and extending through CAL(50), the last word of the block.

2) DIMENSION A(100)

N=6

BUFFER OUT(N, 1) (A(1), A(100))

Binary information is transmitted to unit N from the block area defined by A(1) and A(100), that is, all of array A is transmitted.

250

(23)

105(\*) TAH-1

To: X3J3  
From: Tracy Hoover  
Date: 22 June 1987  
Subject: FYI

General  
-----

Since I have no real contact with Data General's F77 customers, I sent out copies of the grey book to approximately 15 of our biggest customers, i.e. those customers from whom we receive a large percentage of bug reports.

In addition to S8, I sent out a list of questions/statements that I asked them to consider as they were reviewing S8. These questions ranged from the language as a whole to particular features to performance issues to the document itself. In particular, I asked questions about many of the issues that had been raised in the negative ballot comments.

So far, I have received four detailed responses and one short statement. (They didn't have time to prepare a detailed response.) A couple customers responded with more of a DG F77 focus than to Fortran in general, but overall, some good points were made.

The first six items in the list of "Unfavorable Comments" were the ones mentioned most often. Please be aware the some customers were adamant about the inclusion of items such as pointers and BIT, and some said "It would be nice if..."

I'm not requesting any committee action on any of these things; in light of recent developments, I thought you might be amused/interested to read yet another set of reactions to 8x.

251

### Amusing Comments

---

"A reasonable combination of compiler and execution efficiency would satisfy most of our users. None of us is interested in sacrificing both to some professorial ideal of abstract programming: remember that real programmers don't use PASCAL! And they won't use FORTRAN 8x [sic] either!"

"The derived data types are great! Now I won't have to use languages like Pascal and C!!!!!"

"I am overwhelmed but impressed by this draft proposal. Despite the predictions of the demise of FORTRAN by C, Ada, and Pascal devotees, surely now FORTRAN will live forever."

"It looks to me as if the committee were dominated by theoretical programming people instead of by practical problem solvers."

"Please DON'T leave out INCLUDE!!!!!!!!!"

### Favorable Comments

---

1. Applauds free source form, but thinks we should have adopted the entire ASCII character set.
2. Applauds the adoption of namelist; DO...END DO; DO WHILE; and CASE constructs.
3. User-defined data types are 'interesting.'
4. Pleased to see IMPLICIT NONE, unlabelled DO.
5. Delighted with '&' for continuation character.
6. Upward compatibility is critically important.
7. Module construct may be useful in many contexts.
8. Multitude of new intrinsic functions is great.

### Unfavorable Comments

---

1. Pointer facility is mandatory. (3/5)
2. Continue to allow DO loop indices to be of type REAL. (2/5)
3. Disappointed about lack of INCLUDE facility. (4/5)
4. No provision for integers of different lengths. (2/5)
5. Inclusion of BYTE, LOGICAL\*1, BIT, along with facilities to manipulate these types would have been desirable. (4/5)
6. Lack of exception handling. (3/5)

7. Notation of combining several variable specifiers in one statement is unaesthetic and clumsy.
8. Method for specifying precision of REAL and COMPLEX variables is objectionable and carries FORTRAN too far from the machines on which codes are executed.
9. The idea of deprecating certain FORTRAN "locutions is pejorative."
10. More complex array operations will be difficult to optimize and will lead to inefficient programs.
11. Users will continue to use F66 and F77 compilers despite the availability of an F8x compiler because:
  - a. cost of converting existing programs will be high;
  - b. existing compilers are adequate;
  - c. F8x does not provide a significant increase in functionality over DG's F77;
  - d. F8x does not provide badly needed functionality such as byte functions, pointers, bit data type, bit string facility.
12. F8x will not be used if it results in reduced runtime performance.
13. Deprecation of ASSIGN and assigned GOTO is short-sighted.
14. PAUSE should be standardized, not deprecated.
15. New features sections in introduction does not seem to be visible. A couple customers requested a separate section of new features or to have them highlighted in the text somehow.
16. Document issues:
  - a. rather dry
  - b. difficult to read syntax rules
  - c. too much sectional fragmentation -- more indentation
  - d. not enough examples or diagrams, especially for the new array stuff
  - e. index is incomplete.
17. The increase of language complexity is not an issue, but users are not willing to trade performance for an increase in functionality. Moderate increases in compile time is OK, but any increase in runtime performance is completely unacceptable.
18. The term "deprecate" is pejorative.
19. "We will not use Fortran 8x if it results in reduced runtime performances."

Interesting Concepts

1. Forget about upward compatibility. Create a good F8x and require a preprocessor to convert from old features to new features.
2. One user is prepared to accept the removal of deprecated features over a period of a few months!

(29)

Subject: DERIVED TYPE I/O 06/29/87

015(16)JHM-01

To: X3J3

From: J. H. Matheny

REFERENCE

85(9)BTS-4  
93(16) GEW-1  
X3J3/S8.104

INTRODUCTION

This paper consists of notes about what a derived type input/output facility might be. The topic was discussed, with other proposals, at the 93rd Meeting in Monterey CA the week of February 11, 1985. At that time a straw of 25-0-8 said that we should defer this topic until a future time -- that completing /S8 was of the primary importance, and that this topic could be sufficiently satisfied with no additions or changes to the language.

In the discussion, several alternatives were presented, with no clear guidance from X3J3 as to a choice. These were:

1. Do nothing special for derived type,
2. Provide a syntax so that a format processor can import pieces of constant format specification from the derived type.
3. Provide CASE selection within the format specification,
4. Provide fragments of format specification referencable by the format processor.
5. Have the derived type build a character string format, somehow using the provided format specification to handle other list items.

At that time we had to consider variant structures, which complicated the picture.

DESIDERATA

1. A capability to perform arithmetic while editing a result should be available. For example, for type rational it might be appropriate to divide the numerator (as real) by the denominator and output this with a G edit descriptor.
2. A need to mix (intersperse) a derived type with other derived types and/or with intrinsic types.
3. A need to have the derived type structure pure is

255

perceived. A possibility is to include an edit descriptor as a part of the structure, (GEW)

4. It should handle well the multi-line/card case.
5. The module should be hidden from the user, and its components hidden.
6. Pieces of a format specification should be accessible in both a character format specification and in a FORMAT statement.

### POSSIBILITIES

1. Only elements of the structure are considered for input/output. This works for edited I/O, and also for unformatted I/O, list directed formatting, and Namelist formatting. It is what is in /S8 now.

a. A format specification merely addresses the elements as in Fortran 77.

b. A character format specification works. This could be a specification in the Module.

c. The user can concatenate the scrap of format specification into his format for the iolist, in the character case.

d. A character function in the module can perform arithmetic.

2. Syntax in the format specification could retrieve a character string from the module that is a piece of a format specification. This stringlet could be a part of the structure that is the derived type, or it could be a stand-alone character statement in the module that contained (DATA) the specification.

3. Extending #2 above, the format could be a character function in the module that returns the stringlet that accommodates the derived type perhaps using some arithmetic or selection mechanism, and also provides arithmetic on the datum.

### SYNTAX

Some syntax is necessary to transmit a piece of format specification from the module to be concatenated into the format specification. This syntax should work for both a character string, and a character function. A leading % will work.



EFFICIENCY

In Monterey, one of the objections to all of this was efficiency -- the fact that FORMAT statements could be pre-digested by the compiler. This device can also be used for format specifications in character constants, and in general.

There would be a performance cost for this capability only when derived type and the % are used. (Of course there likely would be a memory space effect.)

I believe that this is a straw man. Formats could still be cracked in the compiler, leaving pointers to the string or function in the module. The module could be caused to create cracked pieces of format. The power of the processor is still a business decision -- Fortran 8X would not preclude cleverness.



(25)

Subject: KANJI type

07/01/87

105(16)JHM-02

To: X3J3

From: J. H. Matheny

REFERENCE

#38, Supplement to 104th, Japanese Ballot Response  
#52, Supplement to 104th, Supplement  
X3J3/S8.104 -- Fortran 8X

INTRODUCTION

At the Belleview meeting the Japanese (A. Aoyama, H. Wada) made presentations on their perceived need for a Kanji intrinsic data type. The Module/Derived type model was deemed insufficient, in part because of the lack of specific format editing capabilities.

In looking at the editing requirements, I drafted an intrinsic module for Kanji. I found no need for extensions to Fortran 8X editing. Of course this Module has not yet been reviewed by others for correctness and completeness, and hasn't been compiled. It is a basis for discussion.

MODULE MODEL

MODULE KAN

```
TYPE KANJI (MAX_SIZE)
  CHARACTER (LEN = N) ESCAPE_TO_KANJI
  CHARACTER (LEN = 2 * MAX_SIZE) VALUE
  CHARACTER (LEN = N) ESCAPE_FROM_KANJI
END TYPE KANJI
```

```
!
SUBROUTINE ASSIGN_KANJI (KAN_LEFT, KAN_RIGHT) ASSIGNMENT
  TYPE (KANJI(*)) KAN_LEFT, KAN_RIGHT
  KAN_LEFT%ESCAPE_TO_KANJI = KAN_RIGHT%ESCAPE_TO_KANJI
  KAN_LEFT%VALUE = KAN_RIGHT%VALUE
  KAN_LEFT%ESCAPE_FROM_KANJI = KAN_RIGHT%ESCAPE TO KANJI
END SUBROUTINE ASSIGN_KANJI
```

```
!
FUNCTION CONCAT_KANJI (KAN_LEFT, KAN_RIGHT) OPERATOR (//)
  TYPE (KANJI(*)) KAN_LEFT, KAN_RIGHT
  CONCAT_KANJI = KAN_LEFT // KAN_RIGHT
END FUNCTION CONCAT_KANJI
END MODULE KAN
```

DISCUSSION

In this discussion, think of bytes, 8-bit bytes. The intrinsic character unit is a byte. The requirement is a 2-byte entity that can contain 16-bit characters -- maybe KANJI.

The requirement, as I understand it, is to intersperse 16-bit

259

characters with 8-bit characters in input and in output. Also desired is to intersperse 16-bit characters with 8-bit characters in the source form of Fortran.

I have no understanding of the hardware that can accommodate 16-bit characters. Here I assume that input/output can be thought of as a stream of bytes. The switch (toggle) that says an input stream or an output stream piece is KANJI is itself a pattern of bits in a byte, two bytes, or whatever. This is a serial transmission of the stream of bytes. This pattern is N bytes long in the above code.

Some hardware operates in a parallel mode, rather than in a serial mode. The classic "high-speed" printer is in this class. Also the 3270 style terminal is parallel in that the transmission unit is a screen. Such devices are not considered further here. In some sense, or at some place, a serial stream of bytes can occur -- maybe over a telephone line or into or from a buffer.

Where a pattern of bits can cause the processor to recognize KANJI or Latin characters, the above MODULE may solve the requirement. This pattern of bits is a part of the character string, and hence, given the A edit descriptor, input/output is covered without anything that is not now in /S8. Of course efficiency can be gained by causing the processor to swallow redundant toggle settings. In source form this shift can be caused to happen -- likely the keyboard has something like this now.

The requirement states the need for an intrinsic type. Proposed here is a derived type. This is so that portability can happen. The module, when correct, can be published, distributed with computers and Fortrans, whatever. It will work on my micro at home -- when I have a Fortran SX processor. It would have problems unless I buy and install the appropriate software drivers, hardware cards, and a proper monitor, to accommodate KANJI, but all of this is beyond the scope of Fortran. And I probably would not buy such gear -- I can't read or write Japanese. When I installed MS DOS 3.2, I promptly dropped the elements that gave me Scandinavian O's, the European currency and date forms, etc.

For some years, the architecture of Fortran SX included a core and a set of modules. A module could have been Japanese. A buyer of a Fortran processor could pick and choose the modules he wanted, and the implementor could pick and choose the modules to be implemented and still call the processor Fortran. Most of X3J3 was finally convinced that this COBOL approach had and would lead to a lack of portability, and in due course the derived type/module architecture was developed.

A stated concern (reasonably enough) is execution efficiency. It was felt that an intrinsic type would be more efficient than a derived type. There is nothing in the Fortran 8X Standard that says that a processor cannot look at a USE name, verify the contents of the module, and then generate whatever clever code that it is capable of. To implement such a facility in a processor is a business decision.

The facility described here is incomplete in that the several character intrinsic functions are omitted. And, of course, functions can be developed to concatenate Kanji with Character on both the left and the right. It does not provide a substring capability in KANJI terms.

262



To: X3J3

105(\*)JKR-1

From: John Reid

Subject: Meeting minutes

Date: 15th June 1987

I would like to thank all the scribes for their support at the Seattle meeting. Only two were too late for inclusion in the minutes and these have now arrived and are attached to this paper. Obviously the styles vary, but I think that the product is perfectly acceptable; I hope that you agree. Certainly, this mode of operation makes my job tolerable and I want to continue with it.

I have decided that I cannot avoid taking some responsibility for what is in the scribe notes. I have therefore read them through and in a few cases made some minor alterations. The quality of the typescript varied; some were not 'top copies', some used rather a small font, some used both sides of the paper which gives problems for 'cutting and sticking'. Guidelines are clearly needed. Here is a specimen set of notes.

## Specimen scribe notes

Discussion leader: Reid

Scribe: Another

Reference: 104(\*)JKR-3 AGENDA

Reid: Begin with a brief overview of the topic by summarizing the presentation. Omit this if the title already does it.

Straw Vote: The scribe notes must be posted by air mail to the secretary (J. K. Reid, Blg 8.9, Harwell Laboratory, Oxon OX11 0RA, England) in the week following the meeting. (30-0-0)

Jones: What format do you want?

Ans: Copy these notes as closely as possible. If you do not have bold, use underlining. Please send a top copy, printed on one side of the paper, and I prefer that it is not folded.

Motion: If a proposal in a paper is amended, the discussion leader must provide an amended copy of the paper to the librarian (Marshall) before the end of the meeting (Reid, Adams).

Formal Vote: 27-0. Passed.

763

# 19 Change to SET RANGE

Discussion leader: Ragan

Scribe: Rolison

Reference: 104(\*) Document 45 (from Subgroup 15)

## Summary:

The changes to be discussed were prepared at Torrance but not acted upon in Albuquerque (see Document 66 from Meeting 103). The inefficiency in the current reversion scheme was noted in the Allison ballot and possibly the Hirschert ballot. Currently, the bounds of a ranged array revert to the declared bounds. This is too much work especially in a program with multiple subprograms where the rangeable array is declared in a module. The last subprogram using ranged bounds must remember to restore the declared bounds when it exits. The current scheme is not analogous to what happens to common block storage - the intent of this proposal is to make rangeable bounds analogous to common. That is, when the last subprogram exits, the bounds are undefined. The change basically is that a SET RANGE must now be done before the effective bounds are to be used (this means only the first routine needs to copy the bounds).

Motion: The changes described by Document 45. (Ragan, Hendrickson)

## Discussion:

Reid: Not all operations are valid [on an array that has not had a SET RANGE performed on it].

Ans: It's the same as an identified array that is not alias associated - you can not use it.

Reid: What about use in intrinsics (such as ESHAPE)?

Ans: Item 4 [of the proposal] seems to take care of this. The intent is that an array with no SET RANGE can not be an argument to such an intrinsic. I don't think that the relaxation rule just passed in Albuquerque affects this.

Formal Vote: (32-0)

# 21 Reaffirmation of Albuquerque decisions

Discussion leader: Campbell

Scribe: Rolison

Reference: 104(\*)BTS/LWC-4 Significant Changes to S8 at March meeting

## Summary:

This document summarizes what Brian and I believe were the significant changes made to S8 at the Albuquerque meeting.

## Discussion:

Weaver: Albuquerque was a valid meeting. Why do we need to vote to reaffirm?

Adams: Let's take a straw vote. "Yes" means we should vote to reaffirm. [Scribe records do not show a vote tally - I believe no straw vote was taken.]

B. Martin: Do we need to revote because we now need a 2/3 majority?



Harris: It's more a problem of 1 over 1/2 because we did not have that in Albuquerque.

Weaver: I assumed votes in Albuquerque were valid.

Wearing: In Torrance, we were told we would have a chance to reaffirm.

Adams: Is there a proposal?

Motion: I propose we reaffirm the changes listed in Item 44. (Hoover, Johnson)

B. Martin: Isn't there another change?

Campbell: Yes, construct names on IF and CASE.

Adams: Voting members only...

Harris: Wait - what about discussion? I'd like to poll items 3, 9, and 11 separately.

Adams: OK, everything except those.

Formal Vote: (28-0)

Adams: Is there a problem with 3?

Harris: There's no problem but I'm concerned about interface blocks.

Reid: The general rule on the host scoping unit covers it. Copying the leading statements of an external procedure was a lesser priority than a simple rule.

Allison: What about IMPLICIT rules?

Reid: They're not inherited from the module. [That is, an interface block inherits the IMPLICIT rules from the scoping unit containing it, not from the module containing the subprogram.]

Motion: Item #3. (Hoover, Reid)

Adams: Number 9?

Harris: I think this resolution is rather painful because the parser must be context-sensitive to know when and when not to evaluate an expression. What is the rationale?

Hendrickson: We don't expect inquiry functions typically to have expression arguments and none depend on the value of the argument.

Harris: I agree but why not just prohibit expressions?

Hendrickson: It seems equally unusual to have some functions that do not accept operations (that is, an expression that consists of something other than just a variable).

Hirchert: Some inquiry function can not be evaluated unless the argument is evaluated. There is a distinction in FORTRAN 77 between "variable" and "expression".

Allison: There are array inquiry functions that can not be evaluated at compile time.

B. Martin: Dick and Kevin seem to be arguing about two different things. Why not separate variables from expressions?

Burch: I'm concerned about Kurt's point about allocatable arrays.

Reid: [quotes previous wording] Perhaps we should go back to the previous wording. [To Dick:] How about it?

Hendrickson: I find that to be unacceptable - it's not portable.

Reid: Yes, in these cases one may have to be careful.

Marusak: Is this point really that important?

B. Martin: John's reading makes me think that the old wording is fine. We already have such things in the language. Consider the evaluation of conditional expressions: some processors may not evaluate the second half of an AND.

Harris: I now find the old wording to be acceptable.

Hirchert: If it's a variable, make a rule that it must be defined.

Hendrickson: I've been convinced by John and Bruce that other similar cases already exist. I now think Kurt is right.

Straw Vote: "Yes" means strike item #9 (that is, reject "Campbell number" 1847). (28-9-7)

Motion: Rescind item #9. (B. Martin, no second recorded)

Reid: That's not good enough.

[Discussion on what the motion should be. Bruce defers to John for the following motion.]

Motion: Replace p. 13-3, line 29 and p. 13-4, line 34 of S8.104 by the corresponding text in S8.103A. On p. 13-1, line 42 add "The value of the argument to this function need not be defined. It is not necessary for a processor to evaluate the argument of this function if the value of the function can be determined otherwise." (Reid, B. Martin)

Reid: I'd like to table the motion to think about it. [John attempts to withdraw his motion.]

Harris: I'd rather move to add the amendment now.

Reid: Let's go back to the motion then.

Motion: Move to amend. (Smith, Hendrickson)

Brainerd: But the qualifier is true for all functions. Why just say it here?

Hendrickson: To emphasize the point. Is a processor standard-conforming if it evaluates when the standard says "it is not necessary"?

Ans: Yes.

Formal Vote: (31-0)

Adams: Item 11?

Harris: Is this a constraint that must be checked?

Campbell: In text.

Harris: So then a processor could make an extension that allow this (doesn't check the restriction due to overhead). The user benefit would be lessened because processors may not check it.

Motion: Item 11. (Brainerd, Hendrickson)

Formal Vote: (26-0)

Brainerd: Item #13 is an example of what should not have been done in Albuquerque.

Straw Vote: "Yes" means remove construct names. (8-10-12)

## 23 Response to Lakhwara's ballot

Discussion leader: Ragan

Scribe: Morgan

The session began with Rich Regan describing the modified ballot response. The response included in the pre-meeting distribution 104(\*)JCA-3, page 17, was modified by removing section 3. and replacing it with item 46 on the table (a redrafted response agreed in subgroup). Several mistakes in the modified version were identified and changed.

Regan: I move the response, as modified, be accepted.

Millard: I second.

Adams: Any discussion.

Marusak: Do we know how Anil feels about the response.

Regan: No, but we gave the best response we could.

Adams: As long as we get 2/3 plus one over half, thats ok. He will still get the right to comment on the reply.

Weaver: Is there any vendor present who believes that MODULE/USE is implementable as filtered INCLUDE. This would not be commercially acceptable (marketable).

Rawlinson: I'm not sure what you mean by 'filtered INCLUDE'.

Hendrickson: I have difficulty with Item (ii) on item 46. I'd like the wording changed e.g. delete the last sentence.

Regan: OK change it to 'but X3J3 does not believe that the actual size will be out of proportion...'

Marusak: [some comment on that sentence]

Regan: OK delete the second sentence of point (ii) [on item 46].

There was no further discussion. The vote was taken and passed 27-2.

11th June 1987

5 of 5

105(\*)JKR-1

267



To: Fortran Forum, BCS, NAG, etc.  
 From: John Reid  
 Subject: X3J3 meeting in Seattle  
 Date: 18 May 1987

Note. This is a personal report of the meeting and in no sense does it constitute an official record of it.

1. Summary

The meeting was devoted to deciding whether to send the current draft standard to X3, the parent committee, and processing the responses to the comments made by members with their letter ballots on this issue. The decision was "yes", but the vote (26-9) was such that there is likely to be a delay before the draft is released for public comment while X3 considers the comments accompanying the "no" votes. This was a disappointment to those of us who wish to see the public comment period beginning as soon as possible, but at least we have reached another formal milestone.

Some editorial changes were made (few compared with the previous meeting) and one technical change was made (see section 6). It was a tense meeting and there was some discussion of the membership rules, but the friendly personal relationships that are a hallmark of X3J3 were maintained.

2. X3J3 secretary

Jeanne Martin has resigned as secretary and I have been appointed in her place. She has performed her duties very competently and conscientiously since 1982 and was warmly applauded by the committee to mark its appreciation. I thought hard about whether to accept and decided to do so only on conditions that will greatly reduce the burden and that I thought might be unacceptable because the quality of the minutes will deteriorate. I am recording formal decisions only and relying on committee members to provide scribe notes that I will use to construct the minutes by "cutting and sticking". Since I always have recorded the committee decisions for myself, this will make little difference to my participation in technical discussions.

I would like to make it clear that as secretary I will do my best to ensure that the minutes are an accurate and unbiased record of the meeting, but these notes are personal notes and do include my personal opinions. They are constructed by making minor changes to a report that I write for people in my own organisation to explain how I have been representing them.

3. Voting during the public comment period

The chairman has received a ruling from X3 on changing the draft standard during the public comment period. A new standing document (S12) will be created, starting with the present draft standard. Previously, it had been assumed that changes to S12 could be made by normal majority votes and that the 2/3 rule would apply only to the final adoption of S12 as a replacement for S8. However, we have been told that a 2/3 vote is required for all changes to S12. Personally, I approve of the rule; if about half the committee think one way and the other half think the other, I think that we should keep what we have. However, the immediate effect is that those that do not like aspects of the present draft are concerned that it will be harder to make future changes. This was a reason for several "no" votes.

4. The ballot on forwarding the draft

The vote in the January letter ballot was 29-7 in favour. The changes since then were as follows: Dick Hendrickson (Cray) switched from "no" to "yes", being satisfied with the responses to his comments. One member who voted "yes" has left the committee. Three members were absent (two "yes" votes and one "no" vote in the letter ballot). There were three new members (two "yes" and one "no" vote). Three members switched from "yes" to "no". Thus the final vote was 26-9 in favour. The following voted "no":

Steve Adamczyk (Adv. Comp. Tech.) voted "yes" in the letter ballot, saying "I do not agree with everything in the draft, but it is a reasonable compromise and is ready to go out for public comment". He now thinks that it should be limited to standardizing existing practice and should contain more explanatory text to assist public comment.

Bob Allison (Harris) voted "yes" in the letter ballot, saying that he had serious misgivings but was willing to see if a public review indicates that the public has similar misgivings. He was surprised by the large number of comments and worried by the clarification of the 2/3 majority rule, which will make it more difficult to change the draft during the public comment period.

Valerie Bowe (Unisys) reaffirmed her letter ballot.

Kevin Harris (DEC) reaffirmed his letter ballot.

Alex Marusak (Los Alamos) voted "yes" in the letter ballot but changed to "no" because the clarification of the 2/3 rule will make it harder to change the draft. In particular, he wants to add BIT type.

Len Moss (SLAC) was represented by Sunny Sund, who reaffirmed his vote and was concerned about the 2/3 rule.

Ivor Phillips (Boeing) reaffirmed his "no" vote.

Alison Wearing (NCC, Manchester) is a new member who feels that the document is not sufficiently well written.

Dick Weaver (IBM) reaffirmed his "no" vote.

5. Membership rules

There was some discussion of the membership rules. Familiarity with the whole language will become very important during the public comment period. In fact, a new member has only to attend one meeting on provisional status and is thus given preference over an old member who reverts to provisional status if he or she has not attended two of the previous three meetings. There was sentiment at least to apply to the same rule to new members as old ones, but the standing rules may not allow this.

6. Changes to SET RANGE

At the request of implementors, it was decided that an array with the attribute RANGE should have no effective shape until a SET RANGE statement is executed for it.

7. Interpretation of the LEN function in Fortran 77

The committee received a request from Andrew Tait of Amdahl for an interpretation of permissible arguments for the LEN function. He was told that in a standard-conforming program, the argument must be an expression and not an array or procedure name.

8. Multibyte character sets

Akio Aoyama and Hideo Wada from Japan made a request for X3J3 to consider character sets, such as Kanji, that need a two-byte representation. One possibility might be to introduce another type parameter for type CHARACTER, with a default value that corresponds to the usual one-byte representation. X3J3 pointed out that characters may presently be implemented in more than one byte, but this would probably be too inefficient. Another possibility is to set up an intrinsic module, though i/o may be awkward. The committee decided to explore this further as an example of the use of modules.

9. Presentation by David Jones of Microsoft

Microsoft has its headquarters nearby, and David Jones requested to make a short presentation. He was satisfied with the quality of the document and estimated that Fortran 8x is about double the size of Fortran 77. In a user survey, he found little interest in the Fortran 8x features but a demand for common mainframe extensions.

10. Date of next meeting

The next X3J3 meeting will be in Liverpool, August 10-14. Contact J.L. Schonfelder, Computer Laboratory, University of Liverpool, Liverpool L69 3BX, if you wish to attend. ISO/TC97/SC22/WG5 is also meeting in Liverpool, August 3-7. Again, contact J.L. Schonfelder for details.



28

TO: X3J3  
FROM: Jerry Wagener  
SUBJECT: Standing Document on Membership Policies

At X3J3 meeting #104 in Seattle, it was decided to make the one-page summary of membership policies and procedures an X3J3 standing document. The attached page is proposed as that standing document.

The changes in this version from that included in the distribution for meeting #103 (Los Angeles) are:

1. addition of the last sentence under item 1,
2. "membership dues" changed to "service fees" in item 5.

Otherwise this version is the same as the meeting #103 version.

Most of this material is a condensation and paraphrasing of SD-2 material, and I believe is completely consistent with the SD-2.

2

**Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\***

Jerrold L. Wagener  
Amoco Production Research  
P. O. Box 3385  
Tulsa, OK 74102  
(918) 660-3978

Summary of X3J3 Membership Policies and Procedures

1. There are two forms of membership, principal members and alternate members. Principal members are the voting members of the committee; alternates are designated by principal members to represent them in the event of their absence. Normally a principal member will have one alternate, though more are allowed or a principal member may elect to have no alternate. Membership requirements are the same for both principal and alternate members. Normally there will not be more than one principal member from a given institution.
2. Members (both principal and alternate) are appointed by the X3J3 chair, upon written petition. Application for membership may be made to:

Jeanne C. Adams, X3J3 Chair  
Scientific Systems Division  
National Center for Atmospheric Research  
P.O. Box 3000  
Boulder, CO 80307

The chair may also appoint individuals in various advisory capacities, such as consultant, liaison, or observer.

3. Letters of application for membership should contain descriptions of the proposed member's expertise in Fortran and interest in the Fortran standard, and a statement of the proposed member's ability and willingness to participate in the committee's activities. X3J3 members should be experts in the Fortran language. Effective participation in X3J3 activities normally requires at least a 20% time commitment.
4. Principal members must attend (or be represented at) two out of three consecutive X3J3 meetings in order to avoid provisional status. Those on provisional status may not vote at the next meeting, and normally lose membership if not present (or represented at) the next meeting. Similar rules apply to participation in X3J3 letter ballots. Normally X3J3 has four one-week meetings a year. (The current schedule is the second week of February, May, August, and November, though this schedule is subject to change.) X3J3 meetings are open to the public.
5. Principal members are charged service fees by the X3 secretariat. Currently these are \$175/year for the member and one alternate; additional alternates are currently charged \$75/year. All members receive all committee documents. Anyone else may also receive all committee documents by obtaining observer status, which currently costs \$100/year.

\*Operating under the procedures of The American National Standards Institute.

274

29

TO: X3J3  
FROM: Jerry Wagener  
SUBJECT: Responses to WG5 Resolutions

At X3J3 meeting #101 in Halifax it was decided that I would keep a file of the current status of X3J3 responses to the resolutions passed by WG5 at their Halifax meeting. From time to time the contents of this file would be informally communicated to WG5, and after the responses have been completed they would be formally communicated to WG5. The current contents of the file are as follow. Responses labeled "(unofficial wording)" were constructed since the last X3J3 meeting.

Halifax/1a - Importance of Public Review

Submitting Fortran 8x to public review is X3J3's top priority. X3J3 appreciates WG5's concurrence.

Halifax/1b - Form of the Public Review Document

(unofficial wording)  
Appendix F is now entitled "Removed Extensions", which would appear to be similar to option (d) of this resolution. Options (a), (b), and (c) were considered, but X3J3 decided on keeping the appendix for public review and labeling it as indicated.

Halifax/2 - Size of Language

X3J3 interprets this resolution as WG5's endorsement of its work through 1985, and acknowledges WG5's distress at the reduction of the size of Fortran 8x following the March 1986 X3J3 letter ballot.

Halifax/3 - Temporary Nature of an Extension Features Appendix

(unofficial wording)  
No final decision has been made on this issue, and it may be deferred until after the public review period.

Halifax/4 - Consultation Between WG5 and X3J3

275

X3J3 notes the effective working relationship that has existed between WG5 and X3J3 for many years, and interprets this resolution as WG5's intent that this relationship should continue. X3J3 strongly shares this intent, and welcomes WG5 input at any time.

X3J3 will make every reasonable effort to insure that WG5 members receive appropriate X3J3 documents, particularly premeeting distributions, meeting minutes, and draft standard documents (S8), on a timely basis. Members of WG5 are welcome, and encouraged to attend and participate in X3J3 meetings.

X3J3 will make every reasonable effort to obtain and consider input from WG5 on all issues, and encourages WG5 to meet as needed to provide such input. However, in order to expedite the release of Fortran 8x for public review, the top priority of both WG5 and X3J3, X3J3 cannot afford to routinely delay its work pending WG5 input. X3J3 does not believe that WG5 intended this constraint in adopting this resolution, and assumes that the cooperative activities summarized here satisfy the resolution's intent.

#### Halifax/5 - Distribution of Information

X3J3 concurs, and will cooperate fully in promoting such distribution.

#### Halifax/6 - WG5 Schedule

Completed, as of November 1986.  
Schedule may be updated in the future as may be appropriate.

#### Halifax/7 - ISO Documents

X3J3 concurs, and notes that this is being done.

#### Halifax/8 - Document Numbering

X3J3 concurs, and notes that no action is required from X3J3.

#### Halifax 9 - Language and Style

(unofficial wording)

X3J3 has made extensive efforts to improve the readability of the document. These efforts include the addition of numerous examples and the incorporation of many suggestions for improving the editorial style.

#### Halifax 10 - Processor Conformance

(unofficial wording)

Section 1.4, entitled "Conformance", was added at X3J3 meeting #103 in Los Angeles (February, 1987). This addition provides substantially that requested by this resolution.

#### Halifax 11 - Pointers

(unofficial wording)

Pointers are not included in the public review draft. However, work is continuing on this issue, and a tutorial on pointers is on the agenda for X3J3 meeting #105 in Liverpool in August 1987.

#### Halifax 12 - Data Abstraction

Reinstatement of intrinsic operator overloading was adopted by vote of 22-3 at the August 1986 X3J3 meeting in Halifax. At the same meeting reinstatement of structure constructors was approved by a vote of 25-1. These actions are in accordance with this WG5 resolution and WG5 resolution Halifax/19.

#### Halifax/13 - Deprecated Features

(unofficial wording)

X3J3 appreciates the spirit of this resolution, but prefers to retain two levels of decremental features, obsolescent and deprecated, with only the obsolescent features explicitly identified in the standard itself. Many individuals sympathetic with the concept of deprecated features have advised X3J3 to "go slow" on their removal; the two levels of decremental features should provide a more restrained framework for disciplined language evolution.

#### Halifax/14 - Significant Blanks

Original text on significant blanks added to appendix F at the November 1986 X3J3 meeting.

#### Halifax/15 - Random Intrinsic Procedure

An intrinsic subroutine RANDOM, together with an accompanying subroutine RANDOM\_SEED, was adopted by a vote of 15-12 at the August 1986 meeting in Halifax. This action is in accordance with this WG5 resolution.

Halifax/16 - Character Intrinsic Functions

At X3J3 meeting #102 (Nov'86, Albuquerque) an optional third argument BACK (proposal 102.TAH-1) was added to intrinsic functions INDEX, ISCAN (name changed to SCAN), and VERIFY. This provides the capability of processing a character string backwards for these functions.

Halifax/17 - Translate Function -- (failed)

Halifax/18 - Allocatable Scalars -- (failed)

Halifax/19 - Structure Constructors

See response to WG5 resolution Halifax/12.

Halifax/20 - Operator Renaming

A motion (102.JLW-3, as modified) to allow renaming of operators of the form .xyz. failed 10-23 at X3J3 meeting #102 in Nov'86 in Albuquerque.

Halifax/21 - Internal Procedure Control -- (failed)

Halifax/22 - Name-directed I/O

A NAMELIST facility similar to existing implementations, and replacing the previously adopted name-directed I/O facility, was adopted by a vote of 21-6 at the August 1986 X3J3 meeting in Halifax. This action is at some variance with this WG5 resolution.

Halifax/23 - Procedure Interfaces

X3J3 has investigated providing such a feature (102.KWH-4), but thus far has found no acceptable syntax for providing such a feature in the context of Fortran.