

Supplement to the Minutes of Meeting 107 (Part 2)

## X3J3 Fortran

8 to 12 February 1988

New Orleans, Louisiana

X3J3/220



## CONTENTS

### Supplement to the Minutes

Papers Distributed at the 107th X3J3 Meeting

New Orleans, Louisiana February 8-12, 1988

Note: Document numbers with a 'A' suffix denote documents which were ammended at the 107th meeting.

Item Number		Page Number
9A - 107-CDB-2	Personal letter from Jim Matheney .....	29
15A - 107-IRP-1	The DATA Statement .....	43
21A - 107-KWH-3	REPEAT Function .....	105
22A - 107-KWH-4	Deallocating Function Results .....	107
30A - 107-LWC-1	Suggested Edits to S8 dated June 1987 .....	135
32A - 107-LWC-3	Letter Ballot Edits for S8 dated June 1987 .....	139
34A - 107-PLS-2	Miscellenous Edits .....	149
35A - 107-PCS-3	Changes to Section 9 .....	151
36A - 107-RAH-1	Rewrite of 106-RAH-2 on Source Form .....	157
43 -- 107-CDB-4	Subgroup Nominations for Public Review Letters 22-47 .....	178
44 -- 107-CDB-5	Subgroup Nominations for Public Review Letters 48 and 50-64 .....	272
45A - 107-TAH-1	Backward References in Section Notes .....	305
46 -- 107-JKR-2	Meeting Minutes .....	307
47A - 107-JKR-3	Changes to S16.107 .....	309
48 -- 107-CDB-6	Subgroup Nominations for Public Review Letters 10-21 .....	310
49 -- 107-IRP-4	Modifications to Subgroup Assignments for Public Review Comment Processing .....	323
50 -- 107-IRP-5	Subgroup Assignments .....	324
51 -- 107-JCA-6	UNISYS letter to Kachurik .....	329





52 -- 107-JCA-7	X3/SD-4	Projects Manual .....	331
53 -- 107-JCA-8	X3/SD-7	Meeting Schedule & Calendar .....	355
54 -- 107-JCA-9		Transmittal of December 1987 SD-6, Membership and Officers .....	368
55 -- 107-JCA-10	X3/SD-9	Policy and Guidelines .....	402
56 -- 107-JCA-11		Transmittal of Report of Overage Standards due for 5-Year Review .....	424
57 -- 107-JCA-12		Second Public Review for C .....	425
58 --		Meeting Agenda (as of Monday, Feb. 8) .....	426
59 -- 107-JCA-14		Proposal on Hanzi Processing in Fortran 8x .....	427
60 -- 107-JLS-2		Schonfelder letter to Lorin Meissner .....	429
61 -- 107-MBM-1		CERN Users Votes on Fortran 8x .....	432
62A - 107-AW-1		Revisions to S8.104(June 1987) .....	434
63 -- 107-ADT-1		Response to P. Sinclair's Comments 20 and 21 .....	437
64 --		Suggested Response for a Public Review Letter from the public	443
65 --		Datamation Article on Fortran 8x .....	453
66A - 107-JHM-2		Review, MOSI Draft Standard, P855/draft 7 11/1/87	456
67 -- 107-EAJ-1		Fortran 77 Interpretations .....	460
68A - 107-EAJ-2		Deprecated Features .....	462
69 -- 107-EAJ-3		Re-work of Low-Level Syntax and Source Form .....	463
70 -- 107-EAJ-4		Fortran 77 Interpretation on Multiple DO Terminations .....	467
71 -- 107-BAM-1		Lahey Comments on Fortran 8x .....	471
72 -- 107-MJAB-1		Comments from Thinking Machines Corporation .....	477
73 -- 107-MJAB-2		Thinking Machines Public Commentary on Draft Proposed Standard X3.9-198x .....	488
74 -- 107-WCL-1		Interpretation of Fortran 77 .....	496
75 -- 107-TMRE-4		British Computer Society Fortran Forum Report .....	497



76 -- 107-TMRE-5	UK response (NO) to the ISO Review of X3J3-S8 ....	511
77 -- 107-TMRE-6	British Computer Soc. comments on UK vote to ISO	514
78 --	Japanese Proposal fo Fortran 8x, NCHARACTER type .....	515
79A - 107-LWC-4	Corrections and Edits for S16 dated December 1987	542
80 -- 107-CDB-7	Subgroup Nominations for Public Review Letters 65-91 .....	544
81 --	F. Engel letter on X3J3 response to Fortran 77 Public Review	598
82 -- 107-ALM-2	Compiler-Verifiable Aids to Naming in Fortran 8x	609
83A - 107-JLW-2	X3J3/S14.107 (WG5 Resolutions, Liverpool) .....	611
84 -- 107-PLS-1a	SCRATCH Files .....	618
85A - 107-JKR-4	S8 Edits .....	621
86A - 107-LWC-5	Edits for S8.104 .....	623
87A - 107-JKR-5	Edits .....	625
88 -- 107-LRR-1	PUBLIC vs. PRIVATE fix-up with type definition ...	627



9a

107-CDB-2  
December 31, 1987

From : Carl Burch  
To : X3J3  
Subj : Blank Interpretation in Internal Files  
Encl : Personal letter from Jim Matheny.

**History**

This is an extension of 106-CDB-1, with more research added, mostly courtesy of Jim Matheny.

**Hole in FORTRAN 77**

A recent item on the ARPANET notes system asked what the F77 standard says about blanks read from internal files. I was unable to find anything at all. This is a hole in F8x as well. Jim Matheny points out an F77 interpretation with regard to preconnected files that seems relevant by analogy, at least :

- did not have a BLANK= specifier, blank characters in formatted numeric input fields are ignored because BLANK='NULL' is the default."
- nonleading blank characters in formatted numeric input fields is not specified in the standard and is therefore processor dependent."

The phrase "is not specified in the standard and is therefore processor dependent" strikes me as bureaucratese for "OOPS!". If it is supposed (and desired) to be processor dependent, let's say that in the dpANS, not in an Interpretation. Personally, I think that the time to move ahead is here. I propose that we standardize on BLANKS='NULL' as the default.

**Current Status**

Paragraph 10.6.6 of S8.104 specifies that blanks "are interpreted as zeros or ignored, depending on the value of the BLANK= specifier currently in effect for the unit." The BLANK= specifier is in the OPEN statement, which we don't have if the transfer is using an internal file.

**Proposal**

P.9-4, Line 32

Insert as a new paragraph :

(8) On input, blanks are ~~initially ignored~~ treated as though the format had an initial

P.9-6, Line 3 EN edit descriptor (10.6.6).

Insert after the paragraph : "On input, blanks are ~~initially ignored~~ treated as if the file had been opened with BLANK= NULL specified in an OPEN

P.10-4, Lines 40-41

Insert between lines 40-41 : ", default for a preconnected or internal file," statement (9.3.4.6),  
the

P.10-10, Line 28

Insert after "nonleading blank characters" : "from a file connected by an OPEN statement".

P.10-10, Line 30

Replace "unit." with "unit; ~~nonleading blanks from a~~ preconnected or <sup>an</sup> internal files ~~are initially ignored.~~ is treated as if the file had been opened with

BLANK = NULL.

29

**James H. Matheny**  
41 Silver Spring Drive  
Rolling Hills Estates, CA 90274  
(213) 375-5940

December 1, 1987

Carl D. Burch  
Hewlett Packard  
19447 Pruneridge Avenue  
M/S 47LH  
Cupertino, CA 95014

Subject: Interpretation of BN/BZ during internal file input.

The FIB-2, page 16, addresses nearly the same problem. If a file is connected by an OPEN statement, and BLANK= is not specified, the default is NULL. (Page 12-20, lines 26-27.) If there is no explicit connection by OPEN, the default is processor dependent. The FIB says that this was deliberate so that processors could implement BZ to conform with Fortran 66, and in time wean their users to BN. See also page B9, lines 5-11.

The FIB does not address internal files as such, but I believe that it covers the case. I think that it is finally approved, and thus is a part of the Standard. Page 16 is enclosed.

cc:

E. Andrew Johnson  
MS 10C17-3  
Prime Computer Inc.  
500 Old Connecticut Path  
Farmingham, MA 01701

## DEFAULT MEANING OF BLANKS IN NUMERIC INPUT FIELDS

9a

### Question:

What is the default meaning of blanks in numeric input fields? Is the default the same for preconnected files as for files connected with an OPEN statement?

### Answer:

For the subset language, nonleading blank characters are interpreted as zeros in formatted numeric input fields unless the BN edit descriptor has been used to cause them to be ignored.

For the full language, there are two default cases:

1. If the formatted file was connected by an OPEN statement that did not have a BLANK= specifier, blank characters in formatted numeric input fields are ignored because BLANK='NULL' is the default.
2. If the formatted file was preconnected, the interpretation of nonleading blank characters in formatted numeric input fields is not specified in the standard and is therefore processor dependent.

### Discussion:

For the subset language, the standard clearly states that the nonleading blanks in formatted numeric input fields are interpreted as zeros at the beginning of each formatted input statement. Subsequent interpretation of such blank characters is then subject to control by the BN and BZ edit descriptors.

For the full language, the standard clearly specifies that BLANK='NULL' is the default when the BLANK= specifier is omitted in an OPEN statement, and the standard does not specify any default when files are read without the explicit execution of an OPEN statement. This omission was intentional so that a processor might continue to use the X3.9-1966 interpretation of nonleading blanks as zeros in numeric input fields to provide compatibility with the X3.9-1966 standard. A processor may have a method of specifying the meaning of blanks in numeric input fields at the time of preconnection.

### References:

American National Standard X3.9-1978

<u>Section</u>	<u>Heading</u>	<u>Page</u>	<u>Lines</u>
1.4.1	Subset Conformance	1-3	1-6
12.10.1	OPEN Statement	12-20	18-29
13.5.8	BN and BZ Editing	13-8	13-28
B12	Section 12 Notes	B-9	5-18





- 8) Page 5-13, lines 16-17: replace "whose unqualified name" with "that".
- 9) Page 5-13, line 43: after "MILES" add ", NODE".
- 10) Page 5-13, line 44: after "SKEW" add ", DIRECTION".
- 11) Page 5-13, line 47: after this line insert  
 DATA DIRECTION / 10000\*90.0 /  
 DATA NODE (0:8:2) / 5\*0 /
- 12) Page 5-13, line 51: after the "." add "All 10,000 elements of the array DIRECTION are initialized to the value 90.0. The 5 even numbered elements of the array NODE are initialized to 0."

**End Option 2**

This definition of the list-oriented DATA statement would be more uniform with respect to the object-oriented DATA statement. In particular, it would permit array sections to be initialized. This, of course, is enhancing a deprecated feature. Initializing a variable of derived type or a structure component would still be forbidden.

**Option 3**

All of Option 2 plus the following:

- 13) Page 5-13, line 6: delete "of a derived type, a structure component,".
- 14) Page 5-13, line 18: before the "A" insert "A derived-type variable is equivalent to a complete ordered sequence of its components (4.4.1).".

**End Option 3**

This, additionally, removes the restriction on the list-oriented DATA statement that prohibits the initialization of derived-type variables or structure components.

-----

Whether or not one of the above proposals is accepted I propose the following changes in connection with the DATA statement.

**Proposal 1 (Editorial)**

- 15) Page 5-13, line 19: change "4.1" to "4.4".
- 16) Page 5-14, lines 18-20: delete the first sentence.
- 17) Page 5-14, line 26: after "constructs" insert "(8.1.4.4)".

15a

- 18) Page 5-14, line 37: after the " : A" add " , C".
- 19) Page 5-14, line 39: after this line insert

```
TYPE (STRING (6)) :: FRUIT
```

- 20) Page 5-14, line 42: after this line add

```
DATA (C = 0.0)
```

```
DATA (CHAIRMAN%AGE = 35)
```

```
DATA (FRUIT = STRING(6)(5, 'APPLE'))
```

where CHAIRMAN is defined <sup>in 4.4.1</sup> ~~on page 4-7, lines 8-13~~ and STRING is defined <sup>in 4.4.1.1.</sup> ~~on page 4-7~~ lines 42-45.

- 21) Page 5-14, line 43: insert after "odd" the word "numbered".
- 22) Page 5-14, line 43: insert after "zero." the following sentences "The array C is initialized to zero. The AGE of the derived-type variable CHAIRMAN is set to 35. The derived-type variable FRUIT is initialized with LENGTH equal to 5 and VALUE equal to 'APPLE '".

### End Proposal 1

The current definition of the *data-implied-do* in the list-oriented DATA statement permits a *substring* to be initialized. This is not in FORTRAN 77 and is, in a sense, more general than the object-oriented DATA statement. Moreover, the concept has not been correctly defined with respect to substrings as there are currently no definitions or restrictions on the form of the *substring ranges*.

Proposal 2 corrects this by eliminating this undefined capability.

### Proposal 2 (Error Correction)

- 23) Page 5-12, line 41: delete this line.

### End Proposal 2

Proposal 3 corrects an omission in the constraints of the list-oriented DATA statement.

### Proposal 3

- 24) Page 5-13, line 15: Insert after this line "Constraint: *scalar-int-expr* must not contain any variables, except for *data-i-do-variables* from an outer level of the *data-implied-do*."

### End Proposal 3

21a

Subject: REPEAT Function  
From: Kurt W. Hirschert

107(PROC)KWH-3 (Page 1 of 1)

Problem

In order for a function to be elemental, the attributes of its result must be independent of the value of its arguments. The length of the result of the REPEAT function depends on the length of the argument STRING (OK) and the value of the argument NCOPIES (not OK).

Proposal

Reclassify REPEAT as a transformational function:

1. Page 13-1, line 38. Delete "REPEAT,". *transformational* REPEAT  
Line 39. Insert new sentence: "The ~~transformation~~/function/returns repeated concatenations of a character string argument."
2. Page 13-40, line 3. Replace "Elemental" with "Transformational".
3. *Page 13-40, lines 5 and 6. Change "of type" to "scalar and of type", twice.*
4. *Page 13-40, line 7. Change 'Result ... Character' to 'Result Type, Type Parameters, and Shape. Character scalar'.*

Discussion

10 If we ever adopt the concept of functions which are elemental with respect to only selected arguments, REPEAT could be reclassified as elemental with respect to the argument STRING.

Ω

1 of 1

P. 105

Subject: Deallocating Function Results  
From: Kurt W. Hirschert

107(PROC/DATA)KWH-4 (Page 1 of 2)

### Problem

Section 6 says an allocatable function result variable is not deallocated on completion of the execution of the function, but nowhere does it say when this variable is deallocated.

### Discussion

We could approach this either of two ways:

1. We could find some place in the description of expression evaluation to describe when expression evaluation must also deallocated storage.
2. We can adopt the point of view that the result variable is deallocated just like any other local variable and that it is only its final value that is retained for subsequent expression evaluation, not the storage itself.

Since the latter approach seems to be more consistent with the treatment of nonallocatable result variables and also seems to involve easier text changes, it is the approach taken. Note that if we ever actually admit that an allocatable is really a pointer that is automatically being dereferenced, we may need to reconsider this descriptive decision.

### Proposal

1. Page 6-4, line 25. Delete "or function result".

~~2~~ Page 12-7, line 40. Replace "shape" with "rank".

15 Line 41. Add new sentence: "The shape is also so determined except in the case of a function whose result has been declared to be allocatable, in which case the shape is determined by the final shape of the result variable."

20 *completion of*  
3. Page 12-9, line 41. Add new sentences: "The value of the result variable at the ~~time~~ execution of the function ~~is completed~~ is the value returned by the function. If the function result has been declared to be allocatable, the shape of the value returned by the function is also determined by the shape of the result variable ~~at~~ *when* ~~the time~~ the execution of the function is completed. The value of the result variable must be defined by the function. If the function result has been declared allocatable, the result variable must be allocated by the function prior to being defined."

4. Page C-17, line 2+. Add new section: "

**C.12.x The Result Variable.** ~~Nominally~~ *similar to* the result variable is ~~a variable like~~ any other variable local to a function subprogram. Its existence begins when execution of the function is initiated and ends when execution of the function is

Subject: Deallocating Function Results  
From: Kurt W. Hirschert

107(PROC/DATA)KWH-4 (Page 2 of 2)

terminated. However, because the final value of this variable is used subsequently in the evaluation of the expression that invoked the function, an implementation may wish to defer releasing the storage occupied by that variable from termination of the function until after its value has been used in expression evaluation. ~~In the case of an allocatable function result, this deferral may include the explicitly allocated storage as well as the descriptor that locates it.~~

”

Ω

108

30  
30A

TO: X3J3

From: Lloyd Campbell

Subject: Suggested Edits to S8 dated June 1987

1. p. ii/2, delete "uses of".
2. p. ii/8, add "(predefined)" after "intrinsic" and delete "(prededined)" at p. iii line 30.
3. p. ii/20, delete "information".
4. p. ii/26, change "operations on" to "uses of" and delete "use".
5. p. ii/41, change "MODULE" to "module".
6. p. ii/47, change "use" to "their usage".
7. p. iii/1, change "BLOCK DATA" to "block data".
8. p. iii/31, change "REAL, including" to "REAL which includes".
9. p. iv/12, delete comma after "rules".
10. p. iv/21, change "intrinsic" to "intrinsic functions".
11. p. iv/37, add hyphen in "derived type".
12. p. 1-4/18, change "(e.g., array-spec)" to "(for example, array-spec in R513)". (106-62, #4)
13. p. 2-1/26, add "a" before "module" and add "an" before "internal". (to complete previous partial edit) Also include this change on copy of this constraint at 12-9/18+ & 12-10/26+.
14. p. 2-7/3, change "Upon invocation of a procedure" to "When a procedure is invoked".
15. p. 3-2/27, change "mapping" to "ASCII collating sequence as" and delete "(ASCII)". Also add sentence: "Note also that the intrinsic functions LGT, LGE, LLE, and LLT provide comparisons between strings based on the ASCII collating sequence."
16. p. 4-4/24, change "passed to" to "associated with".
17. ~~p. 4-7/38~~, add hyphen in "type parameter" ~~Also at 4-7/13, p. 7-8/42, 7-9/2,8,9.~~
18. p. 4-9/3, add "(7.5.1.4)" after "assignment".
19. p. 4-9/7, change 'last period to a colon.
20. p. 4-10/17, change "which" to "that".
- ~~21.~~ p. 5-1/4, add comma after "Collectively".
22. p. 5-3/12 & 14, add "type" before "parameters". (twice)
23. p. 5-3/20-22 & 25-26, add "In any context that requires type parameters to agree," to beginning of sentence and delete it at the end of sentence. (twice) (see 46-48)
24. p. 5-4/19, change "such a" to "the".
25. p. 5-4/45 & 46, add "type" after "range". (twice)
26. p. 5-7/40, delete "a" before "function" and before "local". (twice)
27. p. 5-8/18, delete sentence "If the lower bound is omitted, the default value is 1." (is covered in next sentence)
28. p. 5-9/34, delete "or SAVE statement". (add sentence below at 5-11/47+).
29. p. 5-10/13, add "are" before "equal".
30. p. 5-10/42, add "dummy" ~~before "argument"~~, after "specified".
31. p. 5-11/47+, add paragraph "A SAVE statement may appear in the specification part of a main program and has no effect." (see #28 above)
32. p. 6-3/18, make "effective shape" bold and change ". The" to "and the". (one sentence to define effective shape.)
33. p. 6-3/41, add hyphen in "explicit shape".

30A

34. p. 6-4/26, add "or" after the comma.
35. p. 7-1/3<sup>3</sup>, add "are" before "defined".
36. p. 7-6/41, add "if" after "or" and add "is" after "and".
37. p. 7-7/3 (twice), 5, 7, & 12 (twice), add space after the commas.
38. p. 7-8/28 & 34, change "the" to "a".
39. p. 7-8/30, delete hyphen in "exponent-range".
- ~~40.~~ p. 7-10/23, change "a subscript" to "an array element" and in line 24, change "the" to "its". (106-111 §13)
41. p. 7-10/26, delete "name".
- ~~42.~~ p. 8-1/11+, add "Constraint: An entry-stmt must not appear within a block."
- ~~43.~~ p. 8-5/1, change "fragments" to "constructs".
- ~~44.~~ p. 8-5/20+, in new text, change "code fragment" to "CASE construct".
- ~~45.~~ p. 8-7, in list item (3) of new text for 8.1.4.4.4, add "a" before "CYCLE" in "an EXIT statement or CYCLE statement".
- ~~46.~~ p. 8-8, in new text for 8.1.4.5, change "I/O" to "input/output" in text following Examples 2 and 7. (twice)
- ~~47.~~ p. 8-8, in new text for 8.1.4.5, delete "an" in "an ENDFILE or CONTINUE" in text following Example 7.
48. p. 8-10/3, delete "a" after "or".
49. p. 10-1/22-23, delete "or after" and add new item "(3) After a slash edit descriptor" and renumber old (3) to (4).  
(Allow FORMAT(3/I5))
50. p. 10-2/15, delete "a" before "control" and before "character".
51. p. 10-5/7, change "will be" to "are".
52. p. 10-14/29, delete "Acceptable".
53. p. 11-4/14, add colon after "statement".
54. p. 11-5/6, change "keyword" to "argument keywords".
55. p. 12-9/22, add "a" after "If". Also at 12-10/30.
56. p. 13-12/32, change "has the value" to "has a value". ("the" was incorrectly inserted by previous edit)
57. p. 13-21/33, change "has value" to "has a value".
58. p. 13-28/41, after "without" add "counting".
59. p. 14-9/9, change "equal sign" to "equals". (as in Sect. 3)
60. p. B-5/20, delete "see".
- ~~61.~~ p. c-4/42+, add section: "C.7.4 Overloaded Intrinsic Operators.  
The overloading of an intrinsic operator does not cause automatic overloading of any syntactical equivalent operator. For example, overloading .EQ. does not cause overloading of ==." (106-111 #10)
62. p. C-7/42, change "else undefined" to "else maximal record length".  
(106-62 #9 and 106-111 #31)
63. p. C-15/18, change "keyword arguments" to "argument keywords".
- ~~64.~~ p. C-16, in new text, change "machine's" to "processor's" and "machine" to "processor" in paragraph following first example of MODULE SAMPLE.
65. p. C-24/29, change "Examples" to "Example".
66. p. C-24/30, delete "C.13.9.1 Polynomials."
67. p. C-24/29, change "C.13.9." to "C.13.8.1".

TO: X3J3  
From: Lloyd Campbell  
Subject: Letter Ballot Edits for S8 dated June 87

The following suggested edits are from the last X3J3 letter ballot. The page and line numbers have been updated for the current S8. I believe that these edits are worthy of consideration and that most of them will improve the document.

272(2n) Page 2-8, line 8. Change 'entity' to 'object'. [Entities include expressions.] (JR)

292(2n) Page 2-8, line 40, ~~delete 'ranged arrays,'~~ and add sentence 'A ranged array has effective bounds for each dimension that are equal to or lie within the actual bounds; corresponding to the effective bounds, it has effective extents, an effective shape, and an effective size.' (JR)

~~350~~(2n) Page 3-4, line 14, replace "In free source form (3.3.1), a" with "When a statement is labeled using free source form (3.3.1), the". (JTM)

~~352~~(2n) Page 3-4, line 15, replace "In fixed source form (3.3.2), a" with "When a statement is labeled using fixed source form (3.3.2), the". See #350. (JTM)

~~354~~(1d) On page 3-4, line 16, change "1-5; blanks" to "1-5. Blanks" [blanks are once again insignificant, so may appear within labels in both source forms]. (Lm)

428(26) P4-3, L30, replace ~~expression "The" by "In any context which requires that the values of type parameters agree, the"~~ with "The" by ~~expression of the of the variable above in this method.~~ <sup>that</sup> "In any context which requires ~~that the values of type parameters~~ agree, the" (JLS)

435(3) P4-3, L33 & 37, replace "The" by <sup>that</sup> "In any context which requires ~~that the values of type parameters~~ agree, the" (JLS)

~~629~~(3d) P5-4, L25-26: <sup>Replace</sup> ~~What does~~ "or access such a definition" ~~mean either explicit or~~ ~~delete~~ with "or the interface to the function must be explicit". (JLS)

642(2n) Page 5-4 line 42, Change 'Objects' to 'Entities'. (JR)

643(2n) Page 5-5 line 5, Change 'object' to 'entity'. (JR)

~~714~~(2n) p. 5-8, line 41, change "dummy array argument" to "dummy argument array". (LC)

869(2g) [5-15/19] replace "the array" with "its array" <more specific>





~~1381~~ (3n) 8-5/1 : "fragments" --> "constructs"  
8-5/2 : Add to end of line: ! Construct 1  
8-5/8 : Add to end of line: ! Construct 2  
8-5/15 : Add to end of line: ! Construct 3 (BAM)

delete  
1396(3) Page 8-10, Lines 33-34.

This sort of text should not be part of a standard. (WGS-GS)

~~1519~~ (3y) 9-2/2 / 11+, edit Add: ~~"In particular~~ The number of files and units available, the allowed values for units and the form of file names, the number of records in a file and the maximum record size are all processor dependent."

~~1575~~ (3n) [10-13/3] replace "a name" with "an optionally qualified name"  
<avoid confusion on different use of the term "name" for namelist i/o>

1597 (3n) [10-14/39] before "list items" add "scalar" <go all the way>

~~1606~~ (3y) [10-15/18-10-15/41] delete & replace with copy of [10-11/45] <sup>+ 10-12/17 ] -</sup> with the following edit:  
(a) [10-12/2 (copy)] replace "comma, and slash" with "comma, slash, equals, and ampersand"  
(b) [10-12/4 (copy)] replace "comma, or slash," with "comma, slash, equals, or ampersand,"  
<parallel list-directed. Note that the "termination" (in [10-12/10 (copy)]) still applies; encountering "=" or "&" would be an error>

1706 (2n) [12-2/16] replace "a keyword argument" with "an argument keyword" <see 2:5.2>

~~1707~~ (3) Page 12-2, line 33 Is an explicit interface required if the function result is ranged? If so, after 'allocatable' in line 39, insert 'or ranged'.

1737 (3n) [12-5/27+] add line "providing its interface is explicit." <avoid confusion>

1808 (2n) p. 12-13, line 9: insert "dummy" following "first" (LR)

1818 (3n) p. 12-14, line 22: "and" -> ". A statement function"  
Comment: An internal procedure may not be supplied as a dummy argument. (LR)

1823 (3) p. 13-1, lines 13<sup>q</sup> 14: delete "external" <sup>(+ tunic)</sup> (LR) Also change "an" to "a".  
Comment: Why is this distinction made? What about passing it to an internal procedure or a module procedure?

1836 (3y) 13-2/35+ : edit Add: "The <sup>R</sup> EFFECTIVE\_RANGE and EFFECTIVE\_PRECISION functions each return a processor-dependent value that is greater than or equal to the declared value."  
↳ corresponding

~~1841~~ (3y) 13-2/47-48: edit We shouldn't use a phrase like "best fit". Replace the last sentence with "The models have processor-dependent parameters which describe the arithmetic used by the executing program."

~~1844~~(3y) 13-3/17 : edit Begin a new paragraph with "Examples" and add a new paragraph before it. "This standard only allows one model for integers and requires at least two models for real numbers. The only requirement on the real models is that one (default double) have a larger EFFECTIVE\_PRECISION than the other (default real)."

1860(3) p. 13-5, line 8: preface with the section title "13.7.8 Array Geometric Location Functions" (see 13.9.14) (LR)

1889(3) 13-17/2-3 : edit Replace "[E(1:DIM-1)...ARRAY." with "(d1,d2,...dDIM-1,dDIM,...dn) where (d1,d2,...dn) is the shape of ARRAY". This makes this text consistent with the text in other functions.

1912(3) [13-22/n before "numbers" add "positive" <avoid confusion>

1917(3) 13-23/25-26:edit Replace "[E(1:DIM-1)...ARRAY." with "(d1,d2,...dDIM-1,dDIM,...dn) where (d1,d2,...dn) is the shape of ARRAY". This makes this text consistent with the text in other functions.

1919(3) 13-23/30: edit Replace "[E(1:DIM-1),E(DIM+1:n)]" with "[d1,d2...dDIM-1,dDIM+1,...dn)".

~~1928~~(3) p. 13-29, line 29: append the sentence "It must not be an allocatable array that is not allocated or an alias array that is not alias associated." (LR)

~~1983~~(3) p. 13-25, line 7: following "scalar", insert "It must not be an allocatable array that is not allocated or an alias array that is not alias associated." (DH) (LR)

1936(3) 13 25/21 edit Replace "10" with ":". The use of DIM= is overly redundant for a one dimensional array.

1941(3) 13 26/7-8 : edit Rewrite the two sentences into one, eliminate the "Furthermore". Change "and" to ",". Change ". Furthermore," to ", and".

1974(3). Page 13-32 lines 34 & 42 : Before 'extent', insert 'effective'.

1985(3). Page 13-35 lines 9 & 17 Before 'extent', insert 'effective'.

1993(3y) 13-37/26, 'edit A better example is  
PACK (M, M.NE. 0, VECTOR = [2,4,6,8,10,12]) = [9,7,6,8,10,12].  
This shows where the elements come from.

~~2021~~(3y) [13-41/26] add sentence prior to first one in Case (i):  
"If BACK is absent or has the value .FALSE., STRING is processed from left to right."  
<say up front the usage of BACK>

~~2023~~(3y) [13-41/29] replace "The default" with "Note that the default" <goes with above edit>

~~2025~~(3y) [13-41/32] replace first sentence in Case (ii) with:  
"If BACK has the value .TRUE., STRING is processed from right to left."  
<parallel Case (i) above>

2033 (3) 13-43/23-24:edit Replace "E(1 n)" with "[d1,d2,...dn]" and "[E(...)]" with "[d1,d2,...dDIM-1,MAX(NCOPIES,0),dDIM,...dn]"

2035 (3) 13-43/29-30:edit Replace "SOURCE( ... omitted" with "SOURCE( r1,r2,...rDIM-1,rDIM+1,...rn+1)".

~~2106~~ (3) Page 14-3, line 21 After 'label.', insert 'Labels with the same or different digit sequences in the same or different scoping units cannot be associated.'

~~2107~~ (3) Page 14-3, line 23 After 'unit.', insert 'Exponent letters with the same or different letters in the same or different scoping units cannot be associated.'

~~2111~~ (2) Page 14-3, line 29 After 'rank.', insert 'Operators with the same or different letter sequences in the same or different scoping units cannot be associated.'

~~2115~~ (3) 14-5/15 edit Change to "deallocation of parent entity".

2147 (3) Page 14-5, line 1g The phrase:  
"Termination of execution of the executable program"  
should agree with page 14-4, line 35; and thus should read  
"Termination of execution of the ~~scoping unit~~  
procedure (Am) (JR)

2234 (2a) A-5/28, "nonstandard conforming syntax" --> "nonstandard syntax" <sup>become</sup> (Am)

2235 (2d) Page B-1, line 7, Change 'ANSI X3.9-1978' to 'FORTRAN ~~FORTRAN~~ 77'. (JR)

2237 (2a). Page B-1, lines 10-11, Change 'ANSI X3.9-1978' to 'FORTRAN ~~FORTRAN~~ 77'. (JR)

2238 (2a) Page B-1, line 11, After 'available' add 'in ~~FORTRAN~~ FORTRAN 77'. (JR)

2239 (2d) Page B-1, line 13 Change 'logical IF and block IF' to 'the IF statement (8.1.2.4) or IF construct'.

2252 (2a) Page B-1, line 21 After 'TO', insert 'statements'.

2264 (2a) Page B-2, line 9 After 'TO', insert 'Statements'.

2280 (2a) Page B-3, line 17, replace "Though the" with "The". line 18 replace "time, features" with "time. Features". (JRM)

2281 (2a) [B-3/20] replace "only provided" with "provided only" <usage>

2287 (3a) Page B-4, line 49, Add a paragraph:  
"It is understood that not all ENTRY statement usage is this simple; it is nevertheless believed that more complex uses of the ENTRY statement can be replaced through the straightforward use of internal procedures, the CASE construct, the IF construct, and if necessary, the unconditional GO TO construct." (Am)

~~2314~~ (3y) Page B-6, line 20+, Add a paragraph:

"There are, of course, other uses of the Computed GO TO that do not translate directly into a ~~SELECT~~ CASE construct (for instance, one cannot 'drop through' from one case to the next). It is believed, however, that straightforward use of the ~~SELECT~~ CASE construct, the IF construct, if necessary the unconditional GO TO construct, and perhaps internal procedures, instead of the Computed GO TO will result in a program code that is easier to write, easier to read, and easier to maintain."

(Am)

2321 (3y) Page C-2 line 6; replace "composed of intrinsic types and other" with "composed of components of intrinsic type and of other".  
consisting (JTM)

2326 (3y) Page C-2, lines 23+ Provide an example of the use of an EXPONENT\_LETTER statement. Add after line 23:

"For example, with the definitions:  
REAL (PRECISION = 10) B  
EXPONENT\_LETTER (PRECISION = 10) L  
The literal 10.93L7 has the same precision as the variable B.

(Am)

2342 (2n) Page C-4, line 15 Change "The component identified by the reference" to "A structure component".

2436 (2d) p. C-16, line 6: "contents of the" -> "contents of temporary storage back to the" (LR)

~~2441~~ (3n) Page C-17/2+ add the following:

'The CONTAINS statement is provided to delineate textually internal procedures from the host code. This is important primarily for error recovery, since a missing END statement in a file of multiple procedures might be difficult to isolate without CONTAINS.'

(RS)

2442 (2d) p. C-17, line 5: "extension" -> "new feature" (LR) (LC)

2446 (2n) p. C-17, line 13: "rectangular slices" -> "subarrays" (LR)

2474 (2n) Page C-20, line 38 Change "may be passed across" to "are specified".

TO: X3J3  
 FROM: Paul L. Sinclair  
 SUBJECT: Miscellenous Edits  
 DATE: November 17, 1987

~~34~~

342

The following edits to Draft S8, Version 104 are offered for consideration. References to the document are in the form: p/l or p/l-1 where p is a page number (for example, 4-5), and l is a line number. Material copied from the draft is in quotes. Items in apostrophes are suggested rewordings.

1. 9-2/9, add after "allowed forms," the following 'a processor-determined set of allowed actions,'.

Justification: needed for description of ACTION specifier in 9-8/18.

- ~~X~~ 9-7/24-25, delete the sentence "Note that SCRATCH must not be specified with a named file.".

Justification: 9-7/2 already says this. 9-7/1 is not repeated.

3. 9-7/35, change "If the FORM=" to 'If this'.

Justification: Consistency with the description of other specifiers (for example, 9-7/14).

4. 9-7/44, add after "access.", the sentences 'This specifier must be <sup>present</sup> ~~given~~ when a file is being connected for direct access. If this specifier is omitted when a file is being connected for sequential access, the default value is processor ~~determined,~~ <sup>dependent.</sup>

Justification: need to specify what happens when specifier is omitted, consistency with FORTRAN 77.

5. 9-8/5, change "If the BLANK=" to 'If this'.

Justification: Consistency with the description of other specifiers (for example, 9-7/14).

6. 9-8/18, add after last sentence:

'For an existing file, the specified action must be included in the set of allowed actions for the file. For a new file, the processor creates the file with a set of allowed actions that includes the specified action.'

Justification: This is stated for other specifiers. To be consistent, it should be stated here.

- ~~X~~ C-7/42, change "undefined" to 'maximal record length'.

Justification: Consistency with 9-21/1-5.

35

35a

TO: X3J3  
 FROM: Paul L. Sinclair  
 SUBJECT: Changes to Section 9  
 DATE: November 24, 1987

The following edits to Draft S8, Version 104 are offered for consideration. References to the document are in the form: p/1 or p/1-1 where p is a page number (for example, 4-5), and 1 is a line number. Material copied from the draft is in quotes. Items in apostrophes are suggested rewordings.

X 9-1/32, change "processor-dependent" to 'processor-determined'.

Justification: consistency with 9-2/8-10. Also, this is an attempt to reduce the number of terms: processor-determined => there exists for the processor an algorithm to determine value, processor-defined => there exists for the processor a specific value.

X 9-1/33, change "processor-dependent" to 'processor-defined'.

Justification: See above.

3. 9-1/34, delete "input/".

Justification: Only applies to output list.

4. 9-1/37-38, change "An endfile record is written explicitly by the ENDFILE statement. The file must be connected for sequential access." to 'An endfile record is written explicitly ~~to a file~~ by the ENDFILE statement; the file must be connected for sequential access.'.

Justification: Consistency with phrase "implicitly to a file" in following sentence. The sentences are closely related and should be separated by a semicolon.

X 9-2/12, change "processor dependent" to 'processor-determined'.

Justification: Consistency with 9-2/8-10.

X 9-2/28, change "to a unit (9.3.2)" to 'to a unit (9.3.2) or if the file is preconnected, when the file is created (9.3.4)'

Justification: Consistency with 9-6/13-15.

7. 9-3/23, change "the position of a" to 'the position of an external'.

Justification: Consistency with 9-2/23.

X 9-3/40-42, "On output, a new record is created and becomes the last record of the file." This does not explain where the file is positioned prior to data transfer. This would seem to imply that if a program writes successfully to a file, then rewinds the output file (which implicitly writes an endfile record) and then writes a record, the program

is non-standard conforming because an attempt was made to write a record after the endfile record.

~~X~~ 9-3/45-46, delete.

Justification: These lines only apply to sequential access and are redundant after moving lines 9-4/6-8.

10. 9-4/1, change "condition," to 'condition (9.4.3)'.

Justification: Provide reference to definition of term.

11. 9-4/2, change first "endfile record," to 'endfile record <sup>and</sup> ~~but~~ no error condition exists.'

Justification: Completeness and consistency with 9-4/9 and 9-12/30-32.

12. 9-4/3, change first "condition," to 'condition (9.4.3)'.

Justification: Provide reference to definition of term.

~~X~~ 9-4/6-8, move these lines to follow immediately after 9-3/40-42 (as part of the same paragraph).

Justification: These lines only apply to sequential access and should be part of that paragraph and they apply prior to data transfer, not after.

~~X~~ 9-5/18, change "processor-dependent external unit that is preconnected for" to 'processor-defined unit that is preconnected to an external file for'.

~~X~~ 9-5/26, change "to a file" to 'to an external file'.

16. 9-7/3, change "FILE =" to 'FILE='.

17. 9-7/13, add 'Any trailing blanks are ignored.' after sentence ending with "unit."

Justification: 9-7/4-5 states this for other specifiers but this should also apply to FILE= according to FORTRAN 77.

~~X~~ 9-7/16-17, change "processor dependent" to 'processor-defined'.

~~X~~ 9-7/26, change "processor dependent" to 'processor-determined'.

~~X~~ 9-7/44-45, change "the number of characters" to 'measured in character storage units'.

Justification: Consistency with following sentence.

~~X~~ 9-7/46, change "dependent" to 'defined'.

Justification: consistency with 9-21/4.



22. After 9-8/35-36, move 9-9/7-21.

Justification: Consistency with format of OPEN statement on 9-6.

23. 9-8/46-47, change "A given" to 'Each', "the unit specifier must appear." to 'an external-file-unit must be specified.' Also, move these two lines before preceding constraint.

Justification: Consistency with 9-6/45-48.

24. 9-9/1-2, move this paragraph after following paragraph (9-9/3-6).

Justification: Consistency with format of OPEN statement on 9-7.

25. 9-10/24, delete "or print-stmt".

Justification: Redundant since syntax already does not allow any specifiers in a PRINT statement.

26. After 9-10/43, add paragraph

'Constraint: If ~~the~~ REC= specifier is present, an END= specifier must not appear and ~~format~~ must not be \*.'

Justification: This should be a constraint.

- ~~X~~ 9-11/22-23, delete.

Justification: Redundant since discussed in Section 10.1.1 which is referred to in previous paragraph (9-11/20-21).

28. 9-11/24-25, delete "and a REC= specifier must not be present".

Justification: This should be a constraint.

29. 9-11/34-35, delete "and ~~an~~ END= specifier must not be present".

Justification: This should be a constraint.

- ~~X~~ 9-12/1, change "processor-dependent" to 'processor-determined'.

- ~~X~~ 9-12/3, change "processor-dependent" to 'processor-determined'.

- ~~X~~ 9-12/27, change "processor-dependent" to 'processor-determined'.

- ~~X~~ 9-12/28-29, delete "The labeled statement must be in the same scoping unit as the input/output statement."

Justification: This is already a constraint and is redundant here.

34. After 9-12/33, insert '(2) <sup>If</sup> the file specified in the input statement is positioned after the endfile record.'

<sup>↑</sup> an external file, it is

Justification: Consistency with 9-12/23-29.

35. 9-12/34, change "(2)" to '(3)'.

~~36.~~ 9-12/35, change "processor-dependent" to 'processor-determined'.

37. 9-12/36, change "(3)" to '(4)'.

~~38.~~ 9-12/36-37, delete "The labeled statement must be in the same scoping unit as the input/output statement."

Justification: This is already a constraint and is redundant here.

~~39.~~ 9-12/38, delete.

Justification: This is already a constraint and is redundant here.

~~40.~~ 9-14/3-4, change "processor dependent" to 'proprocessor-defined'.

41. 9-14/7, delete "In this case, the file is positioned after the endfile record."

Justification: This is redundant and not applicable here.

~~42.~~ 9-14/12-30, these lines should be merged with 9-12/21-38.

Justification: Separate definition of condition from actions which occur because of condition. Currently actions are discussed in two places.

~~43.~~ 9-14/13-14, delete "If an error condition occurs during execution of an input/output statement, the position of the file becomes indeterminate."

Justification: This is redundant and not applicable here.

44. 9-15/8, delete "file"..

~~45.~~ 9-15/19, change "specifier is" to 'specifier, if any, is'.

~~46.~~ 9-15/42, change "specification." to 'specification, if any.'.

47. 9-16/14, change "agree with the type of" to 'be of the same type and have the same type parameter values as'. Also delete first "the type of".

48. 9-16/17, change "agree with the length of" to 'have the same length as'.

49. 9-17/35, delete. *Also delete first "the length of" and in line 16, change "If" to "Note that if".*

Justification: This is already implied by syntax and definition of external file unit.

50. After 9-18/4, insert paragraph 'The IOSTAT= and ERR= specifiers are described in Sections 9.4.1.5 and 9.4.1.6, respectively.'

51. 9-19/26, change ~~"An" to "An inquire by file or inquire by unit form of the"~~

*↑ "INQUIRE statement" to "inquire-spec-list",*

52. After 9-19/36, insert paragraph 'The IOSTAT= and ERR= specifiers are described in Sections 9.4.1.5 and 9.4.1.6, respectively.'

~~53.~~ 9-19/41, change "processor dependent" to 'processor-defined'.

~~54.~~ 9-20/20, change "processor dependent" to 'processor-defined'.

~~55.~~ 9-21/2, change "maximal record length of the file" to 'length of each record in the file if the file is connected for direct access and is assigned the value of the maximal length of a record in the file if the file is connected for sequential access'.

Justification: Consistency with wording in OPEN statement.

~~56.~~ 9-21/3, change "the number of characters" to 'measured in character storage units'.

Justification: Consistency with following sentence.

~~57.~~ 9-21/5, change "the file does not exist" to 'there is no connection'.

Justification: Need to explain what happens when there is no connection. When there is a connection, value returned is defined whether or not file exists.

~~58.~~ After 9-21/27, add paragraphs

9.6.1.18 READ= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the READ= specifier is assigned the value YES if READ is included in the set of allowed actions for the file, NO if READ is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READ is included in the set of allowed actions for the file.

9.6.1.19 WRITE= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the WRITE= specifier is assigned the value YES if WRITE is included in the set of allowed action for the file, NO if WRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not WRITE is included in the set of allowed actions for the file.

9.6.1.20 READ\_WRITE= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the READ\_WRITE= specifier is assigned the value YES if READ/WRITE is included in the set of allowed actions for the file, NO if READ/WRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READ/WRITE is included in the set of allowed actions for the file.'

Also, renumber following sections appropriately and add these specifiers to list on page 9-19.

Justification: Consistency with other inquire specifiers.

~~59.~~ 9-21/40, change "processor-dependent" to 'processor-determined'.

# MARKED UP VERSION

107(20) RAH-1  
Page 1 of 4

36A

TO: X3J3  
FROM: DICK HENDRICKSON  
SUBJECT: REWRITE OF 106(\*) RAH-2 ON SOURCE FORM

This is a rewrite of sections 3.2.5 and 3.3, which describe source form. I have attempted to describe what is common between the 2 source forms and yet separate the descriptions.

I believe the only substantial changes from the 106 version are the changes from 2640 characters to 19 continuation lines in 3.3.1.4 and the addition of the last sentence in 3.3.1.3. I believe the first change is in accord with the straw votes at meeting 106. The single "&" restriction removes and ambiguity. Is the "&" the first character on the line, which means that the statement continues at the next character position (which happens to be only blanks); or is it the last, which means that this line (which happens to only contain blanks) is to be continued? We could define it either way. I think disallowing ambiguities is generally better than picking either one.

As an open question should we include a restriction about END statements in free form source similar to the FORTRAN 77 one in fixed form? Whatever the reason for the restriction in fixed form shouldn't it also apply to free form? It would make the language consistent and teachable.

There was some discussion at meeting 106 about limiting the length of lines to 1 to 132 characters, rather than 0 to 132. I didn't make this part of the proposal because I believe it would make the description of ";" awkward. I think automatic program generators might create zero length lines and people sometimes hit an extra carriage return by mistake.

**Proposal:** Replace lines 10 thru 41 on page 3-4 and all of page 3-5 with the following.

**3.2.5 Statement Labels.** <sup>A S</sup> ~~Statement labels~~ provide <sup>S</sup> a means of referring to <sup>A</sup> individual statements. Any statement not forming part of another statement may be labeled.

R324 *label* is *digit*[*digit* [*digit* [*digit* [*digit*]]]]

Constraint: At least one digit in a *label* must be nonzero.

If a statement is labeled, the statement must contain a nonblank character. The same statement label must not be given to more than one statement in a scoping unit. Blanks and leading zeros are not significant in distinguishing between statement labels. Blanks may appear anywhere within a label. For example:

(157)

THAT

**3.3.1.3 Free Form Statement Continuation.** The character "&" is used to indicate that the current statement is continued on the next line ~~which~~ is not a comment line. Comment lines cannot be continued; an "&" in a comment has no effect. Comments may occur within a continued statement. When used for continuation, the "&" is not part of the statement. No line may contain only a single "&" as the only nonblank character.

THAT

**3.3.1.3.1 Noncharacter Context Continuation.** If an "&" is the last nonblank character on a line or the last nonblank character before an "!", the statement is continued on the next line ~~which~~ is not a comment line. If the first nonblank character on the next line is an "&", the statement continues at the next character position following the "&"; otherwise, it continues with the first character position of the next line.

NON COMMENT

**3.3.1.3.2 Character Context Continuation.** If a character context is to be continued, the "&" must be the last nonblank character on the line and an "&" must be the first nonblank character on the next line ~~which~~ is not a comment line and the statement continues with the next character following the "&". The "&" signifying continuation cannot be followed by commentary.

T/

**3.3.1.4 Free Form Statements.** - A label may precede any statement not forming part of another statement. Note that no Fortran statement begins with a digit. A statement must not have more than ~~X~~ continuation lines.

39

**3.3.2 Fixed Source Form.** In fixed source form, each line must contain exactly 72 characters and there are restrictions on where a statement may appear within a line.

**3.3.2.1 Fixed Form Commentary.** The character "!" initiates a comment except when it appears within a character context or in character position 6. The comment extends to the end of the line. Lines beginning with a "C" or "\*" in character position 1 and lines containing only blanks are also comments. Comments may appear anywhere within a program unit, ~~X~~ may precede the first statement of the program unit, and have no effect on the interpretation of the program unit.

**3.3.2.2 Fixed Form Statement Separation.** The character ";" separates statements, or partial statements, on a single source line except when it appears in a character context or in a comment. If the sequence delimited by one or more ";" separators contains no characters or only blank characters, the sequence is ignored.

IF THE FIRST NONBLANK CHARACTER ON A LINE IS AN "!", THE LINE IS A COMMENT LINE.

43

107-CDB-4  
February 1, 1988

**From : Carl Burch**

**To : X3J3**

**Subj : Subgroup Nominations for Public Review Letters 22-47**

Please find attached the annotated copies of Public Review letters 22-47, marked with my recommendations for subgroup assignments. All subgroup assignments are negotiable between the Subgroup Chair and the Public Review Working Group. Any omissions noted should be brought to the attention of the Public Review Working Group.



P. 178



THE UNIVERSITY OF TEXAS AT DALLAS

Center for Applied Optics  
13920 Maham Road, #3016  
Dallas, Texas 75240

November 19, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

22-1

22-2

22-3

I think that the array triplet notation is a timely extension. I also believe that the ability to allocate arrays is a timely extension. I think that the MODULE/USE feature is an unneeded change. I am concerned that the DO WHILE statement was not in the draft. I feel that the COMMON statement must not be deleted from any FORTRAN of the future. I also feel that the DIMENSION statement must not be deleted from a future FORTRAN standard.

22-4

22-5

Too many new features have been added. ← 22-6

Sincerely,

Danny Chu

Subgroup Nominations:

- 22-1 GEN
- 22-2 DATA
- 22-3 PROC
- 22-4 CIO
- 22-5 GEN
- 22-6 GEN
- 22-7 DATA
- 22-8 GEN

87 DEC 28 P2:04

X3



THE UNIVERSITY OF TEXAS AT DALLAS

Center for Applied Optics  
13920 Maham Rd., #3016  
Dallas, Texas 75080

November 19, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I think that the array triplet notation is a timely enhancement. I also believe that the ability to allocate arrays is an useful improvement. I think that the MODULE/USE feature is a needlessly complex extension. I am concerned that the POINTER data type was missing. I am also unhappy that the DO WHILE statement was not defined. I think that the DIMENSION statement cannot be removed under any circumstances. I also think that the alternate RETURN statement cannot be taken out of from any FORTRAN of the future.

22-3, cont.  
22-7  
22-4, cont.

22-1, continued

22-2, c

22-5

FORTRAN should remain an easy-to-learn language

22-8

Sincerely,

Danny Chu

87 OCT 28 P2:03

X3





THE UNIVERSITY OF TEXAS AT DALLAS

Center for Applied Optics  
13920 Maham Rd., #3018  
Dallas, Texas 75240

November 19, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

22-1, cont.

22-2, cont.

22-3, cont.

I believe that the array triplet notation is an useful enhancement. I also think that the ability to allocate arrays is a valuable enhancement. I think that the MODULE/USE feature is an unneeded modification to the language. I am unhappy that the DO WHILE statement was not in the draft. I think that the COMMON statement cannot be taken out of from any FORTRAN of the future. I also think that the DOUBLE PRECISION statement cannot be removed from a future FORTRAN standard.

22-4

22-5

FORTRAN should remain an easy-to-learn language

22-8, cont.

Sincerely,

Danny Chu *Danny Chu*

07 DEC 28 P2:04

X3



THE UNIVERSITY OF TEXAS AT DALLAS

Center for Applied Optics  
13920 Maham Road, #3018  
Dallas, Texas 75240

November 19, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I believe that the array triplet notation is an useful addition. I also feel that the ability to allocate arrays is a valuable addition. I think that the MODULE/USE feature is an inefficient change to FORTRAN. I am shocked that the DO WHILE statement was not in the draft. I think that the COMMON statement cannot be removed ever. I also think that the DIMENSION statement cannot be removed under any circumstances.

22-2, conf  
22-3, conf  
22-4, conf

22-1, conf

22-5

Sincerely,

Danny Chu

*Danny Chu*

87 DEC 28 P2:04

X3

DECUS Meeting  
Dallas, Texas X3 SECRETARIAT

December 15, 1987  
'87 DEC 22 AM 1:57

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. **WE NEED A "BIT" DATA TYPE** (23-1) (23-2)
- (23-3) → **AND A SIMPLE STRUCTURED MECHANISM.**
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. **INCLUDE IS MORE APPROPRIATE.** (23-4) (23-5)
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. (23-6)
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. (23-7)

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

Chris Jahn CHRIS JAHN  
 Company: SOFTWARE LEXCOGRAPHY  
 Address: 5810 PRESTON VIEW #2076  
DALLAS, TX 75240

Subgroup	Nominations:
23-1	DATA
23-2	DATA
23-3	DATA
23-4	PROC
23-5	GEN
23-6	DATA
23-7	GEN

DECUS Meeting  
Dallas, Texas  
December 15, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. 24-1
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. 24-2
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. 24-3
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. }

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

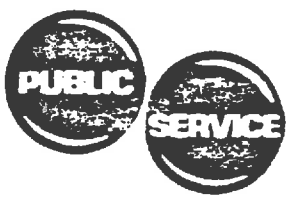
Walter Bantier

Company: SYNTECH  
Address: 8401 Skillman #1105  
Dallas, TX 75231

Subgroup	Nominations:
24-1	DATA
24-2	PROC
24-3	DATA
24-4	GEN

#25

43



**WISCONSIN PUBLIC SERVICE CORPORATION**

December 14, 1987

Public Comment for Dpans Fortran Revision  
X3 Secretariat  
Attn: Gwendy Phillips  
Computer and Business Equipment Manufacturers Association  
Suite 500  
311 First Street, NW  
Washington, DC 20001-2178

Public Comment for Dpans Fortran Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY 10018

87 DEC 22 AM 5:7

X3 C

Gentlemen and Ladies:

Fortran 8X

Since IFM's use of Fortran is significant within GPG applications, I must concur with IBM's vote of No.

It is evident that the developers of GPG have been polled and have voted against 8X. Therefore, WPSC's interfaces with this product must be compatible.

Sincerely,

*Patrick T. Christofferson*

Patrick T. Christofferson  
Systems Development Projects Supervisor  
Information Services Department

Subgroup Nomination:

ks

P. 185

43

#26

Earl Z. Damewood, Ph.D.  
INCO Alloys Int'l. Inc.  
P.O. Box 1958  
Huntington, WV 25720  
December 16, 1987

Public Comment for Dpans Fortran  
Revision - X3 Secretariat  
Attn: Gwendy Phillips  
Computer and Business Equip. Manuf. Assoc.  
Suite 500  
311 First Street NW  
Washington, DC 20001-2178

Dear Ms. Phillips:

In my opinion the suggested 8X Revisions to Fortran 77 would overly complicate the language and detract from its current simplicity, usefulness and elegance.

26-1

A much more constructive thing to do, especially by a particular vendor, namely IBM, would be to make the existing Fortran 77 CICS compatible for on-line use. Only being able to run under TSO or VM severely limits its use and integration with other languages and systems.

} Comment not pertinent to F8X1

Sincerely,

*Earl Z. Damewood*

Earl Z. Damewood, Ph.D.  
Chief Systems Engineer

Subgroup Nonin GEI

- cc: J. C. Coe - Inco
- R. P. Lett - Inco
- R. C. Moore - Inco
- American National Standards Institute - New York
- IBM General Products Division - San Jose, CA
- IBM Local Support Office - Huntington, WV

43

#27



Département de Physico-Chimie

Laboratoire de Réactivité et Mécanismes  
en Chimie Inorganique

Associé au CNRS UA 331

Service de Chimie Moléculaire

N/Réf. : DPC/SCM 87 - 618

Ms Jeanne ADAMS, Chair  
X3 - Information Processing Systems  
American National Standards Institut  
Scientific Computing Division  
NCAR  
P.O. Box 3000  
Boulder Colorado 80307

Saclay, December 8, 1987.

Dear Ms. Adams,

Among the new intrinsic functions provided in the "FORTRAN 8X" project (document X3J3/S8.104 of June 1987) appears the EPSILON Function (page 13-7, line 5).

The definition given for that function, namely : "number that is almost negligible compared to one" lacks accuracy, and should be replaced by the following : "the smallest positive number  $\epsilon$  representable into the processor such that, for the given processor,  $1 + \epsilon$  be strictly greater than 1. This number is equal to  $b^{1-m}$  where  $b$  is the base of internal representation of real numbers (function RADIX, line 9), and  $m$  the number of digits of the mantissa (function DIGITS, line 2)".

Besides, it should be explicitly stated that whenever "temporary real numbers" are used, as is the case when an arithmetic coprocessor is active, the functions DIGITS and EPSILON return the values corresponding to "permanent real numbers" and not to "temporary real numbers".

The enclosed article, submitted to "Communications of the ACM", provides the necessary explanation and comments for the need of the above sentence.

Yours sincerely

*Edgar Soulié*

Dr. Edgar Soulié

Subgroup Name  
27-1 DATI  
27-2 DATI

P.J. : article entitled "The machine precision  
in the presence of an arithmetic coprocessor"

P. 187

THE MACHINE PRECISION IN THE PRESENCE OF AN ARITHMETIC COPROCESSOR

Edgar Soulié  
IRDI/DESICP/DPC/SCM et UA 331 du CNRS  
CEA CEN/SACLAY  
91191 GIF SUR YVETTE CEDEX FRANCE

The machine precision is defined as the smallest positive number  $\epsilon$  representable into a processor ("the machine") such that, for the given processor,  $1 + \epsilon$  is strictly greater than 1. This number depends on the base  $b$  chosen to represent real numbers into the machine, and on the number  $m$  of digits in base  $b$  which constitute the mantissa of the representation :  $\epsilon = b^{1-m}$ .

An algorithm published by Malcolm (1) enables the determination of both  $b$  and  $m$ , and thus of  $\epsilon$ .

A recent standard published by IEEE (2) specifies that the basis  $b$  should be equal to 2, and that the number of digits in the mantissa should be at least 24 in single precision, and 53 in double precision.

When no arithmetic coprocessor is present or active, the above values have resulted from the execution of Malcolm's algorithm on an IBM PC-XT, several "PC compatible" machines as well as a SUN 3/160. These machines therefore comply with the corresponding requirements of the IEEE standard.

When an arithmetic coprocessor is active however, the number of digits returned by Malcolm's algorithm is 64, both in single and double precision. This surprising result must be ascribed to the use of "temporary real numbers" in the arithmetic coprocessor (see for example ref. 3). The latter numbers each occupy 80 bits, among which 64 store the mantissa. The transfer of a number from the arithmetic coprocessor to the central processing unit results in a loss of precision by elimination of those binary digits beyond the 24<sup>th</sup> (or the 53<sup>rd</sup> in double precision).

The actual precision of calculations is therefore determined by the representation of the "permanent real numbers" and not of the temporary ones. In other words, Malcolm's algorithm is deceived when an arithmetic coprocessor is active, with the consequence that the machine precision  $\epsilon$  cannot be determined automatically by the execution of the algorithm.

Programs that use machine precision in order to determine at which iteration to stop an iterative calculation such as the resolution of an equation, the minimization of a function, etc., may not be kept portable (1)



reason.

Since the project of a new standard (5) for the FORTRAN language entails numeric intrinsic functions returning the values of  $b$ ,  $m$  and  $\epsilon$ , it should be specified in this standard that the values of  $m$  and  $\epsilon$  returned by these intrinsic functions pertain to the permanent real numbers, and not to the temporary real numbers.

A similar statement should be added in the standard of any programming language providing with the values of either  $m$  or  $\epsilon$ .

Thanks

Thanks are expressed to Dr. Gérard Langlet for fruitful discussions.

## References

- (1) Michael Malcolm, An algorithm to reveal the properties of floating point arithmetic, *Communications of the ACM*, 15, 949 (1972)
- (2) IEEE Standard for binary floating-point arithmetic, standard ANSI/IEEE 754-1985, the Institute of Electrical and Electronics Engineers, New-York, July 1985.
- (3) iAPX 86/88, 186/188 User's Manual-Programmers's Reference INTEL Corporation, Literature Department, 3065 Bowers Avenue SANTA CLARA California 95051, May 1983 - table 6-2 page 6-5.
- (4) J. Larmouth, Fortran 77 portability, *Software-practice and experience*, 11, 1071 (1981)  
Edgar Soulié, Quelques caractéristiques de bons programmes, *Journal de Chimie Physique et de Physico-Chimie Biologique*, 80, 397 (1983).
- (5) Programming Language FORTRAN, American National Standards Institute, Inc., Draft S8, version 104, June 1987. page 13-7.

M. Claude Bourstin  
Secrétaire de AFNOR/CT97/SC22/GT5  
AFNOR Division Informatique  
Tour Europe Cédex 7  
92080 PARIS 1a DEFENSE

Monsieur Chavy  
Directeur de l'Informatique  
Commissariat à l'Energie Atomique  
29, rue de la Fédération  
75015 PARIS

Mr. J.L. Côté  
ISO/TC98/SC22 Secretariat  
Treasury Board of Canada  
140 o'Connor street  
10<sup>th</sup> Floor L'Esplanade Laurier  
OTTAWA Ontario KIA OR5  
Canada

M. François Genuys  
Président de AFNOR/CT97/SC22  
IBM  
36, av. R. Poincaré  
75016 PARIS

Ms. Jeanne T. Martin  
Chair of ISO/TC97/SC22/WG5  
Lawrence Livermore Laboratory  
P.O. Box 808, L-300  
LIVERMORE California 94550  
Etats-Unis

M. Christian Mas  
Animateur de AFNOR/CT97/SC22/GT5  
IBM - Service 3580  
Tour Septentrion Cédex 9  
92081 PARIS 1a DEFENSE

P. 191

#28

December 16, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

28-1

I am against adoption of the proposed FORTRAN 8x standard because it is overly complex and deviates too much from the FORTRAN implementations currently in use. ← 28-2

FORTRAN is popular in the scientific community because it is easy to use and widely available. Ease of use is essential to us because we use the computer as a tool; our main interest is not in the abstract art of computer science.

28-3

The proposed standard is not a refinement or specification of our existing tool, but a substantially redesigned tool which is both foreign to us and much more difficult to learn. If some of you really want a language that looks like that, please don't call it FORTRAN.

Proposed capabilities such as array operations could much more elegantly be provided within the existing language by function and subroutine libraries. At the same time, the proposed standard omits truly useful constructs such as bit arrays.

28-4  
28-5

I also strongly oppose the attempt to do away with COMMON, DIMENSION, and EQUIVALENCE, which are deeply imbedded in virtually every existing FORTRAN program. Adoption of the proposed standard would eventually prevent recompilation of our existing programs, which we have neither the time nor desire to rewrite.

28-6

Keep FORTRAN simple. Just vote "NO".

Sincerely,

*Jesse W. Black II*

Jesse Black  
Suite 1200  
One Lincoln Centre /LB25  
5400 LBJ Freeway  
Dallas, Texas 75240

Subgroup	Nominations:
28-1	GEN
28-2	GEN
28-3	GEN
28-4	GEN
28-5	DATA
28-6	GEN

# CEBMA Catalyst

An Information Technology Firm of Peat Marwick

The Catalyst Group  
Peat Marwick Main & Co Telephone 312 938 1000  
303 East Wacker Drive  
Chicago, IL 60601

#29

43

December 14, 1987

X3 Secretariat  
CEBMA  
Attn: X3J3 Public Comment  
311 First Street NW  
Washington, D.C. 20001

87 DEC 21 P 1:44

X3

Dear Sirs,

The following comments concerning the proposed X3J3 (FORTRAN 8X) standard are submitted as my personal opinions and may not represent the opinions of my employer. My concerns with this proposed standard are all in the area of size and support. They include:

- 1) The "NO" vote by representatives of three major vendors is a major concern, because it may indicate a long period before conforming compilers are available. } 29-1
- 2) The proposed standard is more than a standardization of existing industry practice. ← 29-2
- 3) The FORTRAN language will become too large. ← 29-3
- 4) The FORTRAN standards committee is attempting to make the language all things to all people - losing the benefits of small, well focused languages. } 29-4

The reasons for each of my concerns covers largely the same ground - the size of the resulting FORTRAN 8X language.

With a small language, a programmer is expected to be familiar with all of its features. As a language grows, s/he can be comfortably familiar with a smaller and smaller percentage. Then, while there might be a particular construct ideal for a situation, the programmer is unfamiliar with it and does not use it - negating the benefit of having it. } 29-3, c

For many years, the public expression of algorithms occurred in ALGOL - because it had a minimal but sufficient control syntax. This allowed the details of the algorithm to be clearly expressed without depending on the features available in a particular language. The reason algorithms were not expressed in FORTRAN, COBOL, or APL is that each of these hid what was being described. FORTRAN and COBOL did not possess full control structures while APL was difficult to read for one not versed in it.

P. 193

**EC** Catalyst

X3J3 Public Comment  
December 14, 1967  
Page 2

Today, PASCAL is commonly used for the same reasons. ADA<sup>1</sup> is not the language of choice, because its richness limits its usefulness to those who use it frequently. PASCAL is chosen because almost anyone (even including COBOL programmers) can read it.

It is impossible to utilize a new standard until compilers are available. Regardless of a standards technical merit, it ultimately succeeds or fails depending upon whether it is used.

A vendor might have two basic reasons for not approving a standard - technical flaws or insufficient resources to develop a new compiler. The former will, hopefully, be revealed during the public comment period. If, however, major vendors do not support a standard then, at best, development of new compilers occurs slowly and receives limited resources. This can be overcome by public demand - if the public wants the new features.

} 29-5

Item by item the proposed additions may be useful - but most of them have not been "battle tested" nor demanded by large numbers of practitioners. Placing them in the standard today is a long term commitment and significant investment. I am unwilling to make this investment unless I am sure there will be an adequate return.

} 29-6

The identified function of the standards committees is to standardize extensions to the language found in industry. In the FORTRAN 8X standard (as with COBOL 85), the proposed standard has gone beyond this. I am afraid that, as with COBOL 85, this will lead to a long span of time before it is fully adopted by industry.

} 29-2 cont.

I recognize and appreciate the effort the committee has expended in drafting the proposed standard. Until the number of additions are reduced and major vendors publicly commit to supporting the new language, I am unable to support its adoption as a standard.

Very truly yours,

*Burton M Strauss III*

Burton M. Strauss III  
Senior Consultant  
Peat Marwick Main & Co.

Subgroup      Nominations:  
29-1      GEN  
29-2      GEN  
29-3      GEN  
29-4      GEN  
29-5      GEN  
29-6      GEN  
~~29-7~~



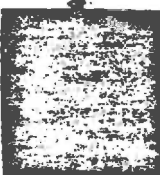
The University of Liverpool

With the Compliments of

J.L. Schonfelder, M.Sc., D.I.C., Ph.D., AFIMA, MBCS  
Director of the Computer Laboratory

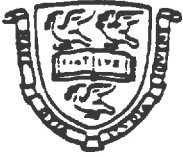
Centre for Computer Studies,  
Chadwick Building,  
P.O. Box 147,  
Liverpool L69 3BX.  
Tel: 051 - 709 - 6022 Ext. 2954  
Email: JLS @ UK.AC.LIV.IBM

<u>Subgroup</u>	<u>Nominations :</u>
30-1 DATA	30-16 GEN
30-2 GEN	30-17 GEN
30-3 DATA	30-18 CIO
30-4 CIO	30-19 CIO
30-5 GEN	30-20 PROC
30-6 PROC	30-21 PROC
30-7 DATA	30-22 PROC
30-8 CIO	30-23 DATA
30-9 DATA	30-24 PROC
30-10 GEN	30-25 GEN
30-11 DATA	30-26 GEN
30-12 CIO	30-27 DATA
30-13 GEN	30-28 PROC
30-14 DATA	30-29 DATA
30-15 GEN	30-30 PROC
	30-31 CIO



43

#30



# The University of Liverpool

PROFESSOR L.M. DELVES  
DIRECTOR

CENTRE FOR MATHEMATICAL SOFTWARE RESEARCH  
VICTORIA BUILDING BROWNLOW HILL P.O. BOX 147 LIVERPOOL L69 3BX

TEL: 051 - 709 - 6022 EXT. 2018  
TELEX NO: 627095 UNILPL G

To: X3J3

9 December 1987

Comment on S8.104: Fortran 8X draft standard

I broadly welcome the new standard, and believe that the additional functionality which it includes will be welcomed and used by most current Fortran 77 sites. I believe that X3J3 has done well in keeping the language reasonably tidy while maintaining compatibility with F77. I have however the following detailed comments and recommendations, in order of priority:

## 1) Pointers

- 1.1) The lack of a pointer facility is a glaring omission which must be repaired before the standard is finalised. Pointers are widely used in scientific programming to provide dynamic and irregular storage allocation and management. They can also provide space and time efficient solutions to data sharing problems, by obviating the need for copying. The clumsiness of F77 codes simulating what should be done with pointers is well known; and all major languages (e.g. Ada, Pascal, Algol 68, C) now include a pointer facility.
- 1.2) Pointers should be fully typed, and pointers to any object should be available. My own preference would be for a pointer attribute to be introduced, but I would be willing to go along with (almost) any syntax.
- 1.3) Multi-level pointers (up to at least "pointers to pointers") are useful in practice, but simple level pointers would do at a push.
- 1.4) The introduction of a Pointer facility should be matched by the deletion of the IDENTIFY and SET RANGE facilities, whose functionality is subsumed by pointers with considerable potential increase in language regularity.

IDENTIFY : half of a pointer facility

SET RANGE : is a recipe for opaque coding.

Its effect can be achieved by pointing to a slice of the array whose range is being set. The number of occasions on which extensive multiple range assignments are needed (when SET RANGE would save a few repetitious statements) is small, and will become smaller as users get used to dynamic storage allocation and slicing.

## 2) ALLOCATE/DE ALLOCATE

p. 196

30

30



3) Exception Handling

I am sorry to see no exception handling mechanism in the published draft. That in Ada is proving to be very useful; a facility should be easy to add to F8X without disturbing the rest of the language, and indeed I am aware that X3J3 has made proposals in the past. Let me urge you to put exception handling back in.

} 30-4

4) What a pity that:

X3J3 felt it impossible to include:

4.1) Multi-tasking facilities: MIMD machines are already common, and will be increasingly so. A decision that it was premature to establish a standard for multi-tasking probably looked safe in 1980; it looks less safe now, and F8X will now be instantly non-portable as soon as it hits the market.

} 30-

4.2) Generic facilities (à la Ada). You have modules, type definitions, procedure and operator overloading and a degree of type genericism in the parametrised reals. It is quite a small step to allowing generic coding with types left undefined in the source code, but instantiated at run time (Ada ensures that "run time" can always be, at the latest, "link time").

} 30

It is too late to include 4.1) or 4.2) now - too much delay would, in my opinion, ensue. A pity, though.

*L. M. Delves*

L. M. Delves



43

FROM THE DIRECTOR OF THE COMPUTER LABORATORY  
J.L. SCHONFELDER, M.SC., D.I.C., PH.D., AFIMA

CHADWICK BUILDING P.O. BOX 147 LIVERPOOL L69 3BX

TEL: 051 - 709 - 6022  
TELEX NO: 627095

## The University of Liverpool

JLS/MJ/2954

18th November, 1987

TO: X3 Secretariat/CBEMA  
Fortran Public Review  
311 First St, N.W.  
Suite 500  
Washington D.C.  
20001-2178  
U.S.A.

RE: The Proposed Revision of the Fortran Standard

### 1. GENERAL COMMENTS

The language described in the published draft includes on the whole a welcome and much needed enhancement to the expressive power of Fortran. It also provides a framework which will allow an orderly evolution of the language. This is a piece of linguistic infrastructure that is essential for a language which has a lifetime longer than a few years and an applications base larger than a small number of academic users. It is also entirely commendable that this standard indicates an attempt to apply a reasonably deliberate approach to modern procedural language design, rather than simplistic "standardisation of existing Fortran practice".

The provision of a proper global data and program packaging, or encapsulation facility by the MODULE/USE extension is particularly noteworthy. This is a vastly superior linguistic device to the common, but in detail highly disparate extension using textual INCLUDE. The proposed MODULE/USE facility provides all the functionality of the simple textual include but, it also provides vastly more. It supports along with some of the other extensions a "semantic extension" capability. It does this without interacting and causing any serious syntactic or semantic incompatibility with existing non-standard vendor extensions. Although MODULE/USE provides for the functionality of INCLUDE, plus a great deal more, it does so in a way which allows the coexistence of all current vendor extensions.

This situation should be contrasted significantly with the incorporation of the NAMELIST facility. In this case the functionality has been modelled closely on a particular set of vendors' extensions. It is therefore certain to be entirely incompatible with other vendors' current offerings. To the extent that this functionality is needed it should have been included in a linguistically consistent and regular manner; with particular care being taken that no existing extension is needlessly made unable to coexist.

The process of standardisation by canonising some existing vendor's, usually ad-hoc, Fortran extension is not a desirable mode of operation. It is one of the strengths of this draft that it shows evidence of careful design rather than simple codification of often far from optimal common practice.

In spite of a generally favourable reaction to the proposed standard there are a number of significant flaws in the extended language it seeks to define. The most serious of these is the lack of a pointer capability. Without such a facility the construction and manipulation of such important data-structure as lists and trees is excessively cumbersome and inefficient. It is therefore strange, not to say perverse for the proposed Fortran 8X to include such powerful data-abstraction and structuring facilities as those offered by derived types, and the dynamic storage management capabilities of allocatable arrays without permitting the use of dynamic aliasing to provide the needed pointer capability.

There are other strange major lacks. There is no facility for handling variable length strings, or sequences of objects such as characters where the length of the object is determined by the value assigned rather than by the declaration of the name. Nor is there any input/output facility for handling data-transfer where the number of objects (usually characters) required is determined by the data rather than the format; Fortran 8X has no facility for character stream I/O. Also an entirely adequate syntactic mechanism has been included for floating-point data-types, but a similar mechanism has been omitted for integer.

As well as these unfortunate lacks there are a few facilities that have been included that are far from desirable or at least are defined in ways that are distinctly questionable. For example, it is both confusing and at the risk of being rude, absurd to preserve the keyword PARAMETER for the attribute used to declare a symbolic constant. The facilities for data initialisation are defined in a highly irregular fashion and need sorting out. Another glaring example is the RANGE/SETRANGE facility. The ability to take "ganged" identical sections of a number of arrays, to indicate this and to work with the sections in a concise manner is clearly useful. However, the facility as defined is cumbersome, restricted in functionality, dangerous in its notation, and above all relies on vastly too much syntactic linguistic baggage to be included in its present form.

A final general comment is that although the evolutionary model with "removed", "obsolescent", and "deprecated" features appears to be sound, the committee has been altogether too conservative in assigning features to the various categories. The conservatism would be justified

if Fortran was a proscriptive and closed standard. It is not. A Fortran processor is explicitly permitted to extend the language in any way that does not conflict with the standard (unlike a language like Ada which has a standard that proscribes completely the language for both the programmer and the processor). This permissiveness in the Fortran Standard would allow much more emphatic use of the evolutionary possibilities in the architectural design. Facilities could be removed from the standard, and provided no new facility produced a conflict with such facilities vendors would be free to maintain them in their products. Users of course would be more dramatically encouraged to stop using such facilities. The example of Hollerith in Fortran 77 is a good guide. Most vendors still support Hollerith but few new Fortran 77 programs use Hollerith. Some of the obsolescent features should therefore be "removed", more of the deprecated features designated "obsolescent", and additional features "deprecated". If for no other reason than to do so opens more options for the 1990's Standards Committee.

The remainder of this communication presents more detailed comments relating to each of the topics summarised in this general section. Each "negative comment" or "suggestion for change" will be presented with a reasonably detailed proposal for what needs to be done to the language as currently defined to make good the indicated problem. These will be technical descriptions rather than detailed editorial instructions for changes to the current draft standard document. It is considered that it is the responsibility of X3J3 to make the detailed editorial changes necessary to implement those comments or suggestions, with which it agrees.

Finally, the length of this document should not be taken as a sign of disapproval of Fortran 8X, as a whole. The language in aspects other than those explicitly mentioned is a significant and highly desirable extension to, and improvement of, Fortran.

2. DESIRABLE FACILITIES NOT CURRENTLY INCLUDED

The following sections include detailed discussions facilities that are highly desirable but which are not currently included. All of these facilities provide much needed functionalities and their absence in most cases causes a significant linguistic irregularity. The features involved are:

- 1. Pointers
- 2. Stream I/O
- 3. Selected range integers.

It should be noted that provided both adequate pointers and stream I/O are included the currently absent functionality of variable length strings can be implemented via a suitable MODULE. Without pointers an adequate MODULE for such objects is all but impossible. Without stream I/O a somewhat restricted MODULE is possible but string I/O would remain very cumbersome.

2.1 Pointers (30-7)

P. 206

The current proposed Fortran 8X introduces derived types and

heterogeneous data structures. It also introduces dynamic storage management and dynamic aliasing. However, these facilities are so constrained that the only dynamic sized data structure is an array, and dynamic aliases are restricted to be little more than renamed array sections. Dynamic data structures such as trees, lists, etc. cannot be produced easily; nor can structures with dynamically sized components be manipulated. The best that can be done with structures is to allow them to be parameterised and hence automatic sized structure components can be handled; structures with runtime determined sizes can be constructed on the stack but not on the heap. The programmer will still be forced to resort to various complex non-portable tricky coding techniques to map any dynamic - heterogeneous data-structures onto arrays. Fortran programmers will still have to map pointer operations into array indexing. This is clearly unreasonable. The language is forcing most of the costs of implementation for heap storage management and other pointer associated overheads, but delivering few if any of the benefits.

A minimal pointer facility could be easily produced by removing the restrictions currently applying to ALIAS-IDENTIFY and the ALLOCATABLE functionalities. For example allowing objects, including components of derived types, to be declared simultaneously to have both attributes would create a basic strongly typed pointer. Allowing both scalar and array objects to be ALLOCATED, and an alias to have multiple targets in IDENTIFY statements provides the essential manipulations of the pointer itself. A rule that provides for automatic de-referencing of pointers to their current targets in expression contexts and on the left of assignment provides the desired semantics in virtually all cases.

However, having these two very closely related attributes is adding unnecessary baggage to the language. It is also basically a distinction without functional advantage. An ALLOCATABLE object may be ALLOCATED to associate the named object with space, an ALIAS may be associated with space by IDENTIFY, but an ALIAS may not be ALLOCATED, and an ALLOCATABLE may not be identified. However, when a name is declared with either attribute what is created is a descriptor for an object. ALLOCATE and IDENTIFY are merely alternative ways of associating the descriptor with actual space. Having the two separate attributes and the restrictions on which methods of association are allowed seems only to add to language size and user confusion. The two attributes could usefully be collapsed into one. The possible attribute keywords which could be appropriate to characterise such descriptor declarations could be one of, say, VIRTUAL, DYNAMIC, NAME, or even POINTER.

The use of dummy arguments that are allocable and procedures that deliver results that are allocatable are the two features of the allocatable array facility which really only makes sense when viewed as part of a pointer model; in each case what is actually passed is the descriptor, or pointer to the space not the space itself. All other operations on allocated arrays (except course ALLOCATE and DEALLOCATE) treat them like any other array; i.e. the space is referred to not the descriptor. An alternative regularisation is possible which would include effective pointers, and preserve simple allocatable arrays for those applications that merely require such objects. This would retain the ALLOCATABLE attribute, the ALLOCATE and DEALLOCATE statements, but

P. 201

disallow allocatable dummy arguments and allocatable function results. The allocatable attribute then becomes a strictly local property permitting dynamic management of the size of such objects. This effectively removes the pointer-like properties from the facility. The locality of the allocatability of such objects also removes the need for an ALLOCATED function.

The pointer properties and functionality should then properly be provided by objects having the, say, VIRTUAL attribute. Such objects would need to be passable as arguments, returnable as results. They could be associated with space either by allocation or identification. VIRTUAL objects would be entirely appropriate as components of derived data-types; where they would allow linked lists and trees to be handled conveniently.

The optimisation problem caused by unconstrained pointers can be handled by requiring any static object that is to be used as the target of a pointer in an IDENTIFY to be declared as such a potential target; a TARGET attribute could be used for this purpose (Virtual objects should always be potential targets).

The problem of dangling pointers can be dealt with by requiring that an object is only deallocated if no other virtual object is associated with the space or parts of the space to be deallocated. Similarly, global pointers or dummy pointers must all be disassociated from any local non-saved objects before a return is exacted.

If pointers are not implemented then it is difficult to see how the ALIAS/IDENTIFY, ALLOCATABLE, etc. facilities can be justified. A language with data-structures and dynamic storage management overheads but no pointers is simply not reasonable or acceptable. If pointers are not to be added the language should be modified to restrict storage management requirements to stack operations. Heap should not be forced unless pointers are included.

## 2.2 Stream I/O

30-8

The lack of a method for reading and writing data; particularly character data, in such a way that the number of items read or written is determined by the data stream rather than the format or list is an extremely serious omission. This is most critical for input, but can also complicate output. The problem in its simplest form is that each READ/WRITE statement in Fortran automatically starts on a new record. It is not possible to simply read in the next character from the input stream, and to go on doing this until end-of-record is flagged before starting the next record. This makes the processing of varying length data by Fortran extremely difficult, and makes communication even at the file level between Fortran and other languages complicated, e.g. a character file written by a 'C' program could be extremely difficult to read with a Fortran program. At the very least it must be possible to turn off the automatic new record on each I/O statement, and to provide a flag for end-of-record.

Stream I/O could be treated as a characteristic of the connection. A formatted file could be connected for STREAM access rather than SEQUENTIAL or DIRECT. For such a connection only "sequential"

READ/WRITE statements would be permitted and no implicit end-of-records would occur. Each READ/WRITE would start transfers from the character position following the last character read or written. End-of-records would have to be indicated explicitly by the Format. There would also need to be some way of detecting an attempt to read past an EOR. The IOSTAT variable for stream I/O could be defined to return positive for errors, zero for successfully completed transfer, -1 if an EOR was detected, and less than -2 if EOF detected. The VALUES= specifier could be used to determine how many items were transferred. If PAD='YES' was specified then the list item being transferred at the time EOR is reached should be converted using the padding. If such a conversion is possible one is added to the VALUES=, if not VALUES= is not incremented and the last list item is undefined. If PAD='NO' then the list item being transferred is undefined. For List-directed stream I/O an EOR would be considered significant. Only the current record would be searched for input items and only the current record written. An EOR(NUNIT) intrinsic subroutine would be needed to force the taking of new records or a NEWLINE (position-spec-list) statement added. Stream I/O would never be considered to be printing.

Alternatively any sequential file could be managed for stream I/O by using the ci-control-list of the READ/WRITE statements. The REC= specifier could be permitted for sequential access with allowed values zero and one. One would be default and would mean start transfer from the start of the next record. Zero would mean start transfer from current position. A READ/WRITE would need to define the position after transfer as where the file pointer ends up, not at the start of next record as now. The use of IOSTAT to Detect EOR would need to be more subtle. IOSTAT would be set to -1 only if EOR was encountered when data was expected and it could not be correctly interpreted as a separator.

Of the two alternatives the former is probably preferable.

2.3 Ranged Integers (30-9)

The Fortran 8X proposal very effectively solves the linguistic, if not algorithmic, portability problems associated with the existence of multiple different floating point approximation methods for the representation of real, complex, and other floating point quantities. This obvious model should be extended to handle the entirely analogous problem of various different integer ranges. A suitable keyword for integers would be FIGURES. The type-spec would now become

```
INTEGER{range-selector}
range-selector IS ([FIGURES=]type-param-value)
```

A non-default declaration would be, say,

```
INTEGER(FIGURES=5) :: I
```

The processor would then be required to select its shortest integer representation for which HUGE(I)>= 99999

We would also need to have the coercion that in any operation the result will be in the representation of the longer operand. Obviously non-default integers must not be permitted in storage association

contexts. We would also need the equivalent of the MOULD (MOLD) arguments for the INT intrinsic to determine which representation was to be used for the converted result (see comment 4.7). Simple integer constants must for compatibility with Fortran 77 produce default-integer values. It will therefore, be necessary to have an analogue for the real EXPONENT LETTER. An obvious regularisation would be to extend this statement. If we defined it to be

EXPONENT LETTER numeric-type-spec defined-exponent-letter  
numeric-type-spec IS INTEGER [range-spec]  
OR REAL precision-selector

Constraint: the letter must not be the same as any other exponent-letter or the letters E, D or H.

e.g. EXPONENT LETTER INTEGER(FIGURES = 5)F

an appropriate constant could be written as

1234F, 1F4, 100F2, 10000F

The rules relating to REAL(\*,\*) dummy arguments and parameterised data-types with PRECISION and EXPONENT\_RANGE parameters could be taken over mutatis mutandis for INTEGER(\*) and Datatypes parametered by FIGURES.

3. THE RANGE/SETRANGE SECTIONING FACILITY

30-10

The current facility is irregular cumbersome and of severely restricted functionality. It uses a declaration attribute to permit a guaranteed conformant section to be taken in a number of arrays and for subsequent use of this section to be indicated by the unqualified array name. This use of the unqualified array-name is undesirable. users will find it extremely surprising to have A(1:10) possibly addressing legally more elements than a reference to A. Only one effective range can be applied to any one array. There is no way of having two simple sub-windows (ranges) moving through a single array. Also there is a further complication that all the array inquiry intrinsics are doubled so as to deal with both effective and declared bounds.

This is an impossible linguistic price to pay for such a limited functionality. There is also a considerable processor overhead. A rangeable array will have to be represented by a descriptor that involves two dope vectors. Either this will require the processor to handle a new kind of object or it will simply manufacture a hidden name for the alternative dope vector. The former is a significant processor overhead for very limited functionality, the second is an undesirable redundancy since this approach could be programmed explicitly by the user.

A way of achieving the required functionality already exists by use of aliases. This does not hide the sectioning since it introduces a new object name for the section but in most cases this will be a positive advantage and it does not introduce into the language any new objects. This approach would be particularly appropriate if pointers are included properly.



Alternatively, an approach could be employed which attaches a name to a section subscript list which can be subsequently used in a section reference in place of a complete section-subscript list. This would have a small notational advantage over the use of pointers to achieve the effect. It would however have substantial implementation overheads if a named range was allowed to become another data-type. For this reason it would be entirely undesirable for such a name to be declared and it should therefore be distinguished from other names by use of a reserved special character. A suitable special character would be the \$ which is otherwise still unused. It would be possible therefore to define a name starting with a \$ to be that of a section-subscript list, or range. The actual "outline" of such a name could be established by the execution of a SETRANGE statement, e.g.

$$\text{SETRANGE}(\$WINDOW) = (I,M:N,J:K)$$

would define the range \$WINDOW to refer to a rank 3 section-subscript-list and to select the Ith plane from a rank 3 array and the M:N by J:K rank 2 section from that plane. A statement like

$$A(\$WINDOW) = A(\$WINDOW) + B(\$WINDOW)*C(\$WINDOW)$$

would be a more convenient notation for

$$A(I,M:N,J:K) = A(I,M:N,J:K) + B(I,M:N,J:K)*C(I,M:N,J:K).$$

Of the alternatives simply adding pointers and some mechanism for making multiple identifications with the same range mapping would have minimum linguistic overhead and is therefore preferable, but a named section-subscript-list approach would be acceptable. The current RANGE facility is quite unacceptable and should be deleted from the standard.

#### 4. MISCELLANEOUS PROBLEMS OR DEFICIENCIES

##### 4.1 DATA/PARAMETER Attributes

30-11

The preservation in the attribute form of the mistake in nomenclature in Fortran 77 (a further example of inappropriate adherence to the detail of a common practice) of using the keyword PARAMETER for the statement defining the values for symbolic constants. The obvious keyword is CONSTANT. This should be used for the attribute form where a symbolic constant name is being declared and defined in one operation. Also the keyword CONSTANT should be allowed as the preferred spelling in the statement form; PARAMETER could then be deprecated. (In fact the statement form is entirely redundant and the whole form should be deprecated).

Similarly the use of DATA as the keyword for the initialisation attribute is unfortunate. In fact no attribute is required. The presence of the "= const-expr" in a declaration object list would be sufficient to indicate a variable initialisation, since constant definition is specified by the presence of the CONSTANT (PARAMETER) attribute.

P. 205

The residual ambiguity of how to interpret

REALA= 3.2

if this was the last specification statement in a program unit could easily be resolved by requiring the :: separator if variables are to be initialised in a declaration.

The removal of the need for the DATA attribute would increase flexibility. All variables declared in a statement would not have to be initialised, e.g.

REAL,ARRAY(100):: INPUT = [100[0.0]],OUTPUT

INPUT is initialised to zero but OUTPUT is left undefined.

Another problem with DATA is that the implied DO loop is singularly out of place in the object oriented form of the DATA statement. This form basically works as a set of compile time assignment statements, each one with an assignable object on either side of the =. However as currently defined this form allows a loop on the left of the equals. This is a highly irregular and inherently error prone construction; or at least one that will be difficult to use correctly. If the implied DO is to be retained in this form of the DATA statement then it should include the entire "assignment" within the range of the implied do. This would preserve the regularity of the assignment form.

4.2 NAMLIST

30-12

The need for a standardised version of this functionality is questionable, but if such a functionality is retained it should not be in this form. It is too close to but different in detail too many from existing extensions. Many of these will not be able to coexist with the current proposal hence rendering a number of existing programs both non-standard conforming and unrunable. This functionality should only be retained in a syntactic form consistent with the rest of the I/O style syntactically and not in direct conflict with existing extensions.

The need for, and desirability of, this functionality is questionable. Its use as a diagnostic aid has been overtaken by symbolic debuggers, and its use in production code is rare. Unless there is strong public call for it as justification, this functionality should simply be deleted.

4.3 The Executable/non-Executable Distinction

30-13

Given that run-time management of storage allocation both in stack form for automatic objects, and heap form for allocatable objects must be provided by any F8X processor, the distinction between executable and non-executable statements becomes entirely superfluous. In fact the maintenance of this distinction becomes positively pernicious. Since the creation of storage for any automatic object will necessarily have to be done at run-time when the values of the variable attributes are known there is no valid reason why executable code should not be involved in determining these attribute values. The distinction between

specification expressions and ordinary executable expressions is therefore entirely artificial, complicated to describe, and irregular. In most cases the artificiality of the situation is obvious since it can be programmed around in an entirely straightforward but cumbersome manner.

```
PROGRAM MAIN1
  INTEGER:: N
  READ(6,*)N
  REAL,ARRAY(N,N):: A
  ...
```

is illegal and so is

```
PROGRAM MAIN2
  INTEGER,DATA:: N= F(6)
  REAL,ARRAY(N,N):: A
  ...
CONTAINS
  FUNCTION F(NUNIT)
    INTEGER:: NUNIT, F
    READ(NUNIT,*)F
  END FUNCTION
END PROGRAM
```

However, if we add an extra level of sub-program

```
PROGRAM MAIN3
  INTEGER:: N
  N = F(6)
  CALL AUTO(N)
CONTAINS
  SUBROUTINE AUTO(N)
    REAL,ARRAY(N,N):: S
    ...

  END SUBROUTINE
  FUNCTION F(NUNIT)
    INTEGER:: NUNIT, F
    READ(NUNIT,*) F
  END FUNCTION
END PROGRAM
```

we obtain the obviously intended effect, but in a much less clear and obvious manner. Either MAIN1 or MAIN2 would also be easier to compile producing efficient code. The current restrictions will either produce many additional essentially redundant levels of sub-program, or will force the use of Heap storage when algorithmically stack storage is totally sufficient.

For scoping simplicity it is desirable to retain the restriction that all specification statements appear before any "executable" statements, but reference to internal procedures and accessible module procedures should be permitted in specification expressions; any object so declared to be considered automatic. The application of a simple "declare/define before use" rule would deal with any necessary ordering

P. 201

constraints which are now a matter of execution logic; specification statements should be subject to normal sequential processing. The removal of what is an obsolete vestige of Fortran 66 static storage management would have significant advantages in improving the data-abstraction facilities. A module could define the meaning of assignment, for say, a character string representing a value of this type to an object of the type. This definition assignment semantics could then be employed by user programs to define symbolic constants, without them having to know the internal structure of the data-type. The restriction to constant-expressions for the RHS of the = in such specifications would be reasonable. The processor has the option of executing the module procedure at compile time or delaying to run-time.

This regularisation is virtually essential. The current irregular restriction will require positive additional burdens on both user and processor to enforce, and therefore, are wholly unacceptable.

4.4 ALLOCATE/DEALLOCATE (30-14)

The requirement that an explicit DEALLOCATE be applied before an already allocated array is reallocated is likely to be simply an irritant rather than any significant aid to implementation or contribution to safety. It would be entirely consistent with normal program behaviour if ALLOCATE simply created new space and associated the allocatable object name with this space. If such a name was previously associated with space it would automatically be disassociated i.e. ALLOCATE is equivalent to a DEALLOCATE followed-by ALLOCATE.

4.5 The FORALL Statement/Block (30-15)

This statement/block is potentially extremely useful. FORALL provides a highly useful mechanism for writing what is in effect a "parallel DO loop", which is otherwise missing. It also provides an extremely simple, if crude, synchronisation mechanism; the only construct in the language that does. It should be reinstated in the language proper. It might with minor modification be made into a proper parallel-block construct, rather than a controlled array assignment as at present.

4.6 Class and Scope of Names (30-16)

The name of a type, derived or intrinsic, is essentially in a different class from that of a variable, etc. The rules at present allow

CHARACTER:: CHARACTER

but forbid

type(ATYPE):: ATYPE

This appears to be because the scope and class of type-names is not defined correctly. Intrinsic type-names have the scope of an executable program. Derived type-names have, as said in Chapter 14, the scope of the scoping unit in which the definition is accessible. However derived-type names are incorrectly defined to be in the same class as

variable names, whereas intrinsic type-names are not. Clearly for compatibility with Fortran 77 intrinsic type-names must be in a separate class from variables etc. and so for similar reasons should derived type names. Type-names should form a class.

Also value constructors should be classified as implicitly defined intrinsic procedures (c.f. type-parameter inquiry functions). This would remove an ambiguity/irregularity as to users overloading the type-name to provide an alternative type conversion function. A value constructor is essentially a type conversion function, which simply converts a list of arguments that match the components, to an object of the named type, and so should be described as such.

N.B. the user is permitted to write an overload such as

```
FUNCTION REAL(C)
  REAL:: REAL
  CHARACTER(*):: C
  ! convert character string value to real
  ...
END FUNCTION
```

However it is not clear from the current text whether a function overloading a derived type name in a similar manner would be permitted or not. It should be permitted and classifying value constructors as functions would do this unambiguously.

4.7 Type Conversion Functions/Value Constructors 30-17

The current set of type conversion functions are highly irregular; particularly when viewed along with interpreting value constructors as functions (comment 4.6).

A constructor function has the form

type-name[(output-type-param-value-list)] (component-argument-expr-list)

and the semantics "convert the values in the component-argument-expr-list to a value of type given by type-name consistent with the values provided in the output-type-param-value-list".

The Fortran 77 generic conversion function called REAL could easily be extended consistently to have the form

```
REAL[output-precision-selector] (A)
output-precision-selector IS precision-selector
OR (MOULD = real-object).
```

The constructor could be also extended to allow the optional type-param-value specification of MOULD = type-name-object.

All the intrinsic types should have syntactically similar type conversion functions (including an obvious extension for specified-range

P. 209

integers when included).

The irregular F77 forms of INT, CHAR, ICHAR etc. should be deprecated.

#### 4.9 Print Files

30-18

The current draft preserves the unfortunate and often disastrous bar to portability of allowing the processor to decide when a sequential formatted file is considered to be a printer file. A user therefore has no standard conforming way of forcing an output file to handle the first character of any record as data, or as a printer control. Some processors treat all disk files as non-printer files and swallow the first character only if the file is connected to a real printer. Others treat all files as printer files, and there is a spectrum in between. Some processors provide an additional OPEN specifier which controls how the initial character is handled.

Some form of user specification is necessary. It would be possible to add a value to the ACTION specifier of 'PRINT'. A file connected for ACTION= 'PRINT' would have to be output only and initial characters would be treated as carriage control. ACTION= 'WRITE' or 'READ/WRITE' should then be treated as data only files. The default should be 'UNKNOWN' where it is processor dependent how initial characters are handled on output, and it is a function of the statements used and the processor whether it was input, output or both.

#### 4.10 Derived Type I/O

30-19

Unformatted I/O for derived types is adequate as component by component, but component by component will frequently not be what is needed for formatted I/O of derived types. Unless an adequate extended definition both requiring and providing facilities for the user to specify how formatted I/O is to be performed, derived type objects should not be permitted to appear in formatted I/O lists. The user can write input procedures that read intrinsic values and can convert them as required by the semantics of the type, and similarly by a procedure can convert a value as required to represent it in the file as a series of character records, or part of a record. To include a partial and seriously inadequate facility is simply complicating the job of the next committee in doing the job properly without helping the users in this round.

An obvious possibility for an object of any type, the value of which could be represented by a character string contained within a single record, or field within a single record, would be to allow the user to define a Format procedure that would be associated with a derived type definition and would convert between derived type values and character strings and which could be invoked at the relevant place in the I/O processing. Unfortunately many derived types will require multiple records for their adequate representation in any formatted file. Therefore this otherwise very attractive approach is not directly applicable for the general problem.

No fully satisfactory solution to this problem has suggested

objects in Formatted I/O lists.

#### 4.11 Elemental Functions

30-20

The rules for elemental functions and elemental assignment are unsatisfactory. The fact that a function or assignment is to be callable elementally should be indicated in the definition of such functions. The rules of when this is possible are broadly correct, but those scalar arguments, along with the function result, which are to be extended to permit array valued arguments in an elemental reference should be declared. There is a need for an `ELEMENTAL` attribute which may be specified for scalar dummy arguments and function results (at least one dummy argument must be declared elemental along with the result) which may be elementally extended to be array valued. All elemental arguments must have conformant shapes in any reference..

#### 4.12 Overloading Rules

30-21

The present overload resolution rules would allow the shape of an array or the length of a character string in certain circumstances to resolve an overload. This is too general a rule. A sufficient rule and one which allows much cheaper implementations is that arguments must differ in type and/or rank to resolve an overload. This simplified overload rule will retain the essential functionality but will reduce greatly the potential size of any "specific name" symbol table that would be required to handle overload resolution and procedure linkages.

#### 4.13 Array Intrinsic Functions

30-22

The removal of `RANGE` as an attribute of an object and making it a true section, or based on pointers will reduce the need for duplicate array inquiry intrinsics. The enquiries for "effective" bounds etc. will no longer be needed.

There is a need for additional intrinsic functions to perform the Gather/Scatter operations that are common in sparse matrix applications. Vector valued subscripts are too uncontrolled to be used for this purpose and have rightly been deleted from the language, but the more limited functionality that could be provided by appropriate intrinsic functions is still a significant need.

#### 4.14 Complex as an Intrinsic Structure

30-23

The language should be marginally extended to allow `COMPLEX` to be handled in a fully regular fashion as an intrinsic structure. Basically this would require the imaginary and real parts to be accessible by the notation of component selection, say, as `ZVARXREAL` and `ZVARXIMAG`. This simple regularisation would also add much needed functionality. As it stands at present there is no way to use array notation to assign into the real parts of a complex array. `ZREAL` would provide such a mechanism we would also need (as would be consistent with 4.7) a type-conversion function

`COMPLEX[output-precision-selector] (real-part-exp, imag-part-expr)`

where the `imag-part-expr` argument was optional.

P. 211

4.15 Assignment Overloading

30-24

In general it is illegal to overload in such a way as to seek to define a new meaning for an already existing operation; it is not permissible to redefine the semantics of +, for integers. However it should be permissible to redefine intrinsic assignment if the arguments are of a derived type. Intrinsic assignment is defined for derived type objects with identical type and type-parameters. For many types it will be necessary to define assignment for non-identical parameters and this will almost certainly not be simply component by component assignment. The procedure implementing such an assignment will in most cases also define assignment for equal type-parameter values as well as for non-equal. A rule such as "If a user defined assignment subroutine with at least one argument of a derived type is accessible this is used rather than intrinsic assignment even if intrinsic assignment was applicable".

4.15 Exponent Letters

30-25

Exponent letters E, D or any defined exponent letters form a class of identifiers that is separate from that of any other names, but this is not stated clearly. It should be so stated.

4.16 Array Component Selection

~~30-26~~

A scalar component may be selected from an array of structures producing an array whose shape is that of the structure array and whose type is that of the component. An array component may be selected from a scalar structure producing the component array type and shape. However, the obvious regularity is not permitted of selecting an array component of structures with the obvious interpretation as a multi-dimensional array of the component type.

It might be argued that an array component from an array of structures is an array of arrays and this is not a multi-dimensional array. This argument is spurious. It is a multi-dimensional array in that it can be described perfectly well be a dope-vector like any other array. The argument is confusing the semantic modelling of entities with properties of their representation. A two-dimensional array of reals may or may not represent a matrix. This is a question determined by the detailed semantics of an actual program. Similarly a vector of vectors may or may not be a matrix for similar reasons, but if a vector is represented by a one dimensional array of reals a vector of vectors is most definitely represented by a two-dimensional array of reals.

30-26

Of course allowing array component selection from an array of structures highlights the unfortunate choice of ordering for component selection, viz.

structure % component

It would be much more regular and consistent with Fortran conventions if the ordering was

component % structure

30



Even in the absence of implicit declaration it is common practice among many users to continue to use a naming convention which indicates the type of an object by the first characters of its name. Since the type of a selected component is that of the component this argues for putting the component first. It is also the case that in manipulations of a selected component the major determining factor controlling what is permissible is the attributes of the component. The fact that the object is selected from a structure is of secondary significance. One thinks in the main of manipulating the such and such component qualified by the fact that it is selected from the so and so structure; the structure is conceptually a qualifying feature and hence more logically comes second. Also, since Fortran retains column major ordering putting component first would allow the component and structure subscripts to associate in a simple left to right correspondence when an array component from an array structure is associated with a multi-dimensional array of the component type. The qualification reads better in component % structure order. The % is a logical symbol to use this way round given its frequent use as an abbreviation for "care of".

30  
c.o.

4.17 Very Long and Inconsistent Function Names

30-28

The use of excessive long names and the lack of consistency in the way these names have been constructed is entirely unreasonable. For example, compare

EFFECTIVE\_EXPONENT\_RANGE(X)

for the floating point inquiry function with

DUBOUND(A)

for the array bound enquiry function. Either one should be shortened or the other expanded consistently. That is either we should have something like

EERANGE(X)

or DECLARED\_UPPER\_BOUND(A).

The use of abbreviated names for functions is preferable. Long names for subroutines is reasonable, but functions are referenced in expressions where the effect of the very long names is to obscure the structure of the expression making them both tedious to write and extremely difficult to read.

Another possibility would be to provide both long descriptive names and as well abbreviated aliases. Of these possibilities the most acceptable is to adopt a set of abbreviated names for all the functions. The main changes should be

- |                          |   |         |
|--------------------------|---|---------|
| EFFECTIVE_EXPONENT_RANGE | - | EERANGE |
| EFFECTIVE_PRECISION      | - | EPREC   |
| MAXEXPONENT              | - | MAXEXP  |
| MINEXPONENT              | - | MINEXP  |
| SETEXPONENT              | - | SETEXP  |
| DOTPRODUCT               | - | DOTPROD |

P. 213

#### 4.18 Assumed versus Generic Type-parameters

30-29

The language has two very different semantics indicated by a similar syntax. For type-parameters like character length the declaration of a dummy argument with the type-param-value as an asterisk indicates that the actual value is taken from the associated actual argument. Such type-param-values are dynamically variable at runtime. The creation of such dummy arguments and any local objects with attributes determined relative to such arguments and parameters are necessarily automatic and will be allocated space on a stack at run-time. All such attributes assume their values independently. For PRECISION and EXPONENT\_RANGE parameters (and also for FIGURES for integers if this were to be included) the specification of asterisk values for floating-point dummy arguments indicates that the procedure is to be generic over available floating point representations and these dummy arguments are the generic arguments. An actual parameter value is not assumed. All generic dummy arguments must be of the same representation. All associated actual arguments must have been declared the same, but on different calls actual sets of arguments with different declared precision-selectors and hence possibly different floating point representations may be used.

These two very different semantics should be indicated by different syntax. The problem is really to over-use of the asterisk symbol. Its use has been extended because of its use for assumed-size arrays and Fortran 77 style assumed character lengths, gathering even greater irregularity. Another character should be used to indicate FBX assumed parameter values, and the asterisk retained for the deprecated assumed-size arrays, Fortran 77 assumed character lengths (this should be deprecated), declaring symbolic character constants, and generic procedures. These are all different mostly deprecated and the asterisk becomes an indication that something different is being done with attribute values. A highly suitable character for indicating the regular assumption of type-param-values would be the question mark, ?. This provides a clear indication that when written this value is unknown, the value to be provided later on invocation.

#### 4.19 PAD and TRIM Characters

In many applications the Blank character is not irrelevant even in trailing contexts. This makes its use as a pad character and the character trimmed off by TRIM and ADJUSTL/R functions, unfortunate. Ideally there should be a processor directive to define a character to be used as an alternative to blank. However as there appears to be no mechanism at present for such directives there should be some way of specifying the character to be used as the pad character, in each context where it is required. The TRIM, LEN\_TRIM, ADJUSTL/R functions could have an optional extra argument which could specify the character to be used as "padding". The PAD= specifier for an OPEN statement could be modified to allow padding to be turned on and to specify the character to be used for the logical padding. For instance, it would be possible to say the form of PAD= is

PAD= char-expr

If LEN(char-expr) = 0 no padding is done otherwise padding is performed

using the character char-expr(1:1) as the pad character; the default being PAD= 'blank'

This still leaves the problem of assignment where padding if required would be by blanks (this is why a directive is preferable assignment would also be changed). However this problem could be programmed around. Either accurate precalculation of assigned lengths could restrict the occurrence of assignment padding where critical, or the combination

TRIM (TRIM(String), PAD= ACHAR(0))

To first remove blanks possibly added by an assignment then to remove specific padding, assumed to have been added and to be ASCII null.

5. CONCLUSION

The draft standard is a welcome and desirable enhancement to the capabilities of Fortran, but it needs to be amended, to complete the basic design and to remove a number of infelicities, before it is adopted as a standard that is intended to define Fortran until the turn of the Century and beyond.

P. 215

87 DEC 21 P 1:44

.....EX

~~31-1~~ #31

43

Convex Computer Corp.  
701 N. Plano Rd.  
Richardson, TX 75081

December 8, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

31-1 → I think that the array triplet notation is an important improvement. I also feel that  
31-2 → the WHERE statement is a needed extension. I think that the abstract data typing is ← 31  
an overly complicated change to FORTRAN.

31-4 → I also feel that the numerical precision control is an unneeded modification to the  
31-5 → language. I am unhappy that the IMPLICIT NONE statement was missing. I believe  
31-6 → that the DOUBLE PRECISION statement should not be taken out of from any  
FORTRAN of the future.

I am opposed to the standard as currently proposed. Please fix the problems outlined in this letter in the proposed standard.

Sincerely,

*Renee Kellow*  
Renee Kellow

<u>Subgroup</u>	<u>Nominations:</u>
31-1	GEN
31-2	GEN
31-3	DATA
31-4	DATA
31-5	DATA
31-6	GEN

November 15, 1987

Ms. Catherine Katchurik  
 X3 Secretariat/CBEMA  
 311 First St NW, Suite 500  
 Washington, DC 20001  
 copy to  
 Board of Standards Review  
 American National Standards Institute  
 1430 Broadway  
 New York City, NY 10018

Subgroup Nominations:

32-1	DATA	32-7	PROC
32-2	CIO	32-8	PROC
32-3	PROC	32-9	CIO
32-4	DATA	32-10	PROC
32-5	DATA	32-11	GEN
32-6	DATA	32-12	DATA

I would like to make several comments on the proposed new Fortran standard. I will tell you what I think the new standard should do to improve on Fortran 77, and then my points of disagreement with the draft I have seen.

First, there are several things which either cannot be done in Fortran 77 or can be done only with kludges. The first of these is logical operations on integers. I use these very extensively in programs which directly control hardware registers. The functions AND, OR, NOT (1's complement), and XOR for integers are implemented in most but not all implementations of Fortran 77. While it is theoretically possible to get the equivalent effect using arithmetic operations, it is generally rather slow, even with the best optimizing compilers. I would have expected that these operations would have been included in any new standard. I note that there is a "bit" data type described in an appendix, but not included in the standard. The amount of language constructs associated with this proposed construct is rather large, and except for its proposed array masking function, would be easily supplanted by simple bit functions. Of course, these functions would only be guaranteed to generate the same arithmetic results on positive integers, given the difference between 1's and 2's complement hardware, and would have to be carefully defined to allow the (extremely remote) possibility that someone would write a Fortran 8x on a decimal machine. Nevertheless, they should be included. The cost is negligible, and the utility extremely great for those persons who need them.

32-

The second is the difficulty of using Fortran IO using cursor addressing on a terminal. For example, in Fortran 77 it is impossible to write a prompt on one line and get input on the same line. This is fixed in the F8x draft with the PROMPT keyword. This is very nice as far as it goes.

However, it still appears that a carriage return will be emitted at the end of the line inputted. A facility, perhaps a keyword NODEFAULTCARRIAGERETURN, or something shorter, should be included in the OPEN statement, to the effect that neither on input or output should carriage returns (or linefeeds, which are worse) be generated unless explicitly asked for.

32-2

Fortran lacks any mechanism, such as pointers, for directly addressing memory. I do not propose adding pointers. However, I believe that Fortran should recognize the existence, indeed the dominance, in numbers of existing computers, of computers in which absolute memory locations need to be used. I refer, of course, to memory mapped video screens. There should be some standard syntax to read and write such memory. It could not possibly be portable in execution, of course, but I propose that you include a standard syntax for a "peek" and a "poke" function for memory, and another pair of functions for those computers which have a separate IO address space. I realize that this would graft a big wart onto the logical

32-3

structure of Fortran, but the fact is that such a construct is NECESSARY, and you might as well make a standard syntax for it. At the very least, this would allow easier porting of programs from one machine to another, as the programmer would at least be able to find instances of these constructs by a simple editor search. This is much superior to the expedient of having the user define such functions in assembly language, as is now necessary, as these would be inline functions in most implementations and would thus be much more efficient. It is to be noted, also, that the compiler would recognize that two successive "peek"s of the same location might return differing results. That, of course, is a compiler-dependent implementation issue.

32-3  
cont

Another problem with Fortran 77 is the impossibility of initializing character variables and arrays (and generating constants) with nonprinting characters. The proposed mechanism, involving concatenation of strings and a method of including nonprinting characters, is quite satisfactory.

32-4

I note that some persons have proposed inserting a new construct to allow more than 256 characters, perhaps to accommodate Eastern languages such as Japanese or Arabic. This is not the province of an American national standard, or even an international one. It should be left to the national standards body of the country involved.

32-5

I note the concept of "trigraphs" which would result in character substitutions even in characters strings. This is just plain stupid. If someone wishes to use a non-standard character set, they should be the ones to suffer discomfiture. I note that the corresponding committee generating a C standard did not see any need for a corresponding kludge.

32-6

It is frequently expedient in Fortran 77 to get the bit pattern of one type of object into another type. This had to be done by the very kludgy method of using EQUIVALENCE. I note that the proposed standard has the TRANSFER function, which if I read it correctly, will do this. This is a very useful feature and I strongly support its inclusion.

32-7

I note that there are a large list of proposed intrinsic functions. Consistent with any changes to reduce the size of the proposed language which might render some meaningless, I suggest retaining them.

32-8

One of the defects of Fortran 77 is its paucity of looping constructs. I would have fixed this by simply including all the constructs present in the C language, but the constructs which you have provided are certainly sufficiently expansive that I support their inclusion.

32-9

One defect of Fortran 77 is the lack of any INCLUDE facility, and the lack of a preprocessor or macro facility. I note that you have declined to include the INCLUDE statement found in many implementations of Fortran 77. I presume that this was done for two reasons, one being that a choice would have to be made between then numerous differing syntaxes in different vendors implementations, the other being that hard-coded filenames are intrinsically non-portable. The syntax of the MODULE-USE facility you propose fixes the second objection, and I support it. However, the MODULE construct is flawed in that not all of the rest of Fortran can be included inside a module, in particular code fragments, as opposed to whole subroutines, appear not to be accommodated. This is a serious flaw. This construct, that is small fragments of code, is extremely useful, given the lack of a macro facility. The module concept should be expanded so that any legal construct can be used in it. I note that no macro facility is included in the proposed draft. I have no opinion pro or con on whether it should be there, given the other facilities already there which partially take its place (including your C compiler's preprocessor).

32-10

I now come to my objections to the proposed standard. Some of these

are so severe that I consider it to be fatally flawed. The first of these is the concept of "deprecated features". I note that the list of these include a large fraction of FORTRAN, FORTRAN II, FORTRAN IV, and FORTRAN 66, including the heart of Fortran's storage allocation mechanism, COMMON and EQUIVALENCE. It may have been your intention to include in Fortran 8x a "superior" mechanism ( I don't consider that your proposal is superior, just different), but the idea of deleting ANY part of present Fortran is absolutely, utterly, fatally, flawed. I realize that you don't propose to remove these constructs from Fortran 8x. But simply to propose that they might disappear in the future is a fatal flaw. It is as simple as this: I expect that all my present legal Fortran programs and subroutine libraries will remain valid forever. It is simply a waste of time to have to rewrite a working program or library. To even propose removing vital parts of the present language is stupid, indeed laughable. The idea of deprecated features should be removed from the draft, and replaced with a statement that all of Fortran 77 will remain in future versions, forever. The present draft should be totally rejected until it is changed to state that all of Fortran 77 is expected to be included in future versions.

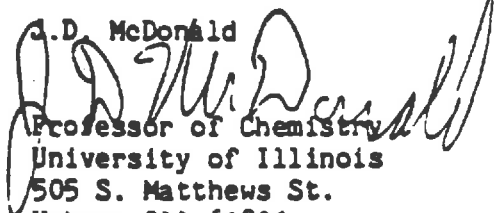
32-11

I have read that some persons believe that the proposed Fortran 8x is simply too large. While it is certainly very large, I am not sure what "too large" would be. It does seem that some of the proposed extensions are redundant. I agree that the proposal to allow operations on sections of arrays without DO loops is a useful extension and is a good reason to make the language larger. However, given that a perfectly useful system for declaration statements now exists, and I expect that it will remain in Fortran in the future, it seems that the whole new mechanism which you propose it eventually replace it is large extension which should be carefully scrutinized. I believe that a small extension to the present syntax, to allow more different sizes of integers and reals, without changing the flavor of the present system, would be better. Since most modern computers have word sizes which are a power of two, it would be proper at recognize that fact, and design a syntax addition which allow use of all the word sizes available, with an adaptation for machines with oddball word sizes. If you were designing a language de novo, your proposed method would be fine, but I feel it better if you simply extended the present method.

32-1

In summary, I strongly suggest that the present proposal be returned to the drafting committee, with a mandate given to that committee to come up with a new, smaller, proposal which recognizes that any future revision should add to the present standard, rather than attempt to supplant any part of it. Without very substantial changes, the present draft Fortran 8x should be rejected.

Sincerely,

J.D. McDonald  
  
 Professor of Chemistry  
 University of Illinois  
 505 S. Matthews St.  
 Urbana Ill 61801



# 33

National Aeronautics and  
Space Administration

Washington, D.C.  
20546

43

X3 SECRETARIAT

87 DEC 11 1987

DEC 9 1987

Reply to Attn of

NTD

X3 Secretariat  
CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001

Dear Sir or Madam:

Enclosed are comments from four of the National Aeronautics and Space Administration's field offices concerning the proposed draft revision to the American National Standard for Fortran.

If you have any questions regarding these comments or require additional information, please call Mary Hofmann at 453-1794.

Sincerely,

Russell S. Rice  
Director, IRM Policy Division

4 enclosures

- ✓ 1. Review of Proposed Standard from Ames Research Center
- 2. Review of Proposed Standard from Jet Propulsion Laboratory
- 3. Review of Proposed Standard from Marshall Space Flight Center
- 4. Review of Proposed Standard from Kennedy Space Center

cc:  
Mr. James H. Burrows, Director  
Institute for Computer Sciences and Technology  
United States Department of Commerce  
National Bureau of Standards  
Gaithersburg, MD 20899



43

Subgroup Memo.

33-1 DATA  
33-2 PROC  
33-3 GEN  
33-4 PROC  
33-5 GEN  
33-6 DATA

A Review of the Proposed Fortran 8X Standard  
David H. Bailey  
NAS Systems Division  
November 25, 1987

Recently the ANSI committee (known as X3J3) that has been working on the new Fortran standard completed its draft and has submitted it for public comment. This new standard adds a considerable number of new features, and in many respects represents a departure from the comparatively simple Fortran language of yesteryear. This note summarizes these new features and attempts to assess their impact on scientific computation both in the NAS project and at NASA Ames in general.

Overview

The goal of the committee in preparing this standard was not to merely standardize a few pet extensions that had come into popular usage, but to modernize the language so that it can continue to function as the premier language of scientific computing well into the next century. The proposed Fortran 8X standard, in spite of its inclusion of many new features, is strictly upwardly compatible with Fortran-77. Thus any standard-conforming Fortran-77 program will also be a standard-conforming Fortran-8X program. The new features that are proposed to be incorporated into Fortran 8X represent in most cases features that have been found to be very effective in other modern languages, such as Pascal, APL, and Ada. For example, structured data types, which were popularized in the Pascal language, have been incorporated into 8X. Similarly, the array operations that many feel are the most significant of the new features of Fortran 8X, have their roots in the array operations of APL.

Summary of New Features

A. Source code format

1. Lower case alphabetic characters are allowed.
2. Long identifiers (up to 31 characters long) are allowed.
3. Code may be optionally entered in a new free format.
4. ., ., :-, :- etc. may be used instead of .LT., .GT., etc.

B. Floating-point precision control

1. REAL declarations may include machine-independent precision levels. For example, the following specifies that X must be represented by a hardware data type with at least 12 digits of precision and an exponent range of at least  $10^{(+ - 80)}$ :  
REAL (12, 80) X
2. Exponent letters may be defined to specify constants with a machine-independent level of precision. Example:

P. 221

X - 3.125F0

C. New data types

- 1. The user may define data structures in a manner quite similar to Pascal and Ada. Example:

```

TYPE LINE
  REAL, ARRAY(2,2) :: COORD
  REAL :: WIDTH
  INTEGER :: PATTERN
END TYPE LINE

```

- 2. Dynamically allocatable data arrays may be defined and controlled.

D. New subprogram constructs

- 1. The RECURSIVE descriptor allows a subroutine to call itself.
- 2. The OPERATOR descriptor in a FUNCTION subprogram allows a function to be identified by a symbol, such as +, \*, /.
- 3. MODULE definitions allow global data and procedures to be defined.
- 4. Keywords may be used in subprogram references. Example:  
CALL SUB1 (A, B, 3., N = 4)

E. New program control statements

- 1. A CASE construct has been defined. Example:

```

SELECT CASE (N)
  CASE (:-1)      |  N .LE. -1
    SIGNUM = -1
  CASE (0)        |  N .EQ. 0
    SIGNUM = 0
  CASE (1:)       |  N .GE. 1
    SIGNUM = 1
END SELECT

```

- 2. New DO syntaxes have been defined. Examples:

```

DO I = 1, 100
  A(I) = 3. * B(I) ** 2
ENDDO

DO
  CALL RANDOM (X)
  IF (X .GT. 0.99) GOTO 10
ENDDO

```

- 3. A WHERE-ELSEWHERE construct has been defined that is similar to the IF-THEN-ELSE construct, except that it applies to arrays.

F. Array constructs

- 1. Most arithmetic operations and intrinsic functions have been extended to handle array arguments. Example:

```

DIMENSION A(50,50), B(50,50), C(100,100)
C(1:50,51:100) = 3. * A + SIN (B)

```

- 2. An IDENTIFY construct has been defined to allow a user to identify a variable with a subsection of an array.

IDENTIFY (DIAG(I) = A(I,I), I = 1:N)

3. Numerous APL-style intrinsic array operators have been defined. As in APL, they are rather general in definition, extending in natural ways to arrays of several dimensions. Examples:

MATMUL	Performs matrix multiplication
DOTPRODUCT	Computes the dot product of two arrays
MAXVAL	Finds the maximum value of an array
MAXLOC	Finds the location of the maximum
TRANPOSE	Transposes an array
CSHIFT	Performs a circular shift in one dimension
MERGE	Joins two arrays in one dimension
PACK	Compresses an array according to a mask
UNPACK	Expands an array according to a mask

G. New I/O constructs

1. The NAMELIST I/O that has been popularized by a number of manufacturers has been accepted essentially unchanged into the Fortran 8X standard.

H. New intrinsic functions

1. Numerous new intrinsic functions have been defined, in addition to the array intrinsics mentioned above.

Examples:

LEN_TRIM	Length of a character string after trimming trailing blanks
SCAN	Scans a string for a character in a set
EPSILON	Machine hardware "epsilon"
RANDOM	Pseudo-random number between 0 and 1
DATE_AND_TIME	Obtains the current date and time

I. Incremental functions

1. A number of constructs from past Fortran standards have been declared "incremental", i.e. they are considered "bad" and are candidates for removal from future standards. In almost all cases their functionality is provided by other Fortran 8X constructs. Examples:

Arithmetic IF statements  
 DO loops with REAL control variables  
 Ending DO loops with other than ENDDO or CONTINUE  
 Assigned GOTO statements  
 COMMON blocks  
 ENTRY statements  
 EQUIVALENCE statements  
 Computed GOTO statements  
 Specific names for intrinsic functions

Discussion and Conclusion

There is no doubt that the current proposed Fortran 8X standard is controversial. At least two points

features to be implemented cleanly and efficiently. Some large scientific centers, notably Los Alamos, have voiced opposition that it does not include some Fortran extensions, such as POINTER types, that they have been relying on. They also are upset at the establishment of "decremental" features.

On the other hand, study of the standards document and discussion with committee members reveals that the preparation of this standard has been a very difficult undertaking. Many pet Fortran extensions were proposed for inclusion, and only a subset could be accommodated without spending even more time and making the standard even more unwieldy than it already is. Working out all the minute ramifications of the array constructs, for example, was very demanding, and whenever a revision was proposed to the draft standard, considerable effort was required to insure there would be no unforeseen side effects with other parts of the standard.

There certainly are features of this standard that in the reviewer's opinion either unnecessary or disappointingly clumsy. For example, I doubt if very many scientific programmers will ever use the Pascal-like structured data types. Similarly, I don't think that the new keyword syntaxes will gain much acceptance among Fortran programmers, who for the most part will not even explicitly declare the types of their variables. 3  
33-2

On the other hand, other features of the standard are badly needed at NASA Ames and other advanced scientific computation centers. Key among these are the constructs for array computation. These constructs would provide, at long last, a means for explicit declaration of vectorizable and parallelizable operations. With the current Fortran language, scientists who generally know perfectly well that a certain operation is parallelizable or vectorizable must first translate that parallel concept into sequential terms for coding. Vectorizing compilers must then attempt to read the programmer's mind, reconstructing the original parallel operation for execution on the underlying hardware. These compilers rarely detect all vectorizable constructs -- in some scientific codes, up to 75% of vectorizable loops are not detected automatically by the compiler and require manual intervention by the programmer to achieve vectorization. Such problems will be completely avoided by usage of the Fortran 8X array constructs. In fact, these array constructs will represent the first concrete steps towards a truly machine-independent parallel programming language. 33-3

Other constructs proposed for inclusion in the Fortran 8X standard are equally crucial for continued progress in advanced scientific computing, both at Ames and elsewhere. These include the RECURSIVE subprogram descriptor, without which a large body of newly discovered advanced algorithms cannot be easily programmed. The modernization of the Fortran source code format is equally crucial. Many an hour of programming time has been P. 22  
33  
33

Also very important in the proposal are the inclusion of machine-independent precision standards. These features will greatly facilitate the development of scientific programs that are truly portable between machines. } 33-6

In summary, while I for one do not like all of the new features in the proposed standard, nonetheless I recommend that Ames strongly support its adoption, largely because it includes many features that we simply cannot do without. If opponents of the proposed standard are successful in derailing it, it may be years before a revised standard is approved. In the meantime, few vendors will adopt any of its features out of fear that they will not be compatible with the revised standard, and Ames programmers will be deprived of the potential benefits.



National Aeronautics and  
Space Administration

Washington, D.C.  
20546

#34

43

X3 SECRETARIAT

'87 DEC 11 10:56

DEC 9 1987

Reply to Air of NTD

X3 Secretariat  
CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001

Dear Sir or Madam:

Enclosed are comments from four of the National Aeronautics and Space Administration's field offices concerning the proposed draft revision to the American National Standard for Fortran.

If you have any questions regarding these comments or require additional information, please call Mary Hofmann at 453-1794.

Sincerely,

Russell S. Rice  
Director, IRM Policy Division

4 enclosures

1. Review of Proposed Standard from Ames Research Center
2. Review of Proposed Standard from Jet Propulsion Laboratory
3. Review of Proposed Standard from Marshall Space Flight Center
- ✓ 4. Review of Proposed Standard from Kennedy Space Center

cc:

Mr. James H. Burrows, Director  
Institute for Computer Sciences and Technology  
United States Department of Commerce  
National Bureau of Standards  
Gaithersburg, MD 20899

Comments on the Proposed New Fortran Standard

Phil Hooper

Kennedy Space Center

It is strongly recommended that the following structured programming constructs be included as extensions to DO:

- a. DO {label} WHILE (scalar-logical-expr)
- b. DO {label} UNTIL (scalar-logical-expr)

implementing top-test and bottom-test loop control respectively.

A number of vendors have implemented DO WHILE as a language extension. A lesser number have implemented DO UNTIL.

In addition, it should be noted that Fortran 8x would be a significant improvement over Fortran 77. The addition of array processing constructs adds a much needed capability to the language. The addition of case statements and the modification of the do loop construct improves the structure of the language. While not yet a "structured" language, much improvement has been made in this area.

Nothing has been deleted from Fortran 8x that was a part of Fortran 77, although some items have been identified as not needed and recommended for future deletion. This means that any program written to the Fortran 77 standard will compile and run correctly with a Fortran 8x compliant compiler.

There are enough differences between the new and old standards that current programmers and programming managers may need to take update seminars to take full advantage of Fortran 8x compilers.

<u>Subgroup</u>	<u>Nominations</u>
34-1	CIO
34-2	GEN
34-3	CIO



National Aeronautics and  
Space Administration

Washington, D.C.  
20546

43 #35

X3 SECRETARIAT

87 DEC 11 12:56

DEC 9 1987

Reply in form of

NTD

X3 Secretariat  
CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001

Dear Sir or Madam:

Enclosed are comments from four of the National Aeronautics and Space Administration's field offices concerning the proposed draft revision to the American National Standard for Fortran.

If you have any questions regarding these comments or require additional information, please call Mary Hofmann at 453-1794.

Sincerely,

Russell S. Rice  
Director, IRM Policy Division

4 enclosures

1. Review of Proposed Standard from Ames Research Center
2. Review of Proposed Standard from Jet Propulsion Laboratory
- ✓ 3. Review of Proposed Standard from Marshall Space Flight Center
4. Review of Proposed Standard from Kennedy Space Center

cc:

Mr. James H. Burrows, Director  
Institute for Computer Sciences and Technology  
United States Department of Commerce  
National Bureau of Standards  
Gaithersburg, MD 20899



Comments on the Proposed New Fortran Standard

John C. Lynn

George C. Marshall Space Flight Center

The proposed draft for the revised Fortran 8x standard appears to be consistent with the Fortran 77 standards currently in effect. The Fortran 8x standard will allow existing programs at Marshall Space Flight Center to run with little or no changes required. Enhancements to use new features of the language will not be difficult.

Although we anticipate no problem with acceptance of the Fortran 8x standard, we believe implementation of the following additional features would be desirable:

- a. Do until statement. } 35-1
- b. Do while statement. }
- c. Range statement (allow user to hard code the actual range of a variable). 35-2

Subgroup	Nominations:
35-1	CIO
35-2	DATA



National Aeronautics and  
Space Administration

Washington, D.C.  
20546

#36 (43)

X3 SECRETARIAT

'87 DEC 11 12:56

DEC 9 1987

Reply to Attn of NTD

X3 Secretariat  
CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001

Dear Sir or Madam:

Enclosed are comments from four of the National Aeronautics and Space Administration's field offices concerning the proposed draft revision to the American National Standard for Fortran.

If you have any questions regarding these comments or require additional information, please call Mary Hofmann at 453-1794.

Sincerely,

Russell S. Rice  
Director, IRM Policy Division

4 enclosures

1. Review of Proposed Standard from Ames Research Center
- ✓ 2. Review of Proposed Standard from Jet Propulsion Laboratory
3. Review of Proposed Standard from Marshall Space Flight Center
4. Review of Proposed Standard from Kennedy Space Center

cc:

Mr. James H. Burrows, Director  
Institute for Computer Sciences and Technology  
United States Department of Commerce  
National Bureau of Standards  
Gaithersburg, MD 20899

43

Subgroup	Nominia
36-1	DATA
36-2	DATA
36-3	PROC
36-4	CIO
36-5	GEN
36-6	DATA
36-7	DATA
36-8	CIO
36-9	PROC
36-10	PROC
36-11	CIO
36-12	CIO
36-13	CIO

**Comments on the Proposed New Fortran Standard**

**Fred T. Krogh and Charles L. Lawson**

**Jet Propulsion Laboratory, Section 366**

**November 13, 1987**

**Comments on Removed Extensions**

- 1. Bit Data Type:** This is a feature we would like to see, although there is something else that would be of more use and which could contain bit data type as a special case. We would like to be able to store and access bit fields that are packed in a word. Probably the easiest way to get this is to allow data structures to be packed and to give the programmer some way to control this packing. Such a feature would be useful in packing information in symbol tables and/or data bases and would make it possible in some cases to interface with external hardware without leaving Fortran. (36-)
- 2. Variant Structures:** This is a feature that we would also find useful. It is almost essential for the implementation of a symbol table that contains a variety of objects. If you want to get rid of Equivalence, which we have mixed feelings about, it would be nice to have something like this to ease the transition. (36-)
- 3. Internal Procedure Name as an Actual Argument:** Software for differential equations, quadrature, nonlinear least squares, and many other areas of mathematical software frequently requires the user to supply some arbitrary function. A very clean interface is possible if the user can pass the name of the function, but keep the code for the function in the calling program. Thus we would like any restrictions on the passing of internal procedures as actual arguments to allow at least this much functionality. Recursive internal procedures, for example, would not be important in this application. (36-3)
- 4. Condition Handling:** This feature is very desirable to have in a language. It is frequently the case that the underlying hardware can check for exceptional conditions with no overhead, that are expensive or awkward to check with software. (36-4)
- 5. Significant Blanks in Free Source Form:** Future extensions of Fortran are likely to find significant blanks useful. They make a lexer easier to write, ensure that code is not written in a way that is very difficult for the human eye. (36-5)

to lex, and offer the possibility of a slightly cleaner syntax in the future. Other languages have significant blanks. The time to make the change is now when there is no problem with backward compatibility. If it isn't made at the time free form source is introduced, then it will be very hard to get in later. We like the idea of free form source, but would rather see it eliminated than have it introduced with blanks that are invisible. Note that programs will certainly be written to convert programs from the old source form to the new form, and such programs could very easily remove unnecessary blanks and insert necessary ones along with the other changes that they make in the source.

#### Other Desired Extensions

6. Pointers: We would like to see a pointer facility added to the language. (36-
7. Support for Object Oriented Programming: We believe that object oriented programming is a very promising way to enable the reuse of software components. The immediate support desired is to allow functions to receive values. Notionally such functions might well look like references to a structure component. Such support would be most useful if one could declare that references to such object are to be done with code that is in line, since in many cases the code involved is very small. Inheritance is probably too much to ask for at this time, but should be kept in mind for the future. (36-
8. Control Structure Exits: We use a Fortran preprocessor that allows control blocks to be labeled and allows exits from a control block by naming the label of the block in an exit statement. We have found that such exits remove any need for the use of unrestricted go to's and thus believe they are well worth having. We suggest the EXIT statement be permitted to reference the label of an IF or SELECT statement as well as that of a looping structure. (36-
9. PRESENT Intrinsic: We have written some codes that contain a large number of options. It would be nice if these codes could be written to take advantage of the PRESENT intrinsic function, but in its present form it would not be useful for this application. What is needed is a way to ask if the "I<sup>th</sup>" argument is present and a way to get the value of the "I<sup>th</sup>" argument. (36-
10. IOSTAT Message: Frequently one wants to check the status of an I/O statement, and in some of the cases would like to print the message that would have resulted from the I/O (36-

statement in some of the error cases. Since there is no standard for the status values returned, there should be some mechanism for printing such messages, without requiring that the system default action be taken.

11. NAMELIST Comments: We use namelist input with very large data sets. We have found it useful to embed comments in the namelist input in order to keep track of the various data sets. We would like to see this supported in the standard. 36-11
12. Syntax for Iterative Control Structures: The Fortran DO loop is a loop. We suggest that the name LOOP be used for all of the new looping structures since it more clearly states the function. We believe it is desirable to qualify indexed loops with the keyword, FOR, and to provide a WHILE keyword form, since this models a frequently occurring construct that is provided in many procedural languages. Thus we would like to have the following forms: 36-12

```
DO 10 I = 1,7,2           (deprecated)
LOOP
LOOP FOR I = 1,7,2
LOOP WHILE (X > 0.1)
LOOP (5) TIMES
```

#### Overall Comments

We are concerned about the size of the language, but there is little that we would like to see removed. We do believe that the size of the language could be reduced if there were more regularity in the language, but don't see how to get there from where we are. With better support for abstract data types, see comment 7 above, we believe that much of the special baggage for handling arrays could be removed. The bottom line though is that the new features are sufficiently desirable that we would like to see something close to your current document approved, rather than living with Fortran 77 for the next 11 years.

## OTHER REVIEWER COMMENTS

1. The standard should include not only the indexed DO loop, but also the other most commonly used loop structures that test the truth of a condition. These are the DO WHILE which tests the condition at the end of the loop. These constructs should be included to improve readability of the source code and to encourage good programming practices. 36-1
2. The namelist standard should allow the user to include comments in the namelist input file. Without this feature we may have to continue to write our own namelist processor. 36-1 con

#37

43

Subgroup Name  
37-1 DATA

**James H. Matheny**  
41 Silver Spring Drive  
Rolling Hills Estates, CA 90274  
(213) 375-5940

December 7, 1987

X3 Secretariat/ CBEMA  
Attn: Fortran Public Review  
311 First Street, N. W.  
Suite 500  
Washington, D. C. 20001-2178

I believe that the Fortran Draft Standard -- X3J3/S8.104, June 1087 -- meets the needs and desires of the Fortran Users community. This community has many overlapping components, including:

- Professional programmers
- Casual users -- people whose expertise is other than programming
- Large scale scientific users
- Commercial users
- Systems programmers
- Micro computer users
- Novice users
- Experienced users

The emphasis of the draft Standard Fortran is, properly, on the needs of large scale scientific users. Some features needed by other classes of users have not been included in the draft Standard. Of these, the most significant, in my view, is the lack of a varying length character capability.

I PROPOSE THE ADDITION TO STANDARD FORTRAN OF A VARYING LENGTH CHARACTER CAPABILITY.

37

**DISCUSSION**

The idea is that, on definition, any character entity has a length. (The exception to this generalization is treated below.) With this facility, a user can treat words and sentences as they exist in normal English (or other natural language) normally, and not as padded by some specified but arbitrary number of blanks.

The syntactic form proposed is the Fortran 77 character statement specification without the inclusion of a length specification. (This has problems dealt with below.) Operations on character entities are permitted whether the length attribute is varying or specified, and concatenation between character entities of either

attribute is defined and permitted. For example:

```

CHARACTER (LEN = 10) A, B    ! Fortran 8X form
CHARACTER *10 C              ! Fortran 77 form
CHARACTER D, E               ! Proposed form
!
D = "abc" ; C = ' defg'
E = D // C // A // B        ! Valid operation, LEN(E) = 33

```

Character functions, intrinsic functions, and substrings are permitted without added restrictions.

The proposed facility requires heap memory management for the object program. Years ago this was considered to be too radical, but now heap memory management is required by the ALLOCATE/DEALLOCATE mechanism of Fortran 8X, so there is little additional complexity added in the compiler or in the run-time environment. (Of course, an implementation need provide this heap memory management only when the source program(s) require it.)

This proposed facility is common practice. Many other languages require or allow heap memory management. A varying length character capability (in a fixed allocation) has been implemented as an extension to several Fortrans, including the CSC Infonet Fortran 77. A true varying character facility, as proposed (with a different syntax) has been in the General Electric Information Systems Fortran for more than ten years.

Calling out known problems with the proposal:

The syntax proposed stumbles across a "mistake" in Fortran 77. If a \*LEN is not specified, the default length is defined to be 1. This proposal uses this form to specify varying length character. However most existing Standard-conforming programs that use this syntactic form would work giving the same (correct) result, but using the varying character capability. The programs that would be incompatible are those that use the truncation capability of Fortran 77 to extract a first character from a string. This is better accomplished using the Fortran 77 substring notation.

The length of a varying length character entity on definition is not immediately known for unformatted input or for formatted input where the edit descriptor is A without a w (width) specification. Thus there are times when the user must specify explicitly the desired length. It is also useful for the user to be able to change a length.

A syntactic form to resolve this problem could be the



existing LEN intrinsic function, but permitting it to appear on the left of an equals sign on assignment, and other places (READ, e. g.) where definition occurs. This is (to me) a natural syntax, and is used in the CSC INFONET Fortran. However this syntax violates consistency criteria of language design, and would require a substantial rewrite of Section 12 of the draft Standard.

Proposed here is an added SET\_LENGTH statement to provide this functionality.

SUGGESTED TEXT (Words delineated by \*\*\* are not a part of the proposal)

1. Page 2-3, line 24+. Add line:

or *set-length-stat*

2. Page 2-9, lines 15-16. Add clause: ... precision objects ", character type objects with unspecified length, " ... must not appear in a storage association context.

3. Page 4-1, line 17. "has" ==> "may have".

4. Page 4-5, line 21. "The length is a type parameter; its" ==> "The length may be a type parameter, or it may be determined when a character entity is defined. Its"

5. Page 5-4, lines 8-9. If neither ... is specified, "the length of the data entity is 1." ==> "the length of the data entity is determined on definition of the data entity, or by a SET\_LENGTH statement."

Page 5-4, line 32+. Add line:

CHARACTER D, E ! Length determined on definition.

6. Page 5-9, line 10. Change: "... character array, character array element, or a..." ==> "... character array with a specified length, character array element, or a ...". \*\*\*assumed size array\*\*\*

Page 5-9, line 14+. Add:

(4) If the actual argument is a character array without a specified length, the size of the dummy array is the sum of the array element sizes.

7. Page 5-18, line 24. Add clause: A data object of type character "with specified length" may be equivalenced ...

Page 5-18, Line 26. Add sentence: "A data object with unspecified length cannot be equivalenced."

8. Page 5-19, line 29. Add clause: ... an array of zero size, "a character string of unspecified length, " a character string of zero length, ... **\*\*\*COMMON statement\*\*\***

9. Page 6-2, line 39+. Add section, renumber BNF:

6.1.3 **SET\_LENGTH** Statement. The **SET\_LENGTH** statement provides a means to set the length of a character entity when the character type declaration does not include a specific or passed length.

R609+ *set-length-stat* is **SET\_LENGTH** (*char-variable* = *integer-expr*)...

Constraint: *char-variable* must not have a *length-selector*.

10. Page 9-4, line 13. Add clause: The file is a character variable "with specified length". **\*\*\*internal file disallowed\*\*\***

Page 9-4, line 13. Add clause: A record of an internal file is a scalar character variable "with specified length".

Page 9-4, line 37+. Add:

(3) An internal file may not be specified as a character variable with unspecified length.

11. Page 9-16, line 17. Add clause: If an entity in the input list is of type character "with specified length", the length ... **\*\*\*unformatted input\*\*\***

Page 9-16, line 18. Add sentence: If an entity in the input list is of type character without a specified length, the length of the character input list item must have been defined by the **SET\_LENGTH** statement (6.1.3).

12. Page 10-8, line 30+. Add paragraph:

For input, if a field width *w* is not specified with the A edit descriptor, the length of the character input list item must have been defined by the **SET\_LENGTH** statement (6.1.3).

13. Page 14-7, line 31+. Add paragraph and renumber:

(1+) Execution of a character intrinsic assignment where the length is unspecified causes the length of the variable to become defined.

Page 14-7, line 33+. Add paragraph and renumber:

(2+) Execution of a masked array character assignment where the length is unspecified may cause some or all of the lengths of the array elements in the assignment statement to become defined.

Page 14-7, line 36+. Add paragraph and renumber:

(3+) As execution of an input statement proceeds, each variable of type character with unspecified length that is assigned a value from the input file causes the length to be defined if the input statement is neither an unformatted READ nor a formatted READ using the A edit descriptor with no w specification. For the unformatted and A edit descriptor with no w case, the length must have been defined prior to the execution of the input statement.

Page 14-8, line 19+. Add paragraph and renumber:

(14+) The SET\_LENGTH statement causes the lengths of the specified character variables to become defined.

14. Page C-4, line 42+. Add paragraphs:

All character objects have a length. When the length is specified in the character type statement, the length is the value provided, or that passed from a calling program. All elements of an array have this length. Storage association is permitted.

When the user does not specify a length in the character type statement, the length is determined at each definition of the character entity. The lengths of elements of an array are determined at the definition of each element of the character array, and are maintained individually. Normally storage is allocated on a heap, so storage association is disallowed.

Expressions and intrinsic functions of type character can contain either or both forms of length specification. The length of a character expression is the sum of the component lengths.

For two forms of input, the length of a datum is not necessarily known by the program or in the data. These forms are the unformatted READ and the formatted READ using the A edit descriptor with no width specification. The SET\_LENGTH statement permits the definition of the length. This usage also permits truncation or extension of a character datum, but does not provide blank padding -- the extended character positions are undefined.

43

# 38

Northern Illinois University  
DeKalb, Illinois 60115

Department of Computer Science  
815 753 0378

Bitnet: T90NWR1@NIU  
Dec 7th, 1987.

Public Comments for Dpans Fortran Revision  
X3 Secretariat  
ATTN: Gwendy Phillips  
Computer and Business Equipment Manufacturers Association  
Suite 300  
311 First Street, N.W.  
Washington, DC 20001-2178

Dear Sirs,

I am strongly opposed to the proposed Fortran 8x standard. I enclose a set of detailed comments on some of the features of which I disapprove.

Sincerely,  
*Neil W. Rickert*  
Neil W. Rickert,  
Professor of Computer Science.

cc: Public Comment for Dpans Fortran Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY 10018

*Subgroup Nominations:*

- 38-1 GEN
- 38-2 PROC
- 38-3 DATA
- 38-4 GEN
- 38-5 DATA
- 38-6 ~~GEN~~ GEN
- 38-7 GEN
- 38-8 GEN

X3  
DEC 11 11 00 AM '87

Comments on the proposed Fortran 8x standard.

Neil W. Rickert,  
Department of Computer Science,  
Northern Illinois University,  
DeKalb, IL 60115  
Bitnet: T9@NWRI@NIU

1. Introduction.

These comments are based on the description of Fortran 8x given in Draft S8, Version 104. All bracketed references (for example [2.4.1]) are references to paragraphs of that document.

My initial reaction to the report was that if the standard were adopted there would be one of two possible outcomes; either the FORTRAN language would effectively be killed<sup>1</sup>, or the new standard would be massively ignored with most programmers continuing to use FORTRAN 77 (or even FORTRAN 66).

On first reading the proposed standard reads as if designed by computer language theoreticians who have rarely programmed in FORTRAN, and who certainly have never liked the language. This is particularly apparent in the discussion of deprecated features, where the rationale given for eventual elimination of some features reveals a surprising ignorance about how these features are used in practice.

I shall divide most of my criticism into two sections; a section on added features, and a section on deprecated features of the language. My commentary will concentrate on the aspects I find most negative. Let me make it clear, however, that I do not find everything negative. It is high time, for example, that support for user defined data structures was included in FORTRAN:<sup>2</sup>

Let me first comment on a feature of FORTRAN which is eliminated in the proposed standard. For although the standard specifies that "no FORTRAN 77 features have been deleted" [OVERVIEW], I find that under the new standard FORTRAN has lost its simplicity<sup>2</sup>. This is indeed a grievous loss, since for many users (in particular physicists, chemists and other non computer scientists) this simplicity was a major attraction of the language.

2. Proposed new features.

There are some new features (for example support for dynamically

---

<sup>1</sup>Some computer scientists have been pushing for the demise of FORTRAN for years.

<sup>2</sup>In honesty, the loss of simplicity began with FORTRAN 77, which made an unfortunate and excessively complex choice in the implementation of character data. The representation of a character string as an array of one-byte integers (as in the language C) would have been a simpler choice, and one more suited to the FORTRAN programming style.

Comments on the proposed Fortran 8x standard.

dimensionable arrays) which are long overdue. On some new features I am neutral, while on others I am mildly negative. I do not discuss any of these in this section. Instead I concentrate my comments on the features to which I am strongly opposed.

An additional concern, which I mentioned in the introduction, lies in the added complexity caused by the number of new features added to the language. If users really want a highly complex language they can already choose ADA or PL/I. They choose FORTRAN because of its simplicity and because of its efficiency when used with good optimizing compilers. Any new standard which threatens this simplicity and efficiency is surely unwise. } 38

Many of the proposed changes in Fortran 8x read as if taken from the PL/I language. If these changes were as important to the future of the language as the standards committee seems to believe, scientific programmers should have been flocking to PL/I in the 1970s. No such mass emigration occurred. A few scientific programmers did briefly experiment with PL/I, but soon drifted back to the FORTRAN world (or in some cases migrated to C). If members of the committee had attempted extensive scientific programming in PL/I they might have discovered why. For the additional cost in complexity, lengthened compile time, longer object code and slower running times were far more than the new features were worth.

(a) Internal Procedures. 38-2

It may come as a surprise that I would oppose internal procedures. It is certainly true that they can simplify programming, and allow more effective information hiding when that is appropriate. Yet I believe that the lack of such a feature has been a major strength of FORTRAN.

In current FORTRAN the separately compilable external subprogram is the principal structuring technique available. The natural result is that programmers automatically organize their code into separate subprograms. I believe that it is as a direct result of this that FORTRAN has developed such extensive libraries of reusable subprograms. And it is generally agreed that these libraries are a major asset of the FORTRAN community.

I cannot prove my claims. They are based partly on instinctive feelings about programmer psychology. However there is some support. For other languages have not developed the extensive libraries that are so apparent in FORTRAN. Although COBOL is almost as old as FORTRAN, substantial libraries have not developed. This is surely due to strong

---

<sup>3</sup> I shall keep returning to this central point of efficiency and simplicity. I know that many computer scientists dismiss efficiency, on the basis that ever increasing speeds of processors eliminate any need for concern. But in the FORTRAN world this view is surely mistaken. FORTRAN programs are constantly pushing the limits of our computing ability to solve ever more difficult problems. Simplicity of the language is equally important, for with increasing language complexity comes increasing compilation time. It is common for FORTRAN programs to need frequent recompilation as algorithms are experimented with at the forefront of scientific computing.

dimensionable arrays) which are long overdue. On some new features I am neutral, while on others I am mildly negative. I do not discuss any of these in this section. Instead I concentrate my comments on the features to which I am strongly opposed.

An additional concern, which I mentioned in the introduction, lies in the added complexity caused by the number of new features added to the language. If users really want a highly complex language they can already choose ADA or PL/I. They choose FORTRAN because of its simplicity and because of its efficiency when used with good optimizing compilers. Any new standard which threatens this simplicity and efficiency<sup>3</sup> is surely unwise. } 38

Many of the proposed changes in Fortran 8x read as if taken from the PL/I language. If these changes were as important to the future of the language as the standards committee seems to believe, scientific programmers should have been flocking to PL/I in the 1970s. No such mass emigration occurred. A few scientific programmers did briefly experiment with PL/I, but soon drifted back to the FORTRAN world (or in some cases migrated to C). If members of the committee had attempted extensive scientific programming in PL/I they might have discovered why. For the additional cost in complexity, lengthened compile time, longer object code and slower running times were far more than the new features were worth.

(a) Internal Procedures. 38-2

It may come as a surprise that I would oppose internal procedures. It is certainly true that they can simplify programming, and allow more effective information hiding when that is appropriate. Yet I believe that the lack of such a feature has been a major strength of FORTRAN.

In current FORTRAN the separately compilable external subprogram is the principal structuring technique available. The natural result is that programmers automatically organize their code into separate subprograms. I believe that it is as a direct result of this that FORTRAN has developed such extensive libraries of reusable subprograms. And it is generally agreed that these libraries are a major asset of the FORTRAN community.

I cannot prove my claims. They are based partly on instinctive feelings about programmer psychology. However there is some support. For other languages have not developed the extensive libraries that are so apparent in FORTRAN. Although COBOL is almost as old as FORTRAN, substantial libraries have not developed. This is surely due to strong

---

<sup>3</sup> I shall keep returning to this central point of efficiency and simplicity. I know that many computer scientists dismiss efficiency, on the basis that ever increasing speeds of processors eliminate any need for concern. But in the FORTRAN world this view is surely mistaken. FORTRAN programs are constantly pushing the limits of our computing ability to solve ever more difficult problems. Simplicity of the language is equally important, for with increasing language complexity comes increasing compilation time. It is common for FORTRAN programs to need frequent recompilation as algorithms are experimented with at the forefront of scientific computing.

Comments on the proposed Fortran 8x standard.

It is often the case with scientific programming that the appropriate degree of precision is highly dependent on the data. It is sometimes useful to compute an answer in both single and double precision, then compare the results to get a crude estimate of the amount of round off error. Imagine a Fortran 8x program which does this by computing the answers to both 7 and 14 digits of precision. If this program happens to be run on hardware where both precision specifications are mapped into the same hardware precision, the remarkable agreement of the two results might be falsely interpreted as indicating very low round off.

With the new standards, imagine the effect on say the IMSL library of mathematical subprograms. Currently many routines in this library are available in both single precision and double precision versions (with different names for the two versions). If this library is to maintain complete portability and be compatible with Fortran 8x, it will need to come in perhaps 19 versions, for precisions of 6,7,.....,15. Imagine the complexity of finding suitable naming conventions, particularly since the first character in a subprogram name must not be a digit. Of course in reality there would only be two different versions, since each version will map into either single precision or double precision on the host computer. But which is which will depend on the computer system.

The committee is creating a Frankenstein's monster.

The intended functionality of the proposed new floating point specifications could be more effectively provided with a suitable preprocessor feature.

(c) The proposed array feature.

38-4

Here we have a set of new features which superficially appear to offer great benefit to the programmer. Certainly the ability to create dynamically dimensioned arrays is long overdue. But most of the new features bring with them costs (in terms of extra complexity and loss of efficiency) which far outweigh the benefits.

The new features mostly echo facilities already available in PL/I, where they really don't work all that well. Implementing them requires generating array descriptors (dope vectors) which are passed along with arguments to subprograms. If the PL/I experience is any indication this will result in longer object code, lengthier linkage code sequences, slower access to array elements due to the need of compiler generated code to repeatedly examine the dope vectors. Indeed the situation will be worse than in PL/I, since the deprecated features of FORTRAN must still be supported. Thus when X(I) is passed as an argument, the calling program must assume that it could be referenced as an array dummy argument, so a dope vector must be constructed and passed to the called program. Then, to maintain compatibility, dope vectors will probably also be needed when purely scalar arguments are used. What a mess!

Several years ago I wrote a simple matrix multiplication subroutine in both FORTRAN and PL/I. I compiled the FORTRAN version on IBM's H-extended compiler with maximum optimization. I used that vendor's PL/I optimizing compiler with full optimization, and with the REORDER option, to compile



the PL/I version. I then examined the code generated for the innermost loop. The difference was disturbing. I have done little serious scientific programming in PL/I since that time.

I also experimented in PL/I with whole array operations on cross sections of an array, believing that this might result in better code. Imagine my surprise when I found that the compiler was often allocating scratch storage for temporary arrays in intermediate steps of the computation. Perhaps this indicates a failure of the PL/I compiler developers, but I suspect it is really a result of the enormous increase in complexity. I notice a statement (in [7.5.1.5]) to the effect that in array assignments the entire right hand side is evaluated before any assignment is made to the target. In many circumstances this specification can only be guaranteed by first making the assignment to a temporary array, then copying the temporary array to the destination.

An important factor in scientific computing today is the existence of high performance parallel computers, vector processors, etc. There is ongoing research, particularly in FORTRAN, in developing compilers which can recognize parallelizable constructs which can then be compiled into code which takes greatest advantage of the new hardware. Some of the new array features will greatly hinder this development.

(d) The PARAMETER statement.

38-5

Here my objection is not to the concept embodied in this statement, but in the use of the word PARAMETER. Unfortunately this word is hopelessly ambiguous, and hence potentially confusing. Most computer scientists use the word parameter as the equivalent of the FORTRAN term "dummy argument". On the other hand, what physicists and mathematicians mean when they refer to a parameter can perhaps best be described as a "variable" which varies less often than other variables, or perhaps as a "constant" whose value can be varied from time to time. The meaning used by scientists is thus more closely approximated by a value passed to subprograms via a variable in named COMMON. These two meanings of parameter are not only mutually contradictory, but also conflict with the proposed use for the term for Fortran 8x.

We must remember that the FORTRAN programmer is likely to spend some time talking with computer scientists, and some time talking with scientists and mathematicians. Communication will be clearer if a common set of terminology is used. Surely a better term than PARAMETER can be found. I would prefer the term NAMED CONSTANT.

### 3. Deprecated features.

In commenting on deprecated features, I discuss them as if they will be removed from future versions of the language. This appears to be the long range expectation of the committee. It is true that these features are not immediately removed, and if users so insist they may never be removed. But, one may surely ask, does a committee which so misunderstands the importance of these features retain any credibility as a representative of the interests of the FORTRAN community?

P. 245

Some of my objections can be inferred from my comments on new features. Thus I obviously do not accept the committee's rationale on the statement function, since I am opposed to the addition of internal procedures. Most of my minor disagreements with the list of deprecated features will go unmentioned; I instead concentrate on my most serious objections.

(a) The computed GOTO. 38-6

The committee suggests that the CASE statement can serve as a replacement for the GOTO. It is certainly true that I can replace

```
GOTO (100,200) L
```

by

```
SELECT CASE (L)
CASE DEFAULT
GOTO 10
CASE (1)
GOTO 100
CASE (2)
GOTO 200
END SELECT
10 CONTINUE
```

but I hardly think that this was the intention of the committee.

I suspect that the committee is under the misimpression that the computed GOTO is used only to create unmaintainable spaghetti code. While it is undeniably so used, it is also used in relatively well structured programs. Consider the following example:

```
CALL SUBR(IERROR,N,X,A)
GOTO (900,950) IERROR
... normal coding in the absence of errors
...
900 Code to print messages and statistics for error 1
...
950 Code to print messages and statistics for error 2
...
```

This example actually provides a reasonably structured approach to gracefully backing out from errors. The use of an integer argument for subroutines to return error indications is common throughout the available scientific subroutine libraries. The use of GOTOs here is surely preferable to the CASE statement; for if a routine calls several subroutines in this manner, a GOTOless approach could require deeply nested CASE statements.

(b) The EQUIVALENCE statement. 38-7

In the draft standard [B.3.1.6] several uses are given for the EQUIVALENCE statement, with suggested replacements. Let me give you an additional use. The EQUIVALENCE statement makes it possible to do list

Comments on the proposed Fortran 8x standard.

processing in FORTRAN. List processing is typically done by using a large array of integers as a pseudo-memory, with the array index as the equivalent of a pointer variable. The EQUIVALENCE statement allows storing of a variety of data types into list structures built in this manner.

I have personally served as a consultant on a large list processing program which modelled electric utility rates. I know of compilers and of parser generators written in FORTRAN; these programs surely do some kind of list processing. I suspect there are many list processing FORTRAN programs out there, often written by programmers who have never heard of the term list processing.

Without the EQUIVALENCE statement it would be difficult to write these programs in Fortran 8x, since pointer data types are not introduced. Even with pointer data types, list processing in an array is often preferable, for it allows the list structures to be written to external memory (disk or tape), and to later be reused without requiring exact address matching.

If anything, the EQUIVALENCE statement is badly in need of more functionality. Indeed the ability to use

```
EQUIVALENCE (A(I),B)
```

(where it is understood that references to B would use the then current value of the index I) would enhance the portability and reliability of the EQUIVALENCE statement.

(c) Passing a scalar to a dummy array. 38-8

This has turned out to be a very powerful feature of FORTRAN. I agree that other proposed features of Fortran 8x allow alternate methods of achieving the same goal, but (as I have described above) at an enormous cost in increased complexity, object code length, subprogram linkage cost, and run time efficiency. I consider the following example a typical use.

```
J=N
DO 188 I=1,N
  CALL SUBR(I,X(J),A(J,J))
  J = J-1
188 CONTINUE
....
SUBROUTINE SUBR(N,X,A)
DIMENSION X(N),A(N)
  code to do one step of the back substitution in
  solving a lower triangular linear system
...
```

With modest values of the array dimensions, the execution time of the equivalent program could easily double if the Fortran 8x array features are used to replace the deprecated feature. It is true that for very large dimensions the efficiency loss may be much less serious, but in many applications linear algebra problems of small dimension are solved perhaps millions of times.

p. 247

#### 4. Summary.

As the saying goes, if it ain't broken, don't fix it. The standards committee seems to believe that FORTRAN is broken. Indeed the proposed standard appears to come from a belief that FORTRAN is near death and needs a massive transfusion and multiple organ transplants. This will surely come as a shock to the troops out in the FORTRAN programming field, who find that FORTRAN is still alive and quite vigorous. It is still true that the FORTRAN compiler is amongst the first developed for new hardware.

For some time now there has been concern over the reliability of programs. As part of this concern FORTRAN has come into criticism for having a number of dangerous features. I understand the committee's concern, and its desire to eliminate these features. There is a growing attitude in some parts of the computer science community that the language must protect the programmer from making mistakes. I vigorously disagree. Good programmers can write sound reliable programs in any language. Poor programmers can write unreliable software no matter how protective the language. The enemy of reliable software is now, as ever, excessive complexity.

My greatest objection to Fortran 8x is that it is philosophically a quite different language from FORTRAN. By way of example the COMMON area and the DO loop are central to the FORTRAN tradition, yet in Fortran 8x they are marked as deprecated features in their well known form. As a philosophically distinct language Fortran 8x should be released under a different name, and not pretend to be a language it is not. That would give scientific programmers the opportunity to choose whether or not they desired these features. It is my belief that they would overwhelmingly reject the new language, much as they rejected PL/I.

If the X3J3 committee believes that future FORTRAN programs will only be written by skilled computer scientists who are well versed in structured programming, abstract data types, information hiding, etc, then they are surely victims of self delusion. As ever many FORTRAN programs will be written by physicists, mathematicians and others whose knowledge includes a mere smattering of computer science.

p. 248

#39

43



THE UNIVERSITY OF TEXAS AT DALLAS

BOX 830688 RICHARDSON, TEXAS 75083-0688 (214) 690-2445

CENTER FOR LITHOSPHERIC STUDIES

X3J3 Chair

X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Subgroup Nominations:

39-1	GEN	39-8	DATA
39-2	GEN	39-9	CIO
39-3	GEN	39-10	DATA
39-4	CIO	39-11	DATA
39-5	PROC	39-12	GEN
39-6	DATA	39-13	GEN
39-7	DATA	39-14	GEN

Dear Sir; .

As the director of the Center for Lithospheric Studies at the University of Texas at Dallas, I represent a group of approximately 40 scientific applications programmers whose main working language is FORTRAN. We view, with some apprehension, many of the changes proposed in 8x. Our main concern is the lack of backward compatibility that will occur with the removal of current features, especially COMMON and EQUIVALENCE. What this means in effect is that 8x is a new language, not a revised FORTRAN as all existing software will have to be rewritten; this is an unacceptable cost.

While we approve of a few of the proposed changes, such as the ability to allocate arrays, the use of inline comments, and the NAMELIST I/O, most of the others appear to be of dubious merit. We anticipate serious problems with dependent compilation and abstract definition of numerical precision. Other features that would be good additions, but are not included, are BIT and POINTER data types, a DO WHILE, a variable length CHARACTER data type, and records like VMS FORTRAN.

39-2

39-4

39-6

39-9

39-10

39-1

39-3

39-7

39-8

39-11

NOV 30 1987

GLOBAL ENG.

P. 249



39-12

Other general comments we wish to make are that slower compilers and applications are undesirable and that too many new features have been proposed. Most of the proposed changes do not appear to have been evaluated from the point of view of the users or the institutions that are currently using FORTRAN. While many of the new features do have some merit, these would be better realized by presenting the package as a new language. The disruption that would result from acceptance of the current draft proposal as the FORTRAN standard would far outweigh the benefits.

39-13

39-14

Regards,



George A. McMechan, Director,  
Center for Lithospheric Studies

Subgroup Nominations:

- 40-1 GEN
- 40-2 PROC
- 40-3 PROC
- 40-4 GEN
- 40-5 GEN

X3J3 SECRETARIAT  
NOV 30 1987

#40  
43

NOVEMBER 23, 1987  
MIS DEPARTMENT  
TRW VALVE DIVISION  
1455 EAST 185 STREET  
CLEVELAND, OHIO 44110

PUBLIC COMMENT FOR DPANS FORTRAN REVISION  
X3 SECRETARIAT  
ATTN: GWENDY PHILLIPS  
COMPUTER AND BUSINESS EQUIPMENT MANUFACTURERS ASSOCIATION  
SUITE 500  
311 FIRST STREET, NW  
WASHINGTON, DC 20001-2178

SUBJECT : USER COMMENTS ON FORTRAN BX PROPOSAL BY X3J3 STANDARDS COMMITTEE.

DEAR MS. PHILLIPS:

I AM WRITING IN RESPONSE TO A MAILING WHICH WAS SENT BY IBM'S LANGUAGE PRODUCTS MANAGER, MILLIE CLARK, TO ALL CURRENT IBM FORTRAN LICENSEES. THE MAILING CONTAINED A SUMMARY OF A PROPOSED STANDARD FOR FORTRAN BX, WHICH IS BEING DEVELOPED BY THE X3J3 FORTRAN STANDARDS COMMITTEE. THE MAILING COVER LETTER REQUESTED THAT FORTRAN USERS SEND THEIR COMMENTS ON THE STANDARD TO A.N.S.I., AND TO THE X3J3 COMMITTEE. BECAUSE I AM THE PRIMARY FORTRAN USER AT THIS DIVISION, I AM WRITING THIS RESPONSE, ALTHOUGH THESE COMMENTS ARE MINE AND DO NOT REFLECT AN OFFICIAL COMPANY POLICY.

MY COMMENTS ARE CENTERED AROUND THREE POINTS:

1. I AM IN FAVOR OF UPDATING THE FORTRAN LANGUAGE.
2. I AM NOT IN FAVOR OF DELETING CURRENT FORTRAN STATEMENTS OR PROPERTIES.
3. I AM GENERALLY IN FAVOR OF A NEW STANDARD LIKE THE FORTRAN BX STANDARD, BUT BECAUSE OF THE DISAGREEMENTS AND CONFLICTS MENTIONED IN THE NEGATIVE BALLOTS INCLUDED WITH THE STANDARDS MAILING, I AM NOT IN FAVOR OF GOING AHEAD WITH IMPLEMENTATION AT THIS TIME.

ON POINT NUMBER 1, I AM IN FAVOR OF EXPANDING THE FUNCTIONS OF THE FORTRAN LANGUAGE. ALTHOUGH I AGREE WITH THE BALLOT OPINIONS WHICH SAY THAT FORTRAN IS A SIMPLE LANGUAGE AND SHOULD STAY THAT WAY, THERE ARE FUNCTIONS WHICH COULD BE ADDED TO MAKE THE LANGUAGE EASIER TO WORK WITH, WHILE STILL KEEPING THINGS SIMPLE. THESE CHANGES INCLUDE -

- A. LONGER VARIABLE NAMES. THE CURRENT LIMIT OF NAME SIZE TO 6 CHARACTERS ← (40-1) MAKES IT VERY DIFFICULT TO COME UP WITH MEANINGFUL NAMES IN ANYTHING OTHER THAN THE SIMPLEST OF PROGRAMS. I SUGGEST THAT A NAME SIZE OF 12 CHARACTERS, INCLUDING DASHES, GIVES A BETTER BALANCE OF SIMPLICITY AND DESCRIPTIVENESS.
- B. A PERFORM-TYPE VERB FUNCTION. THIS FUNCTION WOULD MAKE IT MUCH EASIER TO ← (40-2) WRITE PROGRAMS USING A STRUCTURED APPROACH, INSTEAD OF THE GO-TO TYPE OF LOGIC WHICH IS COMMON IN MANY FORTRAN PROGRAMS.
- C. MODULAR DATA AND LOGIC DEFINITIONS, AS DESCRIBED IN THE BX STANDARD. THE ← (40-3) SIMPLE ADDITION OF A "COPY" TYPE COMMAND WOULD BE HELPFUL. WHEREVER A COMMON LOGIC SECTION OR DATA ACCESS DEFINITION IS USED, A "COPY" COMMAND WOULD MAKE IT MUCH EASIER TO CREATE AND UPDATE THAT AREA.

NOVEMBER 23, 1987 pg 2 of 2

I AM IN FAVOR OF THESE CHANGES BECAUSE THEY WOULD MAKE IT EASIER TO WRITE FORTRAN PROGRAMS, BUT MORE IMPORTANTLY, THEY ALLOW A PROGRAM TO BE WRITTEN WHICH IS EASIER TO MAINTAIN. IN MY JOB, I AM RESPONSIBLE FOR UPDATING AND MAINTAINING A SET OF PROGRAMS WHICH INCLUDE FORTRAN AND COBOL LANGUAGE SUBROUTINES. THE CHANGES WHICH I SUGGEST ABOVE ARE FEATURES OF COBOL WHICH I AM CONVINCED MAKE COBOL PROGRAMS MUCH EASIER TO MAINTAIN. EVEN THOUGH THE ADDITION OF THESE FEATURES INCREASES THE TIME NEEDED TO COMPILE AND RUN THE PROGRAMS, I BELIEVE THAT THE SAVINGS IN PROGRAMMING TIME WOULD OFFSET ANY LOSSES.

ON POINT NUMBER 2, I AM NOT IN FAVOR OF DELETING CURRENT FORTRAN CAPABILITIES. I SAY THIS FOR THE REASON THAT THERE ARE MANY EXISTING FORTRAN PROGRAMS WHICH WOULD HAVE TO BE CHANGED TO SIMPLY STAY COMPATIBLE WITH NEW SYSTEMS. ANY TIME WHICH IS SPENT CHANGING OLD PROGRAMS WITHOUT IMPROVING THEIR FUNCTION IS TIME WHICH COULD HAVE BEEN SPENT ON NEW OR IMPROVED APPLICATIONS. IF THERE IS A GOOD REASON FOR ELIMINATING OLD FEATURES, THE CHANGE SHOULD BE MADE GRADUALLY. FOR EXAMPLE, THE OLD FEATURE COULD BE DISALLOWED IN ANY NEW PROGRAMS OR CHANGES FOR A PERIOD OF TIME, AND THEN MADE OBSOLETE AT SOME POINT IN THE FUTURE. THE X3J3 RESPONSE TO IBM'S BALLOT INDICATES THAT NOTHING WILL BE OBSOLETE BEFORE THE YEAR 2010, WHICH WOULD ALLOW TIME FOR MANY OF THE UNDESIRABLE FEATURES TO DIE A NATURAL DEATH AS THE PROGRAMS WHICH THEY ARE IN ARE REPLACED.

ON POINT 3, I AM IN FAVOR OF A NEW STANDARD LIKE THE 8X STANDARD. I AM NOT IN FAVOR OF THE STANDARD BEING SUBMITTED FOR IMPLEMENTATION UNTIL A NUMBER OF DISAGREEMENTS OR MISUNDERSTANDINGS WHICH ARE REFERRED TO IN THE NEGATIVE BALLOTS ARE WORKED OUT. A KEY PROBLEM SEEMS TO BE THAT THERE IS A DISAGREEMENT AS TO JUST HOW MUCH CHANGE AND WHAT KIND OF CHANGE SHOULD TAKE PLACE IN THE LANGUAGE. I HAVE LISTED MY KEY REQUESTS ABOVE, AND SUGGEST THAT ANY OTHER USER SUGGESTIONS BE TAKEN INTO ACCOUNT, ALONG WITH THE SUGGESTIONS OF THE "EXPERTS" AT IBM, DEC, AND SO ON. A UNANIMOUS AGREEMENT WILL NEVER BE REACHED, BUT THERE IS STILL TOO MUCH DISAGREEMENT. THERE ARE SOME PROBLEMS WHICH MAY BE ONLY MISUNDERSTANDINGS. A GOOD EXAMPLE IS THAT THE IBM BALLOT OBJECTS TO WHAT IS ASSUMED TO BE NEAR-TERM OBSOLESCENCE OF LANGUAGE FEATURES, WHILE THE X3J3 REBUTTAL CLAIMS THAT NO MAJOR DELETIONS WOULD TAKE PLACE BEFORE THE YEAR 2010. THESE AND OTHER PROBLEMS NEED TO BE HASHED OUT, AND THEN SOME SORT OF AGREEMENT COULD BE REACHED.

YOURS TRULY,

*Bob Zoller*

BOB ZOLLER  
PROCESS ENGINEER/PROGRAMMER  
TRW VALVE DIVISION

cc: MR. ROBERT J. ANTHONY  
TRW VALVE DIVISION.

*P. 252*





CONVEX

~~41~~ #41  
43

CONVEX COMPUTER CORPORATION 701 PLANO ROAD RICHARDSON, TEXAS 75081 (214) 952-0200

'87 DEC 14 PM 3:34

Subgroup Nominations:

41-1 CIO	41-5 GEN
41-2 PROC	41-6 GEN
41-3 GEN	41-7 GEN
41-4 DATA	41-8 GEN

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

The X3J3 committee has done a lot of hard work on the proposed FORTRAN 8x standard over the last few years. Many of the new ideas are good extensions to the FORTRAN language such as the CASE statement. Unfortunately, the committee has gone too far in making changes for a single update of the FORTRAN standard and has produced the draft for a new language. This lack of focus and scope is probably a key contributor to the standard being 5 years late at this time. The fact that the proposed standard is late, is *NOT* grounds for approving the proposal as a standard without proper review and correction of items found to be objectional during the review period.

I believe that the MODULE/USE feature and many of the other changes that are proposed make FORTRAN too large and too complex to use. I understand the desire to make FORTRAN a more "modern" language, but FORTRAN has always been a simple language that could be used by professional engineers to get their job done with a minimum of effort. I believe the proposed standard goes against the spirit of the FORTRAN language.

I am surprised that a definition of the POINTER data type and the INCLUDE statement are missing from the proposed standard. It was my understanding that the charter of the committee was to standardize existing practice. POINTERS, the INCLUDE statement, and many other statements are part of existing compilers, but in many cases the syntax is slightly different. The goal of a standard is to make code more portable between different manufacturer's machines. I would suggest a major redirection of the committee to work on the issue of portability of the language between vendors and not on inventing a new language.

I also feel that it is unacceptable to consider deleting statements from any future standard. While this may be viewed by the committee as an evolutionary process, it is viewed by organizations with large volumes of code developed over a period of many years as a threat. The threat is that some committee will removed those constructs from a future standard, thus

requiring expenditures of large sums of money to convert and requalify many older products. This will be a costly undertaking for minimal gain and the risk of major problems in restabilizing and productizing application packages. If I were faced with such a major rework project just to make the move to a more modern language, I would certainly consider moving to a language with a more stable definition such as C. I would suggest that the committee divide the standard into the New FORTRAN language standard and the Current FORTRAN language standard. Standardize the existing practice, add enhancements, and make the Current FORTRAN standard better. Define the New FORTRAN in the New FORTRAN standard. I agree with IBM's ballot comments on this subject.

41  
co

I also believe that it is a requirement for the committee to assess the economic impact that changes in any standard may have on the people who use that standard. The fight over COBOL 8x was not really a "we don't want change" fight, but rather a fight over the economic impact that the proposed COBOL 8x changes would have on the development and sustaining of code. The economic impact of the proposed FORTRAN 8x changes is also *too* great. I believe that such a major change in model of the language violates section 5.3 of the X3J3 Project Proposal which states, "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." The amount of new syntax, new syntax rules, and the dependent compilation model make this a complex language that will be complex to implement. So, both the implementor and the user will have a major economic impact as a result of this proposed standard. The implementor as a result of the dramatic increase in complexity of the language model and the user as a result of potentially having to redo large amounts of code because of possible deletion of features in future standards. If X3 and the X3J3 committee does not take these concerns into account and minimize the potential impact, then another COBOL 8x type problem looms in the future.

I would urge the committee to review the potential economic impact and adopt the IBM proposal as stated in their ballot comments as a way to fix the problem. If the committee does not chose to take such steps, then I believe it is the duty of the X3 committee and the Standards Review Board to force such a redirection to occur.

Sincerely,

Presley Smith  
Manager, Development Software

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

(43) #42  
 Subgroup Nominations  
 42-1 DATA | 42-7 GEN  
 42-2 GEN | 42-8 GEN  
 42-3 GEN | 42-9 GEN  
 42-4 DATA | 42-10 PROC  
 42-5 CIO | 42-11 PROC  
 42-6 DATA | 42-12 GEN

Dear madam;

I would like to register a few of my thoughts on the Fortran 8X specifications I have worked with Fortran for 10 years in a variety of jobs and I would like to continue using Fortran in the future. Unfortunately, I have been studying computer science at the University of Florida for several years and I feel a need for a better language than Fortran77. The proposed standard does not satisfy my needs. Here is my list of comments:

1. I feel that several 8X constructs are too complex. Examples are the declarations using user-defined types and numerical precision specifications. (72-1)
2. I prefer reserved words. I believe that some syntactical kludges in 8X are caused by the lack of reserved words. A conversion program could be written by ANSI and supplied by every compiler vendor with an 8X compiler to change variable names which conflict with identifiers. (42-2)
3. I never have spread an identifier out by inserting spaces and I never have jammed multiple identifiers together. I suggest eliminating the nonsense of compacting blanks from source text. Not many users could have a problem with this, but, once again, a program could be written to eliminate the misery. (4)
4. It seems quite apparent to anyone who uses C, Pascal, Ada, etc that a modern language needs pointers. Surely, one of these languages is close to a reasonable syntax for user-defined types with pointers. Let me suggest Modula-2. (42)
5. I have implemented a Fortran pre-processor and determined that a repeat loop is obviously designed as below:

```
REPEAT
  statements
  ...
UNTIL ( expression )

Also, the syntax for a while is obviously:

WHILE ( expression )
  statements
  ...
END WHILE
```

87 NOV 23 P1:15  
 X3 SECRET

Surely, these are better than a DO with no iteration control. Of course, I do favor the CYCLE and EXIT statements. In fact, I frequently use the following:

```
WHILE ( .TRUE. )
  statements
  if ( expression ) EXIT ! actually my word is BREAK,
END WHILE ! but who cares about trivial details
```

43

- 6. I think that BITS would be nice. However, I would prefer to have the EQUIVALENCE capability to use an integer array and a bit array in the same memory locations. I don't care if the EQUIVALENCEing of integers and bits is portable; I just want to easily manipulate bits for graphic applications. I can deal with all portability problems. I've been doing it for years without bits and I know I could do better with bits. 42-
- 7. I believe that any construct equivalent is power to EQUIVALENCE is going to be confusing if over-used. Therefore, I prefer to leave EQUIVALENCE in the language and let it work with any new data types. It may be non-portable, but let me worry about that. 42-
- 8. Similarly as to 7, COMMON is powerful and potentially dangerous. It should continue to be used with proper respect. 4?
- 9. I do like the array operators. Perhaps I'll get to use them. 42-
- 10. I do like the user-defined infix and prefix operators. They should make some programs clearer. 42-
- 11. I do like the idea of recursion and I see no harm in requiring the word RECURSIVE before a subprogram to make it recursive. 42-1

I am glad to get some input into the process, but I fear that my comments will be basically ignored. I read some of the responses to IBM, DEC and UNISYS and I expect that the committee thinks that it knows what is best for me. I resent that.

I propose that there be an election to select the new Fortran standard. Perhaps, the members of ACM could vote to select one of several competing standards. Surely the committee members who voted against 8X would prefer to let the Fortran community select their proposed standards. Perhaps IBM would present a candidate language and maybe DEC would. Then we could have an election in which my vote would count as much as yours. That would at least give us another chance for a decent extension to Fortran. 4-

Well, I've said enough. I hope Fortran will continue to be my language of choice, but 8X (if approved) would not be. I would feel compelled to steer a large group of users (about 100) to a different language.

Thanks for the opportunity to present my views. I wish I had time to present a cohesive alternative to 8X, but I must study.

Sincerely,

*Ray Seyfarth*

Ray Seyfarth  
 Rt. 4, Box 226  
 Alachua, FL 32615

43 #43

# Swanson Analysis Systems, Inc.

Johnson Road, P.O. Box 65, Houston, PA 15342-0065

TWX 510-690-8655

PHONE (412) 746-3304



September 28, 1987	Subgroup	Nominal
NOV 19 PM 2:50	43-1 EDIT	43-8 I
	43-2 GEN	43-9 C
	43-3 DATA	43-10 G
	43-4 DATA	43-11 P
	43-5 DATA	43-12 D
	43-6 GEN	43-13 D
	43-7 GEN	43-14 G

Mr. Presley Smith  
 Manager, Development Services  
 Convex Computer Corporation  
 701 Plano Road  
 Richardson, TX 75081

Dear Presley:

The following comments are based on a cursory review of the Fortran 8x manual (June 1987). This review, although not in-depth, required considerable time and will probably require a "second pass" and possibly reference to a supplemental text in order to better understand the "new" language. The presentation style and examples are readable and represent a welcome improvement over the 77 standard. The complexity of the language, however, has increased considerably.

In general, we have no objection to improving the language. Our own code also has a documented, publicly used command language that we have to periodically improve (with care and upward compatibility). One unfortunate consequence of the present 8x "improvement" is that the compile time will increase and compilers will be more complex and error prone. We strongly feel that a "language overhaul" was not necessary, but that improvements should have been made to the existing language structure.

**AREAS WHICH WE FEEL NEEDED IMPROVEMENT BUT WERE not INCLUDED ARE:**

- IMPLICIT "NONE" (to force explicit typing and better error checking) ← (4)
- Bit manipulation ← (43-4)
- Pointers (such as Call by address, etc.) ← (43-5)
- INCLUDE statement ← (43-6)

**AREAS WHICH WE FEEL NEEDED IMPROVEMENT THAT WERE INCLUDED ARE:**

- Vector operations (including Intrinsic) ← (43-7)
- Generic Intrinsic names ← (43-8)

**AREAS THAT WERE INCLUDED WITH USEFUL, BUT MINOR, IMPROVEMENTS ARE:**

- CASE construct (Computed GO TO was sufficient) ← (43-9)
- In-line comments ← (43-10)

**AREAS THAT WERE INCLUDED WITH LITTLE OR NO IMPROVEMENT AT A SIGNIFICANT INCREASE IN COMPLEXITY ARE:**

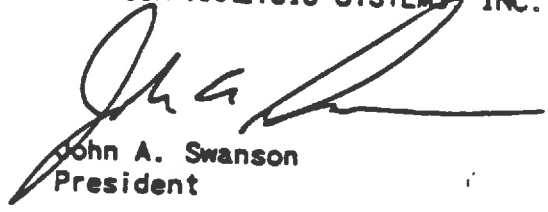
- USE statement (poor substitute for INCLUDE) ← (43-11)
- Derived Types ← (43-12)
- Variable precision ← (43-13)
- Alternate inputs for new commands (i.e., appending keywords to another command vs. direct input of the keyword as a command). ← (43-14)
- Document one or the other.

Mr. Presley Smith  
Convex Computer Corporation  
Page Two  
September 3, 1987

Having just received the 8x manual, we haven't had time to adequately review the language. Additional comments and criticisms may be forthcoming.

Sincerely,

SWANSON ANALYSIS SYSTEMS, INC.



John A. Swanson  
President

bp



INDEPENDENT BROADCASTING AUTHORITY

Crawley Court WINCHESTER Hants SO21 2QA Telex: 477211 Facsimile 822378  
Tel: (switchboard) Winchester S23434: (direct line) Winchester 822525

Engineering Division

43

ETF/sjc/sarah/7909

23rd October 1987

Subgroup Nominations:

- 44-1 CIO
- 44-2 CIO
- 44-3 CIO
- 44-4 DATA
- 44-5 GEN
- 44-6 GEN
- 44-7 CIO
- 44-8 CIO
- 44-9 GEN
- 44-10 CIO
- 44-11 PROC
- 44-12 GEN
- 44-13 GEN
- 44-14 GEN
- 44-15 DATA
- 44-16 CIO
- 44-17 CIO
- 44-18 CIO
- 44-19 CIO
- 44-20 CIO
- 44-21 GEN
- 44-22 PROC

Committee on Computers Information Processing,  
American National Standards Institute inc.,  
1430 Broadway,  
New York 10018,  
USA.

87 NOV 23 AM 11:31

RECEIVED

NOV 10 1987

INFORMATION SYSTEMS

Dear Sirs,

FUTURE ANSI STANDARD IMPROVEMENTS  
Computers & Information Processing : FORTRAN-77

We have for many years made intensive use of FORTRAN as our prime language for numerical analysis and data processing, not only because FORTRAN is recognised as a world standard for exchange of programs but also because it contains excellent sub-routine procedures and a comprehensive sub-set of mathematical functions. In particular, the COMPLEX functions are vital to any work on Radio Frequency and Electronic systems and are not found, to our knowledge, in any other programming language.

Anyone who has had to solve complicated expressions involving COMPLEX quantities in other languages such as BASIC or PASCAL will know how immensely laborious the programming becomes, and how inscrutable the programs remain after they are written, even to the authors themselves.

For this reason alone, we believe that FORTRAN has survived over the years owing to the demand by Radio Frequency and Electronic engineers for the COMPLEX functions. However, with the proliferation of modern powerful microcomputers we regard FORTRAN-77 as being in danger of passing into extinction, for reasons to be explained shortly. The means of guaranteeing its survival must be addressed now.

The implementation of BASIC on 16-bit and 32-bit microcomputers provides the user with many desirable features (apart from COMPLEX functions) not found in the implementation of FORTRAN-77 on these same machines. Unless FORTRAN can meet this new challenge it could become extinct. We cite in three Appendices some features of modern BASIC languages which are of great utility to microcomputer programmers and users. Appendix A is in our opinion the most important and refers to features that would permit the user to INTERACT with the machine without breaking out of a running program.

We offer in the three Appendices our suggestions as to what should be incorporated into a new FORTRAN-88 ANSI standard language, not only to make it useful to mainframe users but also to make it competitive in the emerging world of microcomputers. We speak as serious experienced users



of computers for scientific and technological purposes. Computers are becoming faster, and in our work they need to become very much faster. When that happens, the complexities of problems that are waiting to be solved will put a premium not only upon speed and capacity but also upon the floating-point precision that will then be needed.

We trust that we will not be condemned to a next decade in which computers are vastly faster and larger, but the programming languages remain inadequate to obtain the full potential from the new machines.

It is a curious feature of computer programming in the 20th Century that the 'high-tech' part of the coding can be written in a few hours, but then another several weeks are often spent in 'low-tech' matters of layout, formatting, translating and debugging, hardly the best use of intellectual resources. Any new facilities that can speed up the programmer's task will be welcomed.

It might be argued that microcomputer manufacturers will provide implementations of FORTRAN-77 that will include enhancements, such as COMPLEX\*16. It is our experience that this rarely happens. The manufacturers may not be fully aware of what the programmers want, but what is certain is that they will nearly always provide the full ANSI standard sub-set. We look to ANSI to ensure that the next ANSI standard is sufficiently comprehensive to do justice to the vast improvements expected in the next generation of micros and mainframes.

Yours faithfully,

*E. T. Ford*

E.T. FORD (Head of Service Area Planning Section)

*P. T. Hayter*

D.T. HAYTER (Principal Engineer - Radiowave Propagation Studies)

*R. H. Vivian*

R.H. VIVIAN (Principal Engineer - Advanced Computer Studies)

*C. K. Jacob*

C.K. JACOB (Senior Engineer - Antenna Design)

\*Att.



APPENDIX A

Future FORTRAN Facilities which would be vital on mainframes and microcomputers and, without which, FORTRAN could possibly become valueless in the future.

A. (i) INTERACTIVE OPERATION USING INKEY AS IMPLEMENTED IN BASIC (44-1)

The keyboard buffer should be constantly scanned to determine if any key is pressed during real time execution of a program. Thus the user has the option to interrupt, or to inject changes to variables without interruption, enabling simulations to be undertaken without breaking out of a running program.

We are unaware of any such facility in FORTRAN, either on mainframes or microcomputers. The user becomes effectively 'locked out' from any interaction with the machine from the moment the program commences to execute. In this modern scientific age the inability to interact with the computer is inexcusable.

A. (ii) INTERACTIVE OPERATION FROM ANALOG-TO-DIGITAL CONVERTERS (44-2)

We have used potentiometer boards to interact with microcomputers to carry out real-time tuning simulations on electrical networks. The potentiometers are interrogated by analogue-to-digital converters built into our microcomputers. The converters can only be addressed from within BASIC programs. No facility exists in our FORTRAN-77 implementations, as a consequence of which the language is becoming less and less frequently used.

Our applications for analogue and keyboard interaction are for serious scientific simulations, not for the playing of 'games' using joysticks. Nevertheless it is ironic that analogue interaction was probably devised originally to satisfy the demands of domestic games programs in BASIC.

A. (iii) WRITING TO VDU SCREEN WITH A LOCATE COMMAND (44-3)

A feature available in IBM-compatible BASIC and several other versions of BASIC has become indispensable in user-friendly programming, namely the ability to write directly to any part of the VDU screen whilst disabling the scrolling. In GW-BASIC, for instance, the LOCATE (X,Y) command allows a VDU screen to be filled with static data in the form of tabulated columns with headings, whilst one or more dynamic data values can be made to display the ever-changing position of, say, a digitising-tablet's cursor. These values 'spin' whilst the static data does not. In some of our work on terrain maps there is no other way of coping with this interactive work. A FORTRAN-77 program by contrast (even if it could read from a digitiser in the first place) would cause every line on the screen to continually scroll off the top, utterly destroying the utility of the method.

A. (iv) EXTENDED PRECISION OPTIONS (44-4)

There is an undesirable tendency of microcomputers to implement DOUBLE PRECISION to only 16 significant digits in FORTRAN-77. The more powerful and faster the new computers, the greater the numerical intensity of computations will become in the future. There will be a need for precision up to 24 significant digits. This could be provided in various ways, for example a

(43)

TRIPLE PRECISION declaration as well as DOUBLE PRECISION, or alternatively a user-defined precision (such as is in Xitan's XBASIC) which in FORTRAN could be handled by REAL\*12, REAL\*16, REAL\*20 and REAL\*24 within individual declarations or globally IMPLICIT declarations.

The provision of DOUBLE/TRIPLE/EXTENDED PRECISION to the COMPLEX type should surely become a standard ANSI feature. It is preposterous that Complex Numbers cannot be dealt with in double precision in programs where Double Precision is essential in all other respects. Indeed, this extended precision should be available as standard on all REAL and COMPLEX functions including trigonometric and hyperbolic functions.

A. (v) INTERPRET/COMPILE OPTION (44-5)

What a tremendous advantage there would be if microcomputer programmers could develop and debug their FORTRAN programs in an Interpretive line-by-line mode exactly analogous to BASIC. The sacrifice would be a loss of speed of perhaps 5 to 10 times, but after a successful development the program would then be compiled and run at full speed. Programmers are reluctant to prepare new FORTRAN-77 programs on a microcomputer because it can take 15 minutes to compile, link and re-start a modestly-sized source after each individual debugging operation. In the development of a powerful terrain-illumination program on a microcomputer, we encountered 1000 debugging operations during the 3 months preparation time. We were obliged to use BASIC because each coding correction could then be done in 1 minute rather than the 16 minutes that would have been required in FORTRAN-77. Unless ANSI, together with the microcomputer industry, tackle this problem FORTRAN could cease to be used on microcomputers in the future.

A. (vi) MATRIX ALGEBRA (44-6)

Vast numbers of numeric solutions require matrices and these are frequently implemented in BASIC on mainframe and microcomputers. Matrix arithmetic and inversion would greatly enhance the problem-solving features of an ANSI-standard FORTRAN, provided of course that the routines could support EXTENDED PRECISION and COMPLEX variables as standard within the matrices.

A. (vii) PARALLEL PORT PRINTING (44-7)

A command should be included to permit output to be directed to a parallel printer from within a FORTRAN source program, as distinct from an initial run command.

A. (iii) GRAPHICS (44-8)

High speed Screen Graphics are now an essential tool of the scientific systems-analyst. ANSI should incorporate a universal standardised set of monochrome and colour pixel graphics commands, including the addressing of screen-dump facilities, within the FORTRAN source code. Microcomputer manufacturers could then implement these on their 'ANSI compatible' machines. (We believe it is already possible to include FORTRAN sub-routine calls for driving peripheral graph plotters).

Future FORTRAN facilities which, although not vital, nevertheless would be exceedingly useful and are already implemented in other languages.

B. (i) Unique unambiguous distinction between variable names longer than six characters (44-9)

In BASIC sometimes 40 characters are unambiguous. In FORTRAN at least 16 would make programs much more legible as illustrated by the examples below:

BRANCH4, BRANCH5,  
RANGEKM, RANGEMILES, or RANGE\_MILES  
WAVELENGTHMETRES, WAVELENGTHINCHES, or WAVELENGTH\_INCHES.

B. (ii) LABELS (44-10)

The provision of alphanumeric LABELS to signify blocks of program coding would be a very useful feature (already available in some of Hewlett-Packard's programming languages) and is even absent in BASIC. A label could also terminate a DO-LOOP, for example;

```
DO 'SMITH' K = -6,100,2
  ⋮
LABEL 'SMITH'
```

Putting the alphanumeric name within literals would also allow the use of entirely 'numeric' labels:

```
e.g.: LABEL '1988'
      GOTO '1989'
```

Individual statements, and blocks of coding labelled with 'names' rather than numbers is mnemonically easier for programmers to keep track of events, for example:

```
GOTO ('XVELOCITY', 'YVELOCITY', 'RETARDATION') N.
```

B. (iii) COMMON ALL declaration (44-11)

We recognise and applaud the very powerful SUBROUTINE facilities in FORTRAN which are superior to GOSUB and PROCEDURE in BASIC and many other languages. Standard FORTRAN SUBROUTINE CALLS should remain as at present. One enhancement that could avoid considerable tedium in program code preparation would be the option of a COMMON ALL declaration, after which no COMMON lists would be needed, because all variables would then be commoned with the calling program, analogous to BASIC's GOSUB.

B. (iv) EXTENSIONS TO IMPLIED LOOPS (44-12)

FORTRAN already scores many points for this superb facility, not available in BASIC. It should surely be extended to setting up arrays, using a syntax thus:

```
DIMENSION A (0:4000)
(A(K) = 60*K/2000, K = 0,4000)
or better still: A = 60*K/2000
```

B. (v) MULTIPLE STATEMENTS ON ONE LINE

44-13

To save forests of paper, please let us have the option, within ANSI standard of multiple statements on one line, separated by semicolons. This is already standard on our Honeywell mainframe Fortran implementation. The exclusion of multiple statements on a line of FORTRAN-77 leads to nonsenses like the following example below, especially on 132-character printer paper:

```
X1 = 4
Y1 = 3
Z1 = 2
X2 = 0.9
Y2 = 0.7
Z2 = 0.6
```

why not let us write:

```
Z1 = 4; Y1 = 3; Z1 = 2; X2 = 0.9; Y2 = 0.7; Z2 = 0.6
```

English is read from left to right. Why are programming languages limited to a vertical list down the extreme left side on a long roll of wide paper? I grant that multiple statements on a line should not be overdone, but judicious use can result in neat coding layouts, e.g.:

```
ZREAL = AMPL * COS(PHASE) ; ZIMAG = AMPL * SIN(PHASE)
DO 2000 K=0,100; A(K) = LOG(K/2) ; 2000 B(K) = EXP(K/10)
```

Not the least advantages during program development are that more coding can be scrutinised on the very-limited 25-line format of most VDUs, and paper listings of programs would become so much easier to handle.

B. (vi) 80 CHARACTERS PER LINE

44-14

Let us have 80 characters per line as the future standard. The 72 character line length is historical and reflects the use of teletype machines. Modern VDUs are invariably 80-CHARACTER capable. Lower-case remains essential, of course, for comments and format printing.

B. (vii) PRESET/NON-PRESET VARIABLES OPTION

44-15

A useful tool for debugging purposes would be an option within the FORTRAN source code to do either of the following:

- preset all variables and arrays to zero at run time, for convenience;
- leave all variables unset and invoke an error message such as 'unknown or misspelt variable' to highlight any erroneous or unset variables and arrays when they are encountered during execution.

B. (viii) RS 232 PORTS

44-16

Any revision of FORTRAN-77 should reflect the increasing use of microcomputers. Although we believe that ANSI FORTRAN-77 allows direct access to RS 232 serial ports, it does not allow (to our knowledge) the serial ports to be opened and closed directly from within FORTRAN source code without recourse to machine-code routines.

The new command should allow a serial port to be opened and the baud speed, number of bits, number of stop bits and parity to be specified. There should also be a command to permit constant polling of the attached serial device. Otherwise a 'device time out error' would occur if the 'data set ready' was not detected. The new command should also allow for additional extensions to control the RTS, CTS, DSR, DC line signals and optional line feed and parity checking should also be permitted.

43

Future FORTRAN facilities which, although not essential, nevertheless would be useful, particularly in translating existing programs from one language to another.

44-17

- C. (i) Unlabelled DO LOOPS so that the programmer does not have to remember later the terminating statement number, nor alter it, when preparing code. BASIC uses a very sensible FOR-NEXT structure which does not require to be labelled and this works without ambiguity. An alternative is a 'named' statement rather than a 'numbered' statement to improve user-friendly programming e.g.

```
DO 'SPEED' N = 1,1000
  |
  LABEL 'SPEED'
```

44-18

- C. (ii) Some means of opting to execute a DO LOOP either once minimum or not at all. This would ease the translation of programs from one language to another. It is thought that a WHILE-WEND structure could deal with this potentially serious problem of translation between languages.

44-19

- C. (iii) A WAIT command to suspend program execution until the status of a port had changed.

44-20

- C. (iv) User-defined screen characters to permit the generation of λ, π and the ° degree sign, for instance, on the VDU.

44-21

- C. (v) To make a great improvement in source code legibility it would be useful to have the optional recognition of square brackets allow arrays to be defined with square brackets whilst parentheses would remain essential for formula translation. The square bracket enhancement would have to be compatible with old programs using round brackets, of course.

Example:

```
A[N] = (AMPL[M]*SIN(PHASE[M]) + AMPL[N]*SIN(PHASE[N]))**2
```

44-22

- C. (vi) Specifically as an aid to translating BASIC into FORTRAN a GOSUB call acting on main program code in precisely the same way BASIC would be useful. There would be no argument lists, no local variables, no common declarations or the like. Two examples below

```
GOSUB 3000
```

```
GOSUB 'ROUTEFIND'
```

Followed later by .....

Followed later by .....

```
3000 X = 3
  |
  RETURN
```

```
LABEL 'ROUTEFIND'
X = 4
  |
  RETURN
```

Such a translation feature might have considerable sales appeal for buyers of future FORTRAN compilers.

Subgroup Nominations.

45-1 GEN	45-7 GEN
45-2 DATA	45-8 GEN
45-3 DATA	45-9 PROC
45-4 DATA	45-10 GEN
45-5 GEN	45-11 GEN
45-6 CIO	

#45

43

X3 Secretariat  
311 First Street NW, Suite 500  
Washington, DC 20001-2178

Dec 7, 1987

Sirs:

1987 marks the fifteenth year I have been a Fortran user. I started with WATFIV on an IBM/360 at Princeton; I've since used IBM, DEC, CRAY, CONVEX, SUN, AMDAHL, FPS and even NCUBE Fortran. By and large, I've considered Fortran a necessary evil - necessary because most of my colleagues were literate only in Fortran and because the only decent optimizing compilers were written for Fortran - evil because the profusion of explicit index calculations and the limited set of data types and operations made programming and debugging cumbersome and error prone. Because I had a previous background using APL, I felt these limitations more strongly than most. In order of personal preference, these are the modifications I would support in a revised Fortran standard:

(1) Arrays without subscripts. In the vast majority of cases, programmers know very simply what they want to do with an array. DO-loops and subscripts are a cumbersome, error-prone way to express array manipulations.

45-1

(2) Data structures. Record handling currently is done with equivalenced variables and common blocks - a difficult, nonportable way to handle information. The addition of fixed-length character strings with Fortran-77 (the biggest botch of the current Fortran standard in my opinion) has exacerbated these problems.

45-2

(3) Variable-length character strings. Text is a most unpredictable object - its size and shape is usually irregular and rarely predictable in advance. Manipulating it in fixed-length, predefined buffers is tedious and exceptionally error prone. A null string of zero length is especially desired.

45-3

(4) Automatic variables. Current schemes for dynamic memory allocation are based on indices into a fixed base array and the passing of the beginning element of the area, plus all other relevant variables, to a subroutine. The ability to simply set or change the location of an array would be easier. This is not the same as

45-4

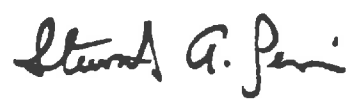
requiring explicit pointers in the language – implicit pointers would be sufficient, perhaps even better.

- (5) Standard include syntax. Without shared definitions of parameters, common blocks, and data types, it is impossible to build a reliable, maintainable body of code. Each computer currently has its own distinct preprocessors and syntax for including shared definitions into a program. This usage should be standardized. 45-5
- (6) DO-WHILE, ENDDO, and BREAK. Once we have array operations, the standard fixed-increment DO-loop becomes much less useful. A DO-WHILE or FOR loop construct would cleanly handle most non-array iterations, especially if accompanied by unlabelled END-DO's and a BREAK construct to exit one or more levels of iteration. 45-6
- (7) Symbolic labels. Numeric statement labels are peculiar to Fortran. Even assembly languages have abandoned them in favor of symbolic labels. The ASSIGNED GOTO construct does provide a method of naming statement labels, but has been subject to so much abuse that it is deprecated in virtually every Fortran style guideline. 45-7
- (8) Longer names. Six characters are just not enough for intelligible, readable variable and function names. Twenty or thirty is more like it. 45-8
- (9) Argument validation. One of the big time-wasters in debugging Fortran is detecting when a routine is called with the wrong number or type of argument. While products are available to catch this and similar problems, realistically their cost is much more palatable when bundled with the Fortran compiler. 45-9
- (10) Macro facility. Conditional code compilation and generation of inline code are currently done in a strongly system-dependent fashion, if at all. A standard macro facility for Fortran would be a way to aid portability and handle system dependencies in addition to the traditional macro function of inline code generation. 45-10
- (11) Inline comments. Again assembly language has allowed these for years, as have most Fortran compilers. It needs to be standardized. 45-11



In summary, while I believe most of these features are available in the proposed Fortran standard, I know that not all are. I ask you to consider including these suggestions in the final Fortran 8-X standard.

Thank you,



Stewart A. Levin  
Mobil Research and Dev.  
13777 Midway Rd.  
Dallas, TX 75234

Subgroup Nominations:

- 46-1 PROC
- 46-2 GEN
- 46-3 GEN
- 46-4 GEN
- 46-5 GEN

#46

43

9812 Matchpoint Place  
 Dallas, TX 75243  
 December 8, 1987

X3J3 Chair  
 X3 Secretariat  
 311 First Street NW  
 Suite 500  
 Washington, DC 20001-2178

Dear Sir or Madam:

From what I have heard about the proposed 8x Fortran Standard, I am very much opposed to it in its current form for the following reasons:

- I understand that the MODULE/USE features will increase my compile times on my PC. The compile times are already unacceptably slow and I have written two different letters to MicroSoft complaining about that already. I understand the benefits of the MODULE/USE feature, but don't believe it is worth the cost. (46-1)
- I have also been told that the language is nearly double the size of the current Fortran language. This will probably force me to purchase additional memory for my PC to run an 8x Fortran compiler on my machine. I guess that big companies might not mind purchasing more memory for all their machines, but I would certainly prefer to use my money for other things other than memory to run a new Fortran compiler on my PC. (46-2)
- I am disappointed that the INCLUDE statement was missing. I use two different machines at work and there is no way to have common source between them because the syntax of the INCLUDE statement is different on each machine. Why didn't the committee standardise the INCLUDE statement? (46-3)
- The thought of statements possibly being removed from some future standard will send my manager at TICOS through the roof. I expect he will also write you a letter on that subject. (46-4)

When I heard about the proposed new Fortran Standard, I was excited until I went to a presentation on what it contained. Now it just scares me that it will be too large to work on my old 8088 IBM clone and I will be faced with three options: (1) don't use Fortran 8x, (2) purchase a new computer, or (3) quit using Fortran all together. None of those options appeal to me. (46-5)

I know that my voice carries little weight, but my position is to leave Fortran alone.

Thanks,

*Edward Smith*

Edward Smith  
 Professional Engineer

CW

43

#47

RECEIVED

DEC 9 1987

ANSI-DCR CENTER

To Catherine A. Kachurak ✓  
CREMA

December 6, 1987

American National Standards Institute  
1430 Broadway  
New, NY 10018

Subgroup Nomination  
47-1 GEN

Dear Folks:

RE: X3.9

I vote "no" on Fortran 8x for all the reasons given by Richard Weaver of IBM in his Dec. 1986 ballot, and in addition because my inner voice was saying that we already have a Pascal - why make another one. I also strongly agree with Mr. Weaver's list of suggested enhancements to Fortran 77, which would still have the language free to be Fortran and yet give it added portability and strength.

47-1

Sincerely,

Mary Hovik

Mary Hovik  
Assoc. Prof.  
Lehigh County Comm. College

87 DEC 11 PM 2:53

X3.9

107-CDB-5  
February 5, 1988

**From :** Carl Burch

**To :** X3J3

**Subj :** Subgroup Nominations for Public Review Letters 48 and 50-64

Please find attached the annotated copies of Public Review letters 48 and 50-64, marked with my recommendations for subgroup assignments. All subgroup assignments are negotiable between the Subgroup Chair and the Public Review Working Group. Any omissions noted should be brought to the attention of the Public Review Working Group.



TO: Carl Burch and Ivor Phillips

FROM: Jeanne Adams *Issue*

DATE: January 27, 1988

SUBJECT: Public Review Comments #48, 50-64

Enclosed you will find public review comments #48, 50-64. Notice that #49 is missing. They sent one for Pascal instead of Fortran. It appears that it will remain missing as well, since they want to keep their numbers straight.

I think these again are all right for the May meeting. But after a few more numbers we may invite them to the August meeting. See you next week.

cc. Jerry Wagener



#48

CW

THE UNIVERSITY OF GEORGIA

OCIS User Services  
Barrow Hall  
Athens, Georgia 30602 (404) 542-5359

Gwendy Phillips  
CBEMA  
NY NY

44

Public Comment for ~~Dpans~~ Fortran-Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY, 10018

11 JAN 1988

A friendly note on the Fortran 8X draft:

I think you are trying to change the language too much. Fortran is not a "do all" language. It is a "formula translation" language for scientist and engineers. Significantly increasing the complexity of the language presents problems to the compiler author and to the average user. We are just starting to get good F77 compilers on PCs and they are cramped. Fortran 77 has been available on many mainframes installations for only a few years.

The average user of Fortran is an expert in some field unrelated to computers. The language must be kept simple (within reason) so that these users do not have to go to "computer experts" to get their work done. One might say that users don't have to use the new features, but if they are listed in the manual most (especially the new users) will try to use them. As the level of complexity of the language increases, the more users will find it difficult to use. As a good rule of thumb, if the user's manual is more than an inch or so thick, the average user will have problems working with it.

48-1

The features I would like to see in the 8x standard include -

- Data Structures ← 48-2
- Some sort of array options to be ready as that hardware becomes more available (but so code will run on "older" machines) ← 48-3
- Bit String Data Type (a bit advances for most users, but worth the effort in a lot of applications) ← 48-4
- IMPLICIT NONE ← 48-5
- New form of DO ← 48-6
- ALLOCATE/FREE (could save memory buffers when used correctly) ← 48-7

Subgroup Nominations:

48-1 GEN	48-5 DATA
48-2 DATA	48-6 CIO
48-3 GEN	48-7 DATA
48-4 DATA	

JAN 13 AM 11:49 88.

LES EX

Sincerely,

Michael G. Miller  
Michael G. Miller  
Manager, Technical Support

P. 274

#50

# Los Alamos

Los Alamos National Laboratory  
Los Alamos, New Mexico 87545

DATE January 5, 1988  
IN REPLY REFER TO: X-7-88-U1  
MAIL STOP MS B257  
TELEPHONE 505 667-7550  
FTS 843-7145

44

## Subgroup Nominations:

50-1 DATA	50-11 GEN
50-2 GEN	50-12 DATA
50-3 GEN	50-13 DATA
50-4 DATA	50-14 DATA
50-5 DATA	50-15 GEN
50-6 DATA	50-16 GEN
50-7 DATA	50-17 DATA
50-8 DATA	50-18 GEN
50-9 DATA	50-19 GEN
50-10 DATA	50-20 GEN

REC-11  
JAN 11 11:11 AM '88  
X3J3

Public Comment for Dpans  
Fortran Revision  
X3 Secretariat Attn: Gwendy Phillips  
CBEMA, Suite 300  
311 First Street, N.W.  
Washington, DC 20001.2178

Dear Ms. Phillips,

The following comments are topics that concern me about the proposed new Fortran standard. Some of the comments are of a general nature (because there are too many individual features to address individually), others, concern specific features. Page and line numbers that I cite in my comments refer to the X3J3/S8.104 June 1987 report.

### 1. The meta-standard

I would like to preface my comments by arguing the point that there is more to Fortran than the written standard. From the stand point of the day-to-day use of the language. Fortran, like other working programming languages, has an unwritten standard as well. That unwritten standard takes the form of a mental model of the general nature of the language. It is the unwritten meta-standard.

The meta-standard exists because the human mind is not capable of applying the thousands of minute details that constitute the language definition when designing a problem solving methodology. (There is a saying about the task of draining a swamp and the problems alligators cause that comes to mind as I write this.) As analysts, we must rely upon a kernel of generalities, i.e., the meta-standard, that we combine in different ways when designing a problem solution. The meta-standard is really the basis functions that span the problem solving space. Without them, we would not be able to solve complex problems.

Take completeness as an example. In Fortran IV, an end of file could be written but there was no standard way of testing for the end of file. Completeness was not achieved in this area until a standard way of testing for an end of file was put into the language. To me, completeness means that all aspects of an area are accounted for--its basis functions completely span the sub-space.

What happens when the basis functions do not completely span the sub-space? You have a sub-space with holes in it. The more holes you have, the more difficult it is to design a problem solution. (We're back to alligators and swamp draining.) When you look at the written standard, sub-space holes come about because of exceptions and limitations that apply to the use of certain syntax forms. (Ref: Weinberg, G. M.; The Psychology of Computer Programming; Van Nostrand Reinhold).

44

Take the rules for using character variables as an example of the effect of sub-space holes on the use of Fortran. One principal that we as users of the language have hung our hat on is that all variable types may be put in common. Another is that variable types may be put in common in any order with all other types. "Ah", you say, "character variables cannot be put in common with other variable types". Right. Because of that sub-space hole, character variables are a source of so many problems that they are unusable in many applications. Consequently, we are forced to use a degraded feature, viz., Hollerith strings, to do what we need to do in situations where it would be logical to expect to use a character variable.

50-1

What is the seriousness of corrupting the unwritten meta-standards of the language? The seriousness is sever. Whole problem solving methodologies are rooted in the unwritten meta-standards. In some cases, when those standards are corrupted, whole computer programs must be thrown out because the method of solving their problems must be completely redesigned.

Consequently, a programming language cannot be changed by looking simply at the written standard. Fortran is not just a programming language. It is a language analysts use to design solutions to complex problems. Because of that, it is the unwritten meta-standards that are really important in the design of a computer program. It seems to me that many of the features that are proposed are so burdened with limits of where they can be used that continuing this trend will only make Fortran die of congenital obesity. One guiding principle that should be followed. I think, when changing a language is that you end up with a lean, mean set of meta-standards.

50-2

## 2. Protectionism

Another area where I have some strong feelings is this controversy of functionalism versus protectionism. I live with the premise that Fortran is not a static, specialty language that is designed to teach "good" programming practices. On the contrary, it's a working language that must be flexible so it can be applied in a dynamic, demanding world. Consequently, I am an antiprotectionist professionally and morally.

Professionally, I object to measures taken in the language to protect me from "shooting myself in the foot." Experience over the past twenty-some years of writing Fortran code has proven to me that there is no such thing as the *a priori* right way of doing things—the right way in one situation may be a terrible way in another situation. It's part of my job, as I see it and as my employer sees it, to make intelligent decisions about how the language should be used to solve each particular problem. For a committee at another place in another time to decide what is in my professional best interest is deprive me, in the here and the now, of my ability to best solve the problem at hand.

50-3

Morally, I object because it's un-Fortran-like as well as being un-American. It is in the spirit of Fortran-ism (as it is understood in the unwritten meta-standards of the language) as well as Americanism that as much unencumbered freedom should be allowed as possible. The US constitution has survived for 200 years because it was designed to stretch with the times and so should Fortran be designed. For a



44

programming language to be viable in a dynamic world, it must not be constrained by the moral dogmas of another time.

### 3. Automatic arrays

I've seen the need for dynamic allocation and deallocation of arrays for over twenty years. For the past six or seven years, I have worked with memory managers so we could have dynamic arrays in our codes. So, coming with that interest and background, I was quite interested in the automatic-array capability that is proposed. My bottom-line judgment, unfortunately, is that it is so bad that we will not be able to use it. Let me go into the details of what I think is wrong with it.

First, there is a rule that you cannot allocate an allocated array. But, there is no test to find out if an array is allocated. That's like being able to write an end-of-file mark on a file without having a way of testing for it.

50-4

Second, there is no provided-for way of changing the size of an array. To accomplish that task in a way that conforms with the rules of the names of the arrays that can be allocated, you have to allocate a temporary array, copy the old array into the temporary array, deallocate the old array, reallocate the old array with its new length, copy the data from the temporary array in to the newly allocated old array, and, finally, deallocate the temporary array. There is an unnecessary data copy involved with this that would not have occurred had there been a provided-for way of changing the size of an array.

50-5

Third, there is no explicit garbage collection capability. I had a problem the other day where an array kept getting bigger and bigger. The problem produced blocks of unused memory scattered through memory that were individually too small for any array to use. The job finally blew off the machine because it couldn't get enough memory. The fact is, collectively the holes added up to a million and a half words of memory—more than ample for what was needed.

50-6

Fourth, the code designer is at the mercy of how the deallocating is implemented. There is nothing in the proposed standard that says anything about garbage collection. If automatic arrays are implemented so memory is allowed to grow to a high water mark and unused blocks are reassigned when possible, you can get into the problem referred to in the previous paragraph. Although there were serious problems in that instance, there are situations where the high-water-mark method of memory management is exactly what you want to do. (I cite a finite-difference, time-marching solution method as an example.) A second implementation method is to garbage collect every time an array is deallocated. However, in all but the most trivial applications, this method is disastrous because of the amount of data that often has to be moved around in memory. In general, there is no such thing as an *a priori* automatic garbage-collection rule that can be used. When to garbage collect is very much application dependent.

50-7

Fifth, the code designer is at the mercy of how the allocating is implemented. In some of our codes, hundreds of arrays are dynamically allocated every time cycle. Those problems then run thousands of time cycles. Typically, if you do the multiplication, solving a problem means that tens to hundreds of thousands of

50-8

arrays are dynamically allocated and deallocated. If we use the wrong allocation scheme, we are finding that run times are doubling and tripling compared to the run times using static arrays. It should be mentioned, perhaps, that these problems require several hours of computer time and make up the lion's share of the runs we have. If we double or triple their run times, we would have to buy another four-processor Cray to get our work done. Congress would have a fit. Research done at universities on allocation schemes establishes that, as with deallocation, there is no such thing as an *a priori* right way to allocate memory. How to allocate memory is, also, very much application dependent.

44

50-8  
co-7

Sixth, the capability is maintenance intensive. The number of lines of code that are required to allocate and deallocate hundreds of arrays can get extensive. It's not only the individual allocate and deallocate lines that are involved but also the lines of logic that are required to decide which arrays to allocate and deallocate that must also be counted. If we want to add a new array, it could take a week to figure out how to do it. It's crazy. We are trying and want to move away from this maintenance intensive style of code design. The proposed standard would actually making it harder for us to design and maintain large codes

50-9

Seventh, automatic arrays cannot be treated like other arrays. Certainly, automatic arrays cannot be passed in common. That's a violation of the meta-standard that any variable can be passed in common. As I understand it (p. 12-5, line 41), a subroutine array argument must be either an automatic array or a static array. I can't call a subroutine once with an automatic array and then again with a static array. If that's true, I give the proposed automatic arrays the big thumbs down accompanied with a hardy Bronx cheer. The amount of work to accommodate that type of restriction is incredible. Whether my understanding is correct or not on passing static and automatic arrays, a point I would like to make is that strong typing is unbelievably maintenance intensive. It's also a violation of the Fortran meta-standard that there is no strong typing in Fortran.

50-10

Lest there be some confusion about how I feel about the proposed automatic array features, let me summarize by saying that it would be a serious mistake to put them into Fortran. In due time, the consensus will come to feel that they are a bad implementation. But, by then the language will have been contaminated and then it will be a mess to clean up.

#### 4. Whole array operations

What about whole array operations? Twenty years ago, I would have gotten my kicks by trying to use them. Today, I will not even consider them. The places where they could be used is so far and few between that to include them would just confuse the readability of the code. We would have to put comments in the code at the places where the whole-array syntax is used to say what it was.

50-11

#### 5. Bit variables

If bit variables are implemented analogously to character variables, forget them. Character variables can't be used in the application space I work in. If bit variables

50-12

44

are implemented the same way, I won't be able to use them either. For the sake of code readability, if I have to use a non-standard capability in one place in a code, I use it everywhere.

50-12, cont

## 6. Derived data types

I think I like derived data types. I haven't wrapped my understanding of them completely around, but so far they seem to make a positive contribution to code development. For one thing, they partially fill the hole created by character variables because they provide a way of mixing several data types into a single entity. That's really why I like them. I use a lot of descriptive vectors for all kinds of different things. Changing those to derived data types, it seems, would soften the maintenance problems that I currently endure. I think their use would reduce the number of lines of code that I currently have to maintain as well as eliminate some multiple maintenance points.

50-13

On the negative side, they violate the meta-standard that variables of all data types can be put in common. I also don't feel comfortable with the syntax used to assign values to derived-type variables (p. 4-9, line 8). The unwritten meta-standard clearly states that there should be an equal sign used in all assignment statements. (An equal sign is important to those of us who use text editors because we use equal signs to find stuff.) I also feel deep down in my bones that there are going to be problems associated with using derived types on dynamically allocated memory.

50-14

## 7. Operators

The following are a couple of opinions that I hold with respect to operators. Even though they are two opinions they both address the same issue.

First, look at operators from the point of view of the use of synonyms. I think that Fortran should support just the minimal set of relational operators-.LT., .LE., .EQ., .GE., .GT., .NE.. There is no end to the number of synonyms that one may like to use. To meet the requirements for a synonym capability, there should be either, the preferred case, a macro-like preprocessor that can convert the synonyms to standard Fortran or, less preferably, a operator derived type capability in the language. One of the rules of technical writing is to use the same word everywhere to refer to a concept. You don't use two different names to refer to one thing-it just confuses the reader. The same is true of programming languages. Don't mess up the readability of code with redundant synonyms. Synonyms also slow down the parsing that the compiler has to do.

50-15

Second, look at operators from the point of view of having generic operators. Traditionally, three notational forms are used with operators: prefix, infix, and postfix. With prefix notation the operator is placed before the operands. Prefix notation is used principally with unary operators, e.g., -5, SIN (THETA), etc., and with multi-operand operators, e.g., MAX (A, B, C, D). Infix notation is used principally with binary operators, e.g., 2 + 2, A .LT. B, etc. (Infix notation doesn't generalize cleanly to other than binary operations.) Postfix notation is use principle with Hewlett-Packard calculators.

50-16

If the trend is towards generic prefix operators, it seems logically consistent that there should be generic infix operators. Why is it logically consistent? Simply because you can map expressions in infix notation onto their counterpart in prefix space.  $2 + 2$  maps onto  $+(2, 2)$ . If the operator should be generic in prefix space, it should also be generic in infix space—there's no difference. If the reasons are sound for adopting generic prefix operators, they should hold for infix operators. If not, then let's not have any generic operators. Fortran should not discriminate on the basis of fix. (P. F-4 ff is littered with fix discrimination. Let's do away with that and make Fortran an equal opportunity language where all operators can become generic.)

44

50-16,  
cont.

### 8. Pointers

Where's the pointers? I can't believe how such an innocent looking feature can have so much power. I've worked with them as an extension to Fortran for ten years. They have become such an important part to the way I think of using Fortran that if pointers were taken away from me, I would take to programming in some other language. Why? I can't tell you. It seems that no amount of discussion works to convince people of how useful they are. You really have to work with them in the trenches to appreciate their significance. I will say that pointers in C, where they are very well implemented, has made C an extremely powerful language. Without pointers, C wouldn't be much of a programming language.

50-17

### 9. Order of declaratives

If there remain any rules on the order declarative statements must appear, they should be lifted. Why in this long established age of multi-pass compilers should the dimensioning declarative appear before the data statement that presets the variable? Why should all the type declarations have to appear before the statement-function definitions? Rules of that nature just serve to confuse the readability of code and engender poor programming practices. If all the rules of that type have been lifted, Fortran will be much better off with them gone.

50-18

### 10. Limits on the number of continuation lines

The guy across the hall was telling me about using an automatic code generation program. He worked for three days and finally arrived at some long, complicated expression only to find that all the parentheses that terminated the expression were on the twentieth continuation line. He said, "when you write your letter about Fortran SX tell them about that."

I've personally had to break long data statements up into several statements so many times its like "deja vu all over again" (as a famous catcher for the New York Yankees once put it). That stupid 19-continuation-line limit has about used up all its toleration points. It's a source of maintenance problems particularly with data statements. Besides, you wouldn't think in this day and age with computers and all that you would have a limit like that.

50-19

So, for the guy across the hall and my damn problems maintaining data statements, we've got computers now, let's go for unlimited continuation lines.

January 5, 1988

44

As a long time user of Fortran, I have a very unsettled feeling about the proposed changes. They just don't fit very well. If it's a question of all or none at all, Fortran, I feel, would be better off with none of the proposed changes.

50-20

Sincerely yours,

*Jack Peterson*

Jack B. Peterson  
Staff Member  
Computational Physics Group

JBP:pml

cy: J. B. Peterson, X-7, MS B257  
Peterson file  
CRMO, MS A150. (2). w/o enc  
Board of Standards Review  
A. Marusak, C-3, MS B265



Department of Computer Science

Harney Science Center (415) 666-6530

Subgroup Nominations:

51-1 DATA  
51-2 GEN  
51-3 DATA

19 December 1987

DEC 28 AM 1:55

ANSI X3 Secretariat  
CBEMA, Suite 500  
311 First Street NW  
Washington DC 20001

ATTN: Public Review of dpr ANS X3.9-198X Fortran

I have reviewed the draft Proposal document, and I have the following comments:

1. The Type Extensions in Appendix F, Section F.1, relating to BIT data type, should be reinstated as part of the full language.

A BIT facility in some form is commonly available as an extension to Fortran 77 in almost all existing implementations, because such a facility is needed in many Fortran applications. Examples of use include graph theory, bit-mapped graphics, and data transmission.

The same functionality is available in most other languages in one way or another, for example in the form of "sets" in Pascal. Bit handling is not a novel or untested area of programming language design.

A standard bit handling facility in Fortran is especially needed, since otherwise the current anarchy will continue, with consequent lack of portability. Apparently X3J3 has spent considerable time working out the best possible way to conform such a feature to the other parts of the proposed language; so it may be assumed that the description in Appendix F gives the most reasonable way to introduce such a feature into Fortran in a standard manner.

2. The feature "Significant Blanks in Free Source Form", described in Appendix F, Section F.5, should be included as part of the full language.

This feature has some impact in simplifying lexical analysis, by making it possible to more easily recognize the boundaries between tokens.

Significant blanks also contribute to the desirable goal of eventually providing tools for the analysis of program source text that will work with more than one language. For example, one would like to have a word processor tool (text editor) that could globally change occurrences of "X" in a text file to "Y", but only when "X" appears as a data object identifier (and not in a comment, for example); and such that the same tool could work for several languages including Fortran, Pascal, Ada [TM], etc. Although other features of the current languages (as well as of the proposed Fortran 8X) preclude the possibility of such tools at present, now is the only foreseeable opportunity to make this change that will be necessary (even if not sufficient) for reaching the eventual goal.

51-1

51-2

44

3. A pointer facility should be added to Fortran along with the other features in the draft proposal.

I have studied recent working papers prepared by members of X3J3, and it appears to me that a pointer facility could be added as a relatively minor extension to the existing ALLOCATE and ALIAS features. This facility would work in much the same way as the pointer facility in Pascal, which is useful for creating linked data structures, binary trees, etc.

51-3

Pointers are provided in all modern programming languages, with a syntax not too different from that recently suggested. It would be unfortunate to add user-defined derived (record) types to Fortran without any way to link them together into data structures such as linked lists.

SUMMARY:

I am greatly impressed by the obviously tremendous amount of effort that has gone into the development of this draft proposal. I believe that the new proposed language will be accepted by the Fortran user community. I also believe that, given current trends in hardware development, the new language will not be "too large" for implementation on even the smallest scientific computers that will be available in one or two years. By careful measures, the proposed language (even with the additions that I have endorsed above) is smaller than Ada [TM], and it has the advantage that at every step the developers of Fortran 8X have kept clearly in mind the possibility of efficient implementation "in the spirit of Fortran".

I hope that my comments will prove to be of value in shaping the further processing of the proposed revised Standard language, and in the general future development of the language.

Sincerely,

*Loren P. Meissner*  
Loren P. Meissner, Ph.D.  
Professor of Computer Science

#52

J. Wayne Gray  
Honeywell O.E.D.  
830 E. Arapaho  
Richardson, Texas 75081

X3J3 Chair -  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Re: Proposed Fortran 8X Standard

Sirs:

Review Letter # 52			
Subgroup Nominations :			
1	GEN	11	—
2	GEN	13	—
3	GEN	13	—
4	GEN	14	—
5	GEN	15	—
6	GEN	16	—
7	GEN	17	—
8	PROC	18	—
9	GEN	19	—
10	GEN	20	—

44

I support the position of IBM, DEC, DECUS, etc. that the proposed standard is NOT acceptable.

Specifically, I disagree with the proposal in these areas:

1. The proposed standard sufficiently changes the language to serve a different market. That new market may need such a language, but the proposed standard would abandon the existing market. 52-1
2. In the market currently served by Fortran, the cost of converting existing programs would be far in excess the practical, and the cost would far out-weigh the benefits of the proposed standard. 52-2
3. The features chosen for (possible) obsolescence are incorporated in the large existing software base, to such an extent that I believe that elimination of those features should not be done, at any future date, in a language called "Fortran". 52-3
4. The proposed new language would be much slower, and Fortran implementations and applications have usually been speed sensitive. 52-4
5. The proposed new language probably could not be executed or compiled on some of smaller machines, and the portability and popularity of Fortran would be decreased. 52-5

In response to the committee's reply to some of the NO votes:

1. I disagree with the committee's stated opinion that "it is important to introduce substantial new capabilities in each revision." 52-6
2. I disagree with the implication of the committee's statement "Evolution of hardware has created a need for language support." that hardware should constrain language structure. 52-7
3. I disagree with the committee's stated opinion that "whether or not Fortran 8X can be implemented in a one-pass compiler is not ... a serious issue....". This was in response to the module and the USE statement. I believe separate compilation and linkage in the manner of the current implementations of Fortran '77 to be more appropriate to the market Fortran serves than method used by Ada, which seems analogous to the proposed Fortran 8X. 52-8



44

I believe that if such a language as described in the proposed standard is needed, it should be separate from Fortran. The market would then be able to drive the selection between these two different languages. That is the more appropriate method of determining which is better, or more needed.

52-9

If such a language were needed, that need would have been reflected in the user community over the years since the last revision of Fortran, and it has not. If such a language were needed, the need would have been prompted software suppliers to service that market, and that has not happened.

In my opinion, the committee exceeded its charter (as I understand it) in this proposal, and I prefer the continuation of the Fortran 77 standard to the current proposal.

52-10

Wayne Gray

X3 5'

'87 DEC 24 AM 11:29

#53

44

DECUS Meeting  
Dallas, Texas  
December 15, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

NOV 19 10 11 AM '87  
X3

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. 53-1
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. 53-2
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. 53-3
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. 53-4

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

Seeger W. Hays

Company: Southwestern Bell Telephone

Address: ONE BELL BLVD RM 3120

DALLAS, TX 75202

Review Letter # 53			
Subgroup Nominations :			
1	DATA	11	
2	PROC	13	
3	DATA	13	
4	GEN	14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	

LAWRENCE LIVERMORE LABORATORY

National MFE Computer Center

#54 /

44

December 23, 1987



X3 Secretariat  
311 First Street  
NW, Suite 500  
Washington, DC 20001-2178

*CW*

Subgroup  
Nomination:  
GEN

Dear Colleagues:

I write to make some general comments about the proposed Fortran 8X Standard. It should be understood that these remarks reflect my personal opinion and are not to be taken as representing the Lawrence Livermore National Laboratory.

It is my opinion that the new proposed standard is an effort to create a new programming language while retaining the old name - Fortran. I would accept new features if old ones can be maintained. I do not approve of the new mechanism for deleting features in future editions of Fortran.

54.1

Whatever is done to Fortran it should be backward (or upward) compatible with previous versions. In its present form I find the 8X standard unacceptable.

Sincerely yours,

David V. Anderson

DVA:aw

JAN -4 P 1 72

University of California  
P O Box 5509  
Livermore California 94550

Telephone (415) 422-4017  
FTS 532-4017  
Twx 910-386-8339 DDE LLL LVMR

P. 287

#55



December 31, 1987

X3J3 Chairman  
X3 Secretariat  
Computer and Business  
Manufacturers Asso  
311 First Street NW,  
Washington, DC 20001

Review Letter # 55			
Subgroup Nominations :			
1	GEN	11	
2	DATA	13	
3	PROC	13	
4	DATA	14	
5	GEN	15	
6	GEN	16	
7	GEN	17	
8	CIO	18	
9		19	
10		20	

RF Monolithics, Inc.  
4441 Sigma Road  
Dallas, Texas 75244 U.S.A.  
(214) 233-2900  
Fax: (214) 387-8028  
Telex: 4630088

44

Gentlemen:

I agree that an updated FORTRAN standard is due. I can not agree that the proposed FORTRAN 8X standard is in the best interest of FORTRAN users.

Specifically, I have major concerns in the following areas:

The complexity of this standard is such that implementation is doubtful on smaller machines. This could totally remove FORTRAN capability from users of personal computers and even some moderately sized mini's. The language that 8X proposes is larger than PASCAL and ADA put together!

55.1

DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by mechanisms other than intrinsic types. The result will be inefficient.

55.2

The new source manipulation capabilities (MODULE/USE) are too complex and are untested in practice.

55.3

The new features added to enhance portability of numerical software are not clearly effective in obtaining the desired portability and are untested in practice.

55.4

The features chosen for possible obsolescence in the future are not justifiable. The cost of replacing statements such as DIMENSION, COMMON, and EQUIVALENCE would be intolerable to many, if not most, users.

55.5

I would very much like to see a CASE statement in FORTRAN, as well as array operations. However, the array concepts included in the proposed standard will likely degrade run-time and compiler performance.

55.8

55.6

I strongly recommend the X3J3 committee withdraw the proposed FORTRAN 8X standard. This standard is such a dramatic departure from the concept of FORTRAN that I am baffled. It would appear that X3J3 has invented a totally new language. If there is any support for this standard, it should be resubmitted for public review as a new language with a name other than FORTRAN.

55.7

44

The proposed FORTRAN BX, if approved, would have such a serious impact on our company that we would probably be forced to switch all new application development and most maintenance to another language.

Sincerely,



Wayne L. Fink, PE  
Manager of Computer Services

cc: Public Comment for dpANS Fortran Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY 10018

G.A. Andersen, President and CEO, RF Monolithics  
D. Ash, Vice President of Engineering

#56

44

L-7184-0000-911  
10 DECEMBER 1987

From: CARL MALEC  
Boeing Advanced Systems Co.  
Advanced Software Technology  
L-7184  
P.O. Box 3707 M/S 33-22  
Seattle, WA 98124-2207  
(206) 241-3387

Review Letter # 56			
Subgroup Nominations :			
1	GEN	11	
2	GEN	13	
3	GEN	13	
4	ED	14	
5	DATA	15	
6	PROC	16	
7	GEN	17	
8		18	
9		19	
10		20	

To: Public comments for Dpans Fortran Revision  
X3 Secretariat  
Attn: Gwendy Phillips  
Computer and Business Equipment Manufacturers Association  
Suite 300  
311 First Street, N.W.  
Washington, D.C. 20001-2178

cc: Public comments for Dpans Fortran Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, N.Y. 10018

Sylvia Sund  
SLAC, Bin 96  
P.O. Box 4349  
Stanford, CA 94305

87 DEC 28 12:05

X3

Subject: The new proposed standard FORTRAN-8X.

I have reviewed the FORTRAN-8X material sent me in October 1987.  
This consisted of the ANSI X3J3/S8.104 description of the proposed  
FORTRAN-8X standard, dated June 1987.

P. 290

The statements below are my opinions, the Boeing Company's official opinions have already been delivered to X3J3.

I agree with the decision of others to vote no, and I agree with their reasons given. But I also find significant additional reasons to oppose the new standard that have not been addressed previously by anyone else that I am aware of.

THEY ARE:

1. FORTRAN-8X is an attempt to create a new language, this new language is simply a Pascal/Ada look-alike. We already have Pascal, and Ada is a fast growing presence in the software community. We do not need another (less advanced kludge) Ada look-alike language. There is no justification for arbitrarily creating a new language similar yet inferior to already existing programming languages. (56.1)

2. FORTRAN has a place in the scientific and engineering computing workplace. It is a well streamlined and easy to use programming language. It is inappropriate to violate the acceptance of a widely accepted, easy to use standard. Changing FORTRAN just so it can use some (not all) of the new programming language constructs found in other languages does not make sense. If the systems analyst or programmer wants to use these other language features they should use a different language, and not change FORTRAN. That's why languages like Pascal and Ada exist. (56.2)

3. FORTRAN is not a logic processing language. Attempts to introduce an incomplete set of language constructs to allow limited logical processing only misleads the potential user. The logic programmer should be pointed to languages like LISP and Prolog. FORTRAN never was the answer to this problem and FORTRAN-8X definitely is not the answer. (56.3)

4. The FORTRAN-8X specification is difficult to read. If we learn only one lesson from the creators of Ada it should be: to make a language specification that can be read and understood. The X3J3 committee's specification of FORTRAN-8X is difficult to read and at times obscure. For the average programmer the new proposed standard document would be difficult to understand. (56.4)

(56.5) → Finally I would like to add my opinions and observations. Some undesirable changes are the variable definition changes, they are too bulky and complex. The specific constructs for entities, blocks, and scoping changes are clumsy and cause poor connectivity, making it difficult for the pieces of the language to connect and interact. (56.6) Some of the proposed changes are valid and good improvements. But, the currently proposed standard has exceeded its charter. It contains excessive and unnecessary changes that are not consistent with the structure and connectivity of FORTRAN-77. (56.7)

CARL MALEC

#57

44

# FLUOR DANIEL

3333 MICHELSON DRIVE  
IRVINE, CALIFORNIA 92730 U.S.A  
TELEPHONE (714) 975-2000

December 22, 1987

Public Comments for Dpans Fortran Revision  
X3 Secretariat  
Attn: Gwendy Phillips  
Computer and Business Equipment  
Manufactures Association  
Suite 300  
311 First street, N.W.  
Washington, DC. 20001-2178

Review Letter # 57			
Subgroup Nominations :			
1	GEN	11	
2	DATA	13	
3	GEN	13	
4	GEN	14	
5		15	
6		16	

Dear X3 Secretariat:

Enclosed in this letter are the accumulated comments of the Engineering Systems Group of Fluor Daniel. This group is part of the company's Information Systems Department and deals mainly with the development, support and maintenance of engineering applications. The members of the group are engineers with some formal programming training, but generally learn from on-the-job training and experience.

Being end-users of the Fortran language, we are not as concerned with the specifics of the proposed Fortran 8X compiler as we are with the general implications of a new language standard. We interpret the proposed language standard as being a complication of an already powerful programming language, in other words, "if it already works, don't fix it".

The array operations feature is probably the best idea of the new language standard, however, it would only be exercised with the development of a new application and could not be justified for existing in-house programs. The numerical computation enhancement could also be beneficial to increased accuracy, but again the cost associated with the training, implementation, testing, and, finally, verification of the application makes this benefit uneconomical.

57.1

57.2

In fact, to take advantage of the new language, a major portion of the old programs would have to be scrapped and new code would have to be written. This luxury is just not possible in our business considering the time and investment required. Although, some of our engineering applications were written many years ago, they are still a valuable asset to the company.

57.3

We especially object to the use of a "free" and "fixed" source form. The reasoning behind allowing two separate programming forms is unclear to us. To maintain large applications, the analyst must be able to quickly follow the coding to determine the operations performed. If the

57.4



44

FLUOR DANIEL

Computer and Business Equipment  
Manufactures Association

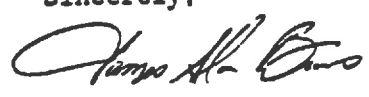
December 22, 1987  
Page 2

analyst is required to follow two different sets of statements, syntax, etc., his task has been unnecessarily complicated. Also, when enhancing the programs, if one analyst takes advantage of the "free" form, all analysts will be required to learn the new form to understand and interface with the "odd" program unit. This cost for training alone would be burdensome.

57.4  
cont.

In summary, we consider the proposed Fortran 8X language standard unacceptable in its current state and recommend it not be implemented as a programming language standard without considerable modification. We hope our comments can be of some assistance to you in providing a general, effective engineering programming language.

Sincerely,



J. A. Beers  
Fortran 8X Advisor,  
Fluor Daniel

JAB:aat  
565/jab010

cc: Public Comment for Dpans Fortran Revision  
Board of Standards Review  
American National Standards Institute;  
New York, NY 10018

Sylvia "Sunnie" Sund  
SLAC, Bin 96  
Stanford, CA 94305

# 58

44

Jim Armstrong  
Research Mathematician, CONVEX Computer Corp.  
701 N. Plano Rd.  
Richardson, TX 75081  
December 9, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the array expression syntax is a needed enhancement. I also feel that the ability to allocate arrays is a timely addition. I think that the MODULE USE feature is an unneeded change. I also think that the numerical precision control is an overly complicated extension. I am not happy that the POINTER data type was not in the draft. I am also shocked that the DO WHILE statement was not standardized. I feel that the COMMON statement must not be taken out of from a future FORTRAN standard.

58.1

58.3

58.2

58.4

58.5

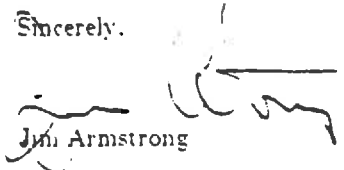
58.6

58.7

FORTRAN should remain an easy-to-learn language.

58.8

Sincerely,

  
Jim Armstrong

Review Letter # 58			
Subgroup Nominations :			
1	GEN	11	
2	DATA	13	
3	PROC	13	
4	DATA	14	
5	DATA	15	
6	CIO	16	
7	GEN	17	
8	GEN	18	
9		19	
10		20	

X3 C

'87 DEC 15 11:24:03

David M. Hill  
Northern Telecom  
2405 Golden Oaks  
Garland, Texas-75042  
December 16, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the ability to allocate arrays is a needed addition. I also feel that the CASE construct is a needed enhancement. I feel that the NAMELIST input/output is a timely improvement.

59.1

59.2

59.3

I believe that the IDENTIFY statement is a poorly designed modification to the language. I also believe that the MODULE/USE feature is an unwarranted change. I think that the abstract data typing is an inefficient addition.

59.4

59.5

59.6

59.7

59.9

I am unhappy that the IMPLICIT NONE statement was missing. I am also unhappy that the BIT data type was not included. I am not happy that the POINTER data type was not in the draft. I feel that the COMMON statement must not be deleted. I also believe that the EQUIVALENCE statement cannot be removed under any circumstances. I believe that the DIMENSION statement must not be removed from a future FORTRAN standard.

59.8

59.10

The committee should not be inventing a new language.

59.11

Sincerely,

  
David M. Hill

Review Letter # 59			
Subgroup Nominations :			
1	DATA	11	GEN
2	CIO	13	
3	CIO	13	
4	DATA	14	
5	PROC	15	
6	DATA	16	
7	DATA	17	
8	DATA	18	
9	DATA	19	
10	GEN	20	

#60



**King  
Computer Search, Inc.**

Technical Recruiting and Executive Search  
9221 LBJ Freeway, Suite 208  
Dallas, Texas 75243

(214) 238-1021

44

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I think that the array triplet notation is an important improvement. I also feel that the WHERE statement is a need extension. I think that the abstract data typing is an overly complicated change to FORTRAN.

I also feel that the numerical precision control is an unneeded modification to the language. I am unhappy that the IMPLICIT NONE statement was missing. I believe that the DOUBLE PRECISION statement should not be taken out of from any FORTRAN of the future.

I am opposed to the standard as currently proposed. Please fix the problems outlined in this letter in the proposed standard.

Sincerely,

*Tisha Motley*  
Tisha Motley, CPC

Review Letter # 60			
Subgroup Nominations :			
1	GEN	11	
2	GEN	13	
3	DATA	13	
4	DATA	14	
5	DATA	15	
6	GEN	16	
7		17	
8		18	
9		19	
10		20	

88 JAN 11 P1:43

X3 secret

#61



**King  
Computer Search, Inc.**

Technical Recruiting and Executive Search  
9221 LBJ Freeway Suite 208  
Dallas, Texas 75243

(214) 238-1021

44

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

61.1

61.2

I think that the WHERE statement is a valuable enhancement. I also believe that the CASE construct is a helpful addition. I think that the MODULE/USE feature is an unneeded addition. I also feel that the numerical precision control is a wasteful change. I am disappointed the the INCLUDE statement was not standardized. I am also unhappy that the POINTER data type was not defined. I believe that the COMMON statement should not be removed ever. I also think that the EQUIVALENCE statement must not be deleted under any circumstances.

61.3

61.4

61.5

61.6

61.7

I agree with IBM's criticisms of the draft proposed standard.

Sincerely,

*Sally King*  
Sally King, CPC

88 JUN 11 P 1:43

X3

Review Letter # 61			
Subgroup Nominations :			
1	GEN	11	
2	CIO	13	
3	PROC	13	
4	DATA	14	
5	PROC	15	
6	DATA	16	
7	GEN	17	
8		18	
9		19	
10		20	

#62



**King  
Computer Search, Inc.**

Technical Recruiting and Executive Search  
9221 LBJ Freeway, Suite 208  
Dallas, Texas 75243

(214) 238-1021

44

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

62.7

62.2

I feel that the IDENTIFY statement is a pointless extension. I am shocked that the 3) POINTER data type was missing. I am also unhappy that the INCLUDE statement was not in the draft. I think that the COMMON statement cannot be taken out under any circumstances. I also think that the EQUIVALENCE statement should not be taken out of any FORTRAN of the future. I believe that the DIMENSION statement must not be deleted from any FORTRAN of the future.

62.3

62.4

I feel that slower compilers are unacceptable.

62.5

Sincerely,

*Sandy Davis*

Sandy Davis

88 JUN 11 P 1:43

X3

Review Letter # 62			
Subgroup Nominations :			
1	DATA	11	
2	DATA	13	
3	PROC	13	
4	GEN	14	
5	GEN	15	
6		16	
7		17	
8		18	
9		19	
10		20	

Review Letter # 63			
Subgroup Nominations :			
1	GEN	11	PROC
2	GEN	12	DATA
3	GEN	13	GEN
4	PROC	14	GEN
5	PROC	15	GEN
6	PROC	16	GEN
7	DATA	17	GEN
8	PROC	18	GEN
9	ED	19	
10	PROC	20	

#63

Alan J. Wallcraft  
 JAYCOR  
 NORDA Code 323  
 NSTL  
 MS 39529  
 01/05/88

44

Dear Sirs,

This draft Proposed Revised American National Standard Programming Language Fortran, X3.9-198x. I am in broad agreement with the draft revision, and in particular I approve of the inclusion of derived data types, array processing facilities, and numerical computation enhancements. But must object to its approval because of the problems I have in some areas as outlined below.

## 1) Source Form.

a) The maximum of 132 characters per line in free source form allows a full line to be displayed on many terminals and most lineprinters. But all FORTRAN compilers I have ever used add a line or statement number to the beginning of the line in their compiler listings. The maximum should be reduced to 126 characters or less, I would prefer 120, to allow compiler listings that include compiler generated line numbers to be printed on lineprinters. 63.1

b) The standard does not appear to address comment lines outside program units. These can be very useful for providing general comments about, say, a group of subroutines. Comment lines outside program units should be explicitly allowed and they should be associated with the following (rather than the preceding) program unit. 63.2

c) The statement separator, ";", is not well defined. For example continuation is defined in terms of "lines" but this does not make sense when a line can have multiple statements on it. In either source form defining ";" as equivalent to a newline and 6 spaces would remove any ambiguity (if this is really the intent). However some very strange text then becomes legal FORTRAN, for example: 63.3

```
IF(A.NE.B)THEN;;T=A;A=B;B=T;;;;;ELSE;;A=0;ENDIF;;;
A=B;X=&
&Y
```

## 2) Statements in Scoping Units.

a) Statement functions should not be allowed in internal subprograms. Otherwise statement functions cannot be made obsolescent because there is no FORTRAN 8X replacement for statement functions in internal subprograms. 63.4

b) COMMON statements should not be allowed in module and internal subprograms. This leads to no loss of functionality, since any required COMMON can be defined in the module or the host subprogram, but removes the possibility of confusion about multiple names for the same object. For example the following is, I think, currently 8X standard:

```
SUBROUTINE TEST
COMMON/A/B,C,D,E
CALL TEST_INTERNAL
CONTAINS
SUBROUTINE TEST_INTERNAL
COMMON/A/I,J,K
K=I;B=C;E=D ! can use I,J,K and B,C,D,E.
END TEST_INTERNAL
END TEST
```

63.5

44

### 3) Interface Block.

It makes no sense to allow the redefinition of the arguments to an internal procedure. Therefore an interface block should not refer to an internal procedure.

63.6

### 4) RANGE.

RANGE adds little or no functionality over ALIAS, and in my judgement will be very infrequently used. RANGE and the associated intrinsic functions should be deleted.

63.7

### 5) OVERLOADING.

At present overloading is the default, but the USE statement provides the potential for a large number of procedures to have explicit interfaces and for these interfaces to be hidden from the casual reader of the text of a subprogram. In fact the major disadvantage of the USE statement is that it can move information vital to the understanding of a routine to some far away section of the code. So it is entirely possible for a procedure to be unknowingly overloaded, and an error in the argument list to such a procedure call could cause a bug that would be very difficult to find. Moreover it is not clear to me that linkers and code development software (such as library utilities) can handle multiple routines with the same name, i.e. the development of a FORTRAN 8X programming environment could involve far more than just the provision of a compiler. Overloading should be included, but not by default, and it should be required that all subprograms except, perhaps, internal subprograms have unique names. The syntax for overloading could be something like, for example:

63.8

```
INTERFACE
SUBROUTINE A_I(I) GENERIC(A)
INTEGER I
END INTERFACE
INTERFACE
SUBROUTINE A_X(X) GENERIC(A)
REAL X
END INTERFACE
```



44

The linker would then never need to know about overloading, since the compiler would substitute the appropriate actual subroutine call for the generic call. With 31 character names there should be no problem providing unique names for all subprograms, FORTRAN 77 has that requirement today with 6 character names. In the response to Ivor Philips ballot that suggested an OVERLOAD statement the single argument against the feature was that non-default overloading could always be extended by a vendor to default overloading, leading to less standard conformance. But this is true of many FORTRAN 8X features, for example non-default RECURSIVE could be vendor extended to default RECURSIVE. This argument is not sufficient to justify the exclusion of syntax that may reduce the cost of implementing FORTRAN without reducing the functionality of the language.

#### 6) RECURSIVE.

a) Recursion is not included in the index (Appendix G), all keywords should be in the index. 63.9

b) So far as I can tell the total discussion of recursion is about ten lines long. Recursion is by now a well known programming technique, but more discussion is required. For example I think the intent is that there be a single copy of all local variables with the SAVE attribute, i.e. these are global to all instances of the subprogram, but all other local variables would be allocated on the stack and each active instance of the subprogram would have its own distinct set. I am not entirely sure, however, that this is the only interpretation of the current document. My own preference would be to ban the SAVEing of local variables in RECURSIVE subprograms, because all local variables, SAVED and non-SAVED, act the same way in non-recursive subprograms but SAVED variables have side-effects in recursive subprograms. It is possible to use registers to hold SAVED variables in non-recursive subprograms, provided their values are stored before returning, but this is not possible in recursive subprograms. SAVED variables are just another form of global storage in recursive subprograms, and the identical effect can be obtained by putting the subprogram in a MODULE that contains the SAVED variables as PRIVATE module variables. 63.10

#### 7) Intrinsic Procedures.

a) A new procedure, say CPU\_CLOCK, similar to SYSTEM\_CLOCK but supplying the actual CPU time used by the program should be provided. All FORTRANs I have ever used has this capability in some non-standard form, so its implementation should not be a problem and in any case there would be the option to not implement the routine in a standard way. The interpretation of the routine on machines with multiple CPUs may be a problem, vendors who intend to support such configurations should be asked to propose the needed extensions. 63.11

#### 8) Removed Extensions.

a) I would like a BIT data type in FORTRAN, but only to provide array MASKING with low storage overhead, i.e. I would support an extension to the LOGICAL data type, such as LOGICAL(LEN=1), in place of a full BIT data type. Implementors would then have the option of supplying or not 63.12

supplying bit or byte length LOGICAL as well as the default word length LOGICALs. Non-default LOGICAL variable could be used everywhere default LOGICALs are legal, except for COMMON storage.

b) I support the removal of all the other features in Appendix F, including the removal of significant blanks, and would add the RANGE capability to the list.

#### 9) Incremental Features.

a) I support the concept of obsolescent and depreciated features, but I am not sure that it will have much practical effect. For example we use NAMELIST I/O heavily and simply will not purchase a computer and/or a FORTRAN 77 compiler that does not support this non-standard feature. Because there are people like us out in the market place almost all vendors support NAMELIST, and I suspect that the depreciated and obsolescent features will continue to be supported, as extensions to the standard if necessary, as long as FORTRAN exists for the same reason.

b) Of the obsolescent features the only one in wide use is the various forms of DO termination. Never the less I fully support its inclusion in the list, since it is simple to make such code standard conforming.

c) It is not clear to me why the DIMENSION statement is "completely redundant" in FORTRAN 8X but not in FORTRAN 77, i.e. why it is depreciated rather than obsolescent. In fact the DIMENSION statement is not redundant in either language because it is the only way to use implicit typing for arrays. Without the DIMENSION statement the IMPLICIT statement is largely useless, and since implicit typing has not been depreciated the DIMENSION statement should be removed from the list.

d) Statement functions can only be depreciated if they are banned from internal subprograms (since there is no FORTRAN 8X replacement for such statement functions).

e) I support the inclusion of storage association features on the depreciated list. In practice, however, I expect these features will always be available in actual FORTRAN compilers.

Yours sincerely

  
Alan Wallcraft



#64

44

FROM THE COMPUTER LABORATORY  
CENTRE FOR COMPUTER STUDIES

TELEX 627095  
TEL 051 - 709 6022

CHADWICK BUILDING P.O. BOX 147 LIVERPOOL L69 3BX

# The University of Liverpool

JSM/MJ/2951

17th December, 1987

X3 Secretariat/CBEMA  
Fortran Public Review  
311 First St., N.W.  
Suite 500  
Washington D.C.  
20001-2178  
U.S.A.

Review Letter # 64			
Subgroup Nominations :			
1	CIO	11	
2	DATA	13	
3	DATA	13	
4	DATA	14	
5	DATA	15	
6		16	

87 DEC 28 11:34  
X3

The language as described in the document X3J3/S8.1 is a long-awaited, welcome revision of the Fortran 77 standard. It will provide a good migration path to a modern language for computing professionals and will greatly improve the teaching of programming and computer science through the medium of Fortran.

I should like to make the following comments on the draft revision which I have ordered in, what I regard to be, decreasing levels of importance.

## 1. Handling of Errors

64-1

As an end-user, teacher, and supporter of Fortran Compilers, I feel strongly that the standard should include references to the capability of processors to detect errors, and report them. For instance I think it is inadequate for the standard to regard a program which violates array bounds as merely non-standard conforming.

The standard should include words which force implementers to write compilers which are at least "capable" of being run in such a way that such errors are reported. A requirement of this sort would not impede the run-time efficiency of programs (e.g. in real time applications) since code for it need only be planted at compile time if required by the user. A restriction of this sort would prevent "early" versions of compilers being issued without basic debugging checks which would only serve to give Fortran a bad name. (Vendors are usually forced to incorporate facilities of this sort eventually because of user pressure).

## 2. Pointers

64-2

The lack of a true pointer facility is a serious omission from the language. A pointer facility is "almost" in the language anyway via

ALIAS, ALLOCATE and IDENTIFY and minutes of X3J3 show that it could be included quite consistently without too much effort.

44

We surely do not want future generations of Fortran programmers to continue to model data structures using arrays and indices.

3. Variant Data Structures (64-3)

I'm sure X3J3 are well aware of the arguments for including this facility. So many problems can be modelled using such data structures that it seems silly to omit it. Since this facility, which takes up very little description, was moved to the Appendix in the interests of reducing the size of the language, one wonders how the RANGE facility remains in the standard when it appears to be useful to a more limited application area.

← (64-4)

4. General Precision Integer (64-5)

I fail to understand why this has been left out when such a good job has been done for Reals. Inclusion of this facility would add to the functionality and improve the regularity of the language.

The above comments are brief but, I hope, to the point. I know that comments of this sort, including detailed arguments, will be submitted from other sources. My intention is to add my voice to those users of Fortran who, I'm sure, are anxious to see the new standard emerge as soon as possible, but who also want it to be as useful as it can be made to as wide an audience as possible.

Yours faithfully,



Dr S. Morgan

~~P. 304~~

45  
45a

107(\*)TAH-1

To: X3J3

From: Tracy Hoover

Date: 4 February 1988

Subject: Backward References in Section Notes

1. Background

Masscomp was asked at the last meeting to provide backward references in the Section Notes for Sections 4, 5, and 6, to the text in the associated section of S8. Most of my suggestions are for placing these references after the subsection headings that were added at the last meeting (106 (\*) TAH-1). Where a subsection heading is not referenced, the backward reference should be placed at the end of the sentence.

For the actual changes that were made to the Section Notes, please see pp. 19-20 of S16.106 (Approved Changes document).

2. Proposal

Add the following backward references to the Section Notes for Sections 4, 5, and 6.

- P. C-2, 1.1a Replace ``C.4.1 Zero.`` with ``C.4.1 Zero (4.<sup>3</sup>4.1).``
- P. C-2, 1.3 Replace ``C.4.2 Intrinsic and Derived Data Types.`` with ``C.4.2 Intrinsic and Derived Data Types (4.3, 4.4).``
- P. C-2, 1.23 Add ``(4.3.1.2)`` before ``.`` at end of sentence
- P. C-2, 1.24 Add ``(4.3.2.1)`` before ``.`` at end of <sup>last</sup> sentence
- P. C-2, 1.42 Replace ``C.4.3 Precision and Exponent Range Parameters.`` with ``C.4.3 Precision and Exponent Range Parameters (4.3.1.2).``
- P. C-2, 1.50 Replace ``C.4.4 Components and Storage of Derived Types.`` with ``C.4.4 Components and Storage of Derived Types (~~4.4~~<sup>4.4.1</sup>).``
- P. C-3, 1.11a Replace ``C.5.1 Type Declaration Statements.`` with ``C.5.1 Type Declaration Statements (5.1).``
- P. C-3, 1.21 Add ``(5.1.1.2)`` before ``.`` at end of sentence
- P. C-3, 1.22 Replace ``C.5.1.1 RANGE Attribute.`` with ``C.5.1.1 RANGE Attribute (5.1.2.8, 5.2.8).``

45a

- ~~P. C-3, 1.36~~ Replace ``C.5.2 Array Element References.`` with  
``C.5.2 Array Element References (5.1.2.4).``
- P. C-3, 1.39 Replace ``C.6.1 Substrings.`` with  
``C.6.1 Substrings (6.1.1).``
- P. C-3, 1.42 Replace ``C.6.2 Structure Components.`` with  
``C.6.2 Structure Components (6.1.2).``

46

To: X3J3

107(\*)JKR-2

From: John Reid

Subject: Meeting minutes

Date: 5th February 1988

The attached scribe notes arrived too late for inclusion in the minutes of the last meeting. Please regard them as supplementing page 44 of X3J3/215.

## 24 Response to Digital's X3 ballot

Discussion leader: Hendrickson

Scribe: Adamczyk

- Hendrickson:** It seems like we removed a lot, but we really just tightened it up.
- Motion:** Document 118 (Hendrickson, Marusak).
- Burch:** On the bit intrinsic discussion, change "no votes" to "those opposing the dpANS". (Hendrickson, Marusak accept this.)
- Moss:** Change the last sentence of multi-byte character section to match pointers section. This avoids the promise that the feature will be implemented. (Hendrickson disagrees.)
- Weaver:** The status of the committee is that no committee time will be given to multi-byte characters (vote on Tuesday). First sentence should also be changed ("X3J3 ... is studying").
- Burch:** Effect was to table consideration, not to say we would never do it.
- Weaver:** Vote was "given an adequate technical proposal, should it be in 8X." It did not get a 2/3 majority.
- Hendrickson:** Agree to change last sentence to match pointers. Suggest that first sentence be "has studied".
- Weaver:** I disagree. I move to amend. (seconded: Hoover.)
- Moss:** I have suggested wording. Strike first sentence. Add "Japanese" before "and Chinese". Last sentence as changed -- to match pointer section ("As yet, an adequate consensus..."). "have studied" is not honest -- we have studied, but do not plan to do anything about it.
- Hendrickson:** I accept the amendment.
- Marusak:** I do not accept it.

5th February 1988

1 of 2

107(\*)JKR-2

P. 307

- La gassé: What Len [Moss] has said is right, but is not reflected in his wording. I agree with Dick [Weaver] that we should call a spade a spade.
- Hirchert: We failed on Tuesday to *immediately* create a task force. The long-term prospects are unknown. This is not an outright rejection of the proposal.
- Marusak: I think this reflects the committee much better than Weaver's "rather draconian" statement. One has to freeze the status at some point. This is more accurate.
- Moss: Nevertheless, the vote is that currently we are not studying this issue.
- La gassé: I suggest the wording "no further activity at this time".
- Hirchert: This is not an out-and-out rejection -- only "at this time".
- Ellis: There will be no formal action, but there will be action. Just nothing happening in full committee.
- Matheny: The subgroup spent several hours -- We *have* addressed this topic at this meeting.
- Paul: This action is pending the public review process.
- Vote: Amendment passes, 13-12.
- La gassé: Suggest adding "within X3J3" to the wording on lack of consensus on multi-byte characters.
- Hendrickson: Change sentence on lack of consensus in each of 1, 2, and 4. (Marusak agrees.)
- J. Martin: In consensus section, change "among the committee" to "within the committee". (Hendrickson, Marusak agree.)
- Adams: In the procedural issues section, keep the first sentence. I feel quite strongly that procedural rules have been followed. I would like to see that sentence in.
- Marusak: The subgroup thought the other way was better. It's a positive statement rather than a denial.
- Adams: I felt the ballot was accusing me personally.
- Marusak: It's still better to respond positively. It's like "Honest Abe Lincoln" rather than "I-am-not-a-crook Nixon".
- Ellis: A positive statement is better. To say we've never done *anything* wrong is a dangerous thing to put down.
- Final Vote: 23-4. Passed with revisions.





To: X3J3

107(\*)JKR-3

472

From: John Reid

Subject: Changes to S16.107

Date: 2nd February 1988

Proposal. Make the following changes to S16.107:

- ~~X~~ Item 39. Change 'lowerbound' to 'lower bound' and 'upperbound' to 'upper bound'.
- ~~X~~ Items 47, 73, 82, 83, and 135. 'Constraint' should not be set in bold.
- ~~X~~ Item 47. Change '*parens*' to '*parent*'.
- ~~X~~ Items 53 and 224. Delete '(Tech.)'.
- ~~X~~ Item 63. Delete ', provided ... ]'. (See item 171.)
- ~~X~~ Item 71. Delete '.' before '!Selects'.
- 7. Item 72. Change 'CONSTRUCT' to 'Construct'.
- 8. Item 73, 74, and 84, titles of sections 8.1.4.1, 8.1.4.1.1, 8.1.4.1.2, 8.1.4.2, and 8.1.4.5. Change 'construct' to 'Construct'.
- ~~X~~ Item 73, R817. Move '*end-do*' left to align with '*do-stmt*'.
- ~~X~~ Item 73, first two constraints. Use tiny font for 'default real, or double precision real'.
- ~~X~~ Item 73. Use tiny font for the whole of 8.1.4.1.2.
- ~~X~~ Item 73, 8.1.4.1.2, second constraint, line 2. Change '*stmts*' to '*stmt*'.
- ~~X~~ Item 74. Use tiny font for the second paragraph of 8.1.4.2.
- 14. Item 75. Change 'i.e.' to 'that is'.
- ~~X~~ Item 82, page 9, lines 1-3, 5-7, and 7-9. Use tiny font for the sentences 'If ... executed.', 'In ... executed.' and 'Unless ... executed'.
- 16. Item 84, page 12, penultimate line of Example 7. Change 'ENDDO' to 'END DO'.
- 17. Item 85. Change 'line 23' to 'lines 23-24'.
- ~~X~~ Item 89. Change 'correct' to 'current'.
- ~~X~~ Item 103. Change 'line 28' to 'line 18'.
- ~~X~~ Item 147. Add 13-12, line 32.
- ~~X~~ Item 147 and 149. Delete the changes to Appendix F. (See item 278.)
- ~~X~~ Item 148. Delete 13-12, line 32 and add 13-20, line 24.
- ~~X~~ Items 163, 260, and 273. Delete '(Ed.)'.
- ~~X~~ Item 172. Add matrix brackets around the matrix.
- 25. Item 197, line 3. Change 'lines 42-44' to 'lines 42-45'.
- ~~X~~ Item 214. Change 'lines 1 and 20' to 'line 20'.
- ~~X~~ Item 231. This should be part of item 230.

2nd February 1988

1 of 1

107(\*)JKR-3

p. 309

48

107-CDB-6  
February 5, 1988

From : Carl Burch

To : X3J3

Subj : Subgroup Nominations for Public Review Letters 10-21

Please find attached the annotated copies of Public Review letters 10-21, marked with my recommendations for subgroup assignments. All subgroup assignments are negotiable between the Subgroup Chair and the Public Review Working Group. Any omissions noted should be brought to the attention of the Public Review Working Group.



Tony Linthicum  
Convex Computer Corporation  
701 N. Plano Rd  
Richardson, Tx. 75243  
December 7, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

10-1

10-3

10-2

10-4

10-5

10-6

I feel that the array triplet notation is a helpful improvement. I also feel that the WHERE statement is an useful improvement. I believe that the inline comment facility is an important extension. I feel that the numerical precision control is an overly complicated change to FORTRAN. I also feel that the internal procedures is an unnecessary modification to the language. I am unhappy that the 3) POINTER data type was not standardized. I am also shocked that the DO WHILE statement was missing. I feel that the COMMON statement must not be taken out of under any circumstances. I also think that the EQUIVALENCE statement cannot be removed from any FORTRAN of the future. I believe that the statement function should not be taken out of ever.

10-7

10-8

Designating COMMON and EQUIVALENCE for future removal is unacceptable.

Sincerely,

*Tony Linthicum*  
Tony Linthicum

Subgroup Nominations:

- 10-1 GEN
- 10-2 GEN
- 10-3 GEN
- 10-4 DATA
- 10-5 PROC
- 10-6 DATA
- 10-7 CIO
- 10-8 GEN

# 11

48

Bonnie Lee Hill  
Convex Computer Corporation  
2405 Golden Oaks  
Garland, Texas 75042  
December 16, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the CASE construct is an useful improvement. I also feel that the NAMELIST input/output is a needed extension.

11-3 → I believe that the MODULE/USE feature is an inefficient change to FORTRAN. I also believe that the abstract data typing is an unneeded modification to the language. I believe that the numerical precision control is an unwarranted extension.

11-5 → I am disappointed that the IMPLICIT NONE statement was missing. I am also shocked that the BIT data type was not defined. I am concerned that the POINTER data type was not included. I believe that the COMMON statement should not be removed from the language. I also feel that the EQUIVALENCE statement must not be deleted from a future FORTRAN standard. I think that the DIMENSION statement must not be deleted from a future FORTRAN standard.

The committee should not be inventing a new language. Instead, the committee should be setting standards for existing practices and intrinsic functions.

Sincerely,

*Bonnie Lee Hill*  
Bonnie Lee Hill

X3  
'87 DEC 28 PM 12

Subgroup Nominations :	
11-1	CIO
11-2	CIO
11-3	PROC
11-4	DATA
11-5	DATA
11-6	DATA
11-7	DATA
11-8	GEN
11-9	GEN
11-10	DATA

48

Ron Lieberman  
Software Engineer/Convex Computer Corp.  
1408 Kesser Dr.  
Plano, Texas. 75023  
December 6, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

12-1

12-2

I believe that the CASE construct is a timely enhancement. I believe that the numerical precision control is a superfluous change to FORTRAN. I am unhappy that the VMS RECORD structures was not in the draft. I think that the COMMON statement must not be taken out of under any circumstances. I also think that the DOUBLE PRECISION statement cannot be removed under any circumstances.

12-3

12-4

The committee should be standardizing existing practice.

12-5

Sincerely,

*Ron Lieberman*

Ron Lieberman

Subgroup Nominations :	
12-1	CIO
12-2	DATA
12-3	DATA
12-4	GEN
12-5	GEN

#13

48

Susan Linthicum  
Philadelphia Life Insurance  
500 N. Akard  
Dallas, Tx. 75221  
December 7, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the CASE construct is a timely enhancement. I also feel that the ability to allocate arrays is a timely addition. I believe that the MODULE/USE feature is an unnecessary change to FORTRAN. I also believe that the internal procedures is a poorly designed modification to the language. I am disappointed that the VMS RECORD structures was missing. I think that the computed GOTO statement cannot be removed ever. I also feel that the ENTRY statement must not be deleted from a future FORTRAN standard. I think that the alternate RETURN statement cannot be deleted from any FORTRAN of the future.

13-1

13-2

13-3

13-4

13-6

13-5

The committee should not be inventing a new language.

13-7

Sincerely,

*Susan Linthicum*  
Susan Linthicum

Subgroup Nominations :	
13-1	CIO
13-2	DATA
13-3	PROC
13-4	PROC
13-5	DATA
13-6	GEN
13-7	GEN

#14

48

Randall Mercer  
Convex Computer Corporation  
6406 Gila Court  
Plano, Texas 75023  
December 7, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

I believe that the CASE construct is a valuable addition. I believe that the IDENTIFY statement is an unnecessary extension. I also feel that the numerical precision control is an unneeded extension. I am shocked that the INCLUDE statement was missing. I am also concerned that the POINTER data type was not defined. I think that the COMMON statement cannot be removed ever. I also believe that the ENTRY statement cannot be removed under any circumstances.

14-1

14-2

14-3

14-5

14-6

14-4

The committee should not be inventing a new language.

14-7

Sincerely,

*Randall Mercer*

Randall Mercer

Subgroup Nominations :	
14-1	CIO
14-2	GEN
14-3	DATA
14-4	GEN
14-5	DATA
14-6	GEN
14-7	GEN

#15

48

John William Torkelson  
Convex Computer Corporation  
PO Box 833851  
Richardson, TX 75083-3851  
December 7, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the CASE construct is a helpful addition. I feel that the MODULE/USE feature is an overly complicated addition. I am shocked that the DO WHILE statement was not included. I am also disappointed that the VMS RECORD structures was not defined.

FORTRAN should remain an easy-to-learn language.

Sincerely,

*John William Torkelson*  
John William Torkelson

15-1

15-2

15-3

15-4

15-5

X3  
'87 DEC 11 11:54

Subgroup Nominations :	
15-1	CIO
15-2	PROC
15-3	CIO
15-4	DATA
15-5	GEN



#16

48

CONVEX Computer BV  
World Trade Center  
Strawinskylaan 737  
1077XX Amsterdam  
Netherlands

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

16-1 I think that the WHERE statement is a valuable enhancement. I also believe that
 16-5 the CASE construct is a helpful addition. I think that the MODULE/USE feature is
 16-3 an unneeded addition. I also feel that the numerical precision control is a wasteful
 16-4 change. I am disappointed that the INCLUDE statement was not standardized. I am
 16-2 also unhappy that the POINTER data type was not defined. I believe that the
 16-6 COMMON statement should not be removed ever. I also think that the
 16-7 EQUIVALENCE statement must not be deleted under any circumstances.

I agree with IBM's criticisms of the draft proposed standard. 16-8

Sincerely,

*Jan Boerhout*

Jan Boerhout-Software Analyst

Subgroup Nominations :	
16-1	GEN
16-2	CIO
16-3	PROC
16-4	DATA
16-5	GEN
16-6	DATA
16-7	GEN
16-8	GEN

X3 J3  
'87 DEC 11 P12:55

#17

48

LTV Aerospace & Defense  
P.O. Box 225907 MSF-73  
Dallas, TX 75265  
October 21, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I believe that the CASE construct is a needed addition. I believe that the numerical precision control is poorly designed modification to the language. I am disappointed that the DO WHILE statement was not included. I am also shocked that the VMS RECORD structures was missing. I am unhappy that the INCLUDE statement was not defined. I feel that the COMMON statement must not be removed ever.

17-1

17-2

17-3

17-5

17-4

17-6

Sincerely,

*Gary Ramsey*  
Gary K. Ramsey

Subgroup Nominations :	
17-1	CIO
17-2	DATA
17-3	CIO
17-4	DATA
17-5	GEN
17-6	GEN

48

Duke University  
Dept of Computer Science  
Durham NC 27705  
October 21, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I feel that the array triplet notation is a valuable extension. I also think that the ability to allocate arrays is a timely extension. I believe that the CASE construct is an useful addition. I am disappointed that the 3) POINTER data type was not standardized. I am also surprised that the VMS RECORD structures was missing. I feel that the DOUBLE PRECISION statement cannot be removed under any circumstances.

18-1

18-3

18-2

18-4

18-5

18-6

Sincerely,

*Jonathan Becher*

Jonathan Becher

Subgroup Nominations :	
18-1	GEN
18-2	DATA
18-3	CIO
18-4	DATA
18-5	DATA
18-6	GEN

#19

48

John P. Kole  
13323 Maham Road #501  
Dallas, Texas 75240  
December 9, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

I believe that the inline comment facility is a valuable extension. I also believe that the NAMELIST input/output is a helpful addition. I believe that the MODULE/USE feature is a poorly designed modification to the language. I also believe that the abstract data typing is a wasteful modification to the language. I am shocked that the INCLUDE statement was missing. I am also not happy that the IMPLICIT NONE statement was missing. I believe that the COMMON statement cannot be deleted from a future FORTRAN standard. I also feel that the EQUIVALENCE statement should not be taken out of from a future FORTRAN standard.

19-1

19-2

19-3

19-4

19-7

19-6

19-5

The committee should not be inventing a new language.

Sincerely,

19-8

John P. Kole

Subgroup Nominations :	
19-1	GEN
19-2	CIO
19-3	PROC
19-4	DATA
19-5	GEN
19-6	GEN
19-7	GEN
19-8	GEN

#20

48



THE UNIVERSITY OF TEXAS AT DALLAS

Center for Applied Optics  
Box 830688  
Richardson, Texas 75083-0688

November 20, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

20-1

20-2  
20-3

I believe that the MODULE/USE feature is a pointless change. I am disappointed that the POINTER data type was not defined. I am also concerned that the DO WHILE statement was not standardized. I feel that the DOUBLE PRECISION statement must not be removed under any circumstances. I also think that the DIMENSION statement cannot be deleted from a future FORTRAN standard.

20-4

I agree with IBM's criticisms of the draft proposed standard.

Sincerely,

20-5

*Nolan Davis*  
Nolan Davis

Subgroup Nominations :	
20-1	PROC
20-2	DATA
20-3	CIO
20-4	GEN
20-5	GEN

87 DEC 28 P2:04

X3

p. 321

18)

SUBGROUP ASSIGNMENTS

SUBGROUP	ASSIGNED COMMENTS (Letter.#)
CIO	6.5 8.1 10.7 11.1 11.2 12.1 13.1 14.1 15.1 15.3 16.2 17.1 17.3 18.3 19.2 20.3 21.1 21.2

SUBGROUP ASSIGNMENTS

50

SUBGROUP	ASSIGNED COMMENTS (Letter.#)
DATA	2.7 2.9 2.10 2.11 5.2 5.3 6.6 9.4 9.5 10.1 10.4 10.6 11.4 11.5 11.6 11.7 11.10 12.2 12.3 13.2 13.5 14.2 14.3 14.5 15.4 16.4 16.6 17.2 17.4 18.1 18.2 18.4 18.5 19.4 20.2 21.4 21.6 21.7

SUBGROUP ASSIGNMENTS

50

SUBGROUP	ASSIGNED COMMENTS (Letter.#)
GEN	1.1 1.2 1.3 1.4 2.0 2.1 2.2 2.3 2.4 2.5 2.8 2.12 2.13 3.1 4.1 4.2 5.0 5.1 5.5 6.0 6.1 6.3 6.4 7.1 7.2 7.3 7.4 7.5 9.0 9.1 9.2 9.3 9.7 10.2 10.3 10.8 11.8 11.9 12.4 12.5 13.6 13.7 14.6 14.7 15.5

(Cont.)



SUBGROUP ASSIGNMENTS

50

(Cont.)

SUBGROUP	ASSIGNED COMMENTS (Letter.#)
GEN	16.1 16.7 16.8 17.6 18.6 19.1 19.6 19.7 19.8 20.4 20.5 21.5 21.8 21.9

SUBGROUP ASSIGNMENTS

SUBGROUP	ASSIGNED COMMENTS (Letter.#)
PROC	2.6 5.4 6.2 6.7 9.6 10.5 11.3 13.3 13.4 14.4 15.2 16.3 16.5 17.5 19.3 19.5 20.1 21.3

107 (X) JCA-6  
P. 1 of 1

(51)

UNISYS

December 14, 1987

C. A. Kachurik  
X3 Secretariat  
CBEMA, Suite 500  
311 First St., NW  
Washington, DC 20001

Subject: FORTRAN 8X  
Ref.: X3LB 916

Dear Cathie:

X3J3's undated response, X3/87-11-126-X, was received in my office on Dec. 7 and circulated for comment. We appreciate the time taken by X3J3 to review the Unisys comments and form the response to us in a timely manner.

While the TC responded to each of our concerns, i.e., compilation cost, object code size, etc., the conclusions reached are not sufficiently adequate to change our vote to yes. We are still concerned that 8X is too big an increment over 77 and that the addition of "modern" concepts has distorted the language nearly beyond recognition.

Consequently, please be advised that the Unisys no vote stands as submitted.

Very truly yours,

*MWB*

M. W. Bass  
Member, X3  
Unisys Corporation

cc: Jeane Adams  
R. P. Kelble  
L. R. Rolison

p. 329

107 (X) JCA-7

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

Doc. No.: **X3/87-12-055-X.I.S.M**

Date: December 14, 1987

Project:

Ref. Doc.:

Reply to:

52

**TO: Members X3, SPARC, JT/AC, SMC  
Officers, X3/TC's, SC's and SPARC/SG's**

**SUBJECT: Transmittal of December 1987 Edition, SD-4, Project Manual**

Attached is the newly updated SD-4. We have added all new projects that have been approved since the last printing in June 1987.

We apologize for missing our last quarter production date. I'm sure that you realize with a 50% turnover in staff assignment, placement and training took a tremendous amount of time and the addition of 11 projects assigned since June fell to a lower than usual priority.

If you have any questions or suggestions, please direct your inquiries to Lynn Barra who is now handling production of this document. Lynn can be reached at 202-737-8888 Ext. 52. We welcome all comments. Now that Lynn has done the first, you can expect SD-4 on a timely basis.

Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: December 1987 SD-4

p. 330

X3/SD-4

December 1987

52

ACCREDITED STANDARDS COMMITTEE\*  
X3-INFORMATION PROCESSING SYSTEMS

---

PROJECTS MANUAL

---

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

SECRETARIAT:

Computer & Business Equipment Manufacturers Association



## X3 Standing Documents

This document is one of a series, developed by X3 and the X3 Secretariat, which provides a "data base" of information on Accredited Standards Committee X3 - Information Processing Systems. Each document is updated periodically on an individual basis.

The series is intended to serve several purposes:

- o To describe X3 and its program to inquirers
- o To inform committee members of the organization and operation of X3
- o To provide a system of orderly administration incorporating the procedures required by ANSI together with supplements approved by the X3 Secretariat, for the guidance of X3 officers, members, subgroups and the Secretariat staff.

The series of Standing Documents consists of the following:

X3/SD-0	Informational Brochure - September 1985
X3/SD-1	Master Plan - May 1987
X3/SD-2	Organization & Procedures - October 1985
X3/SD-3	Project Proposal Guide - May 1987
X3/SD-4	Projects Manual - December 1987
X3/SD-5	Standards Criteria - September 1984
X3/SD-6	Membership and Officers - December 1987
X3/SD-7	Meeting Schedule and Calendar - December 1987
X3/SD-9	Policy and Guidelines - (to be issued)
X3/SD-10	X3 Subgroup Annual Report Format - June 1987

### SD-4, X3 PROJECTS MANUAL

X3/SD-4 provides a listing of the current X3 projects, arranged by technical discipline and cross-referenced to the related ISO/TC97 projects, proposals and approved standards.

Corrections and suggestions for improvement will be welcomed, and should be addressed to:

X3 Secretariat/CBEMA  
 Attn: Lynn Barra  
 311 First Street, NW  
 Suite 500  
 Washington, DC 20001-2178

*P. 332*

# EXPLANATION OF COLUMN HEADINGS AND ENTRIES

52

## X3 PROJECT NUMBER

An arbitrary project identifier. Numbers 1 - 199 were assigned, upon initiation of the X3 Project Management System, to all then-active subcommittee Program of Work line items, under a "grandfather clause". Numbers above 200 were assigned to then-existing pre-standardization studies, and subsequently to all new proposal study projects and liaison projects as they are established. When more than one standard results from one project, separate project numbers are assigned as each new standard is identified.

## X3 PROJECT TYPE

- S** - **STUDY** project to determine the feasibility and need for a development project which has been proposed to X3 (see X3/SD-3). Study projects are managed by X3/SPARC.
- D** - **DEVELOPMENT** project, formally recommended by X3/SPARC and approved by X3, to produce an American National Standard.
- DT** - **DEVELOPMENT** project to produce an X3 Technical Report.
- R** - **REVISION** project, to revise an existing approved American National Standard.
- RF** - **REAFFIRMATION** project, as a result of the ANSI-required five year review when the X3 Technical Committee recommends that an existing American National Standard be reaffirmed without change.
- M** - **MAINTENANCE** project, the status into which a Development project is automatically placed upon approval of an American National Standard by ANSI. Activity for this type of project includes responses to inquiries for clarification and any comments received on experience with its use. As appropriate in individual cases, maintenance activity usually includes also the support by X3 toward adoption of its technical content as an International Standard.
- L** - **LIAISON** project, formal recognition of relations with another standards body on a project for which X3 has no existing standard or work in process. A Liaison project is automatically established for each project established by ISO/TC97, and for others when requested by an X3 Technical Committee and approved by SPARC and X3. These projects as initially established are "passive"--for information receipt only. Upon request by the X3 Technical Committee and approval by SPARC and X3, they may become "active" liaison to permit technical contribution and participation. Upon approval by SPARC and X3, they may also become Development projects, to develop corresponding American National Standards.

- I** - **INTERNATIONAL DEVELOPMENT** project, with an approved New Work Item which X3 has committed to support, and which is intended to result in an International Standard.

## DOCUMENT NUMBER

- X3.n-19xx** = ANSI catalog number of an approved American National Standard. (\* = revision being developed)
- BSR X3.nn** - A Draft proposed American National Standard approved by the originating X3 Technical Committee, accepted by X3 for public comment and submitted to the ANSI Board of Standards Review for concurrent review. It is continued with this identifier until approved by ANSI, when the "BSR" is removed and the year of approval added. (\* = revision of a published standard)
- p-X3/TR** - Proposed X3 Technical Report, being reviewed for approval by X3.
- X3/TR-n** - Published X3 Technical Report.

## ESTIMATED COMPLETION

Anticipated year of approval by ANSI of a completed Standard, or publication of an X3 Technical Report.

## ISO/TC97 PROJECT

Related project within ISO/TC97--however, respective boundaries of the two projects may not exactly coincide. Absence of an entry in this column indicates that no related TC 97 project has been established--two or more entries indicate that TC 97 has divided the work. Within the Project Number, the middle digit(s) identifies the responsible TC 97 Subcommittee, the last digit(s) identifies the project.

## ISO/TC97 DOCUMENT

The ISO documents listed are related to the X3 project in technical subject, but may agree or differ in technical content.

- R** - Published ISO/Recommendation. (\*=revision being developed)
- DP** - Draft Proposal being developed within a subcommittee of ISO/TC97.
- DIS** - New Draft International Standard, approved by a subcommittee and being considered by ISO/TC97. (\* = proposed revision of published Recommendation of the same number)
- DRS** - Draft revision of published International Standard of the same number.
- ISO** - Approved ISO International Standard. (\* = revision being developed)
- DTR** - Draft ISO Technical Report
- TR** - Approved ISO Technical Report

## X3 PROJECTS MANUAL (SD-4)

### FOREWORD

X3 administers its responsibilities for consideration and development of standards within its scope by means of a Project Management System.

- New X3 work is initiated by a Proposal, which, if sufficient interest is found, causes initiation of a STUDY project to determine the feasibility and need for standards on that subject.
- When the Study conclusions are affirmative, a project Recommendation is submitted to X3 letter ballot. If approved by at least 2/3 of the X3 membership, a DEVELOPMENT project is established to produce a standard.
- The project is converted to MAINTENANCE type when the proposed draft is approved by ANSI as an American National Standard.
- It is converted to REVISION type when a substantive change is proposed to and approved by X3, as a result of experience with and comments on the standard.
- The project is converted to REAFFIRMATION when, five years after publication the standard is reviewed and found to require no modification.
- LIAISON projects identify work of an industry, government, professional or international standards body, in which X3 has an interest but for which it has no directly related project.
- An INTERNATIONAL DEVELOPMENT project is one with an approved New Work Item which X3 has committed to support, and which is intended to result in an International Standard.

The Project Management System provides X3 the means used to identify, catalog, monitor and report its activities, and for filing its technical papers. A project may be terminated by X3 decision at any time prior to completion of a standard. However, once an American National Standard is published, the project remains, going through cyclic Maintenance, Revision and/or Reaffirmation stages as required until the standard is withdrawn.

X3/SD-4 provides a listing of the current X3 projects, arranged by technical discipline and cross-referenced to the related ISO/TC97 projects, proposals and approved standards.



<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<b><u>X3A1 - OCR &amp; MICR</u></b>					
0017-R	Print Specifications for Magnetic Ink Character Recognition Incorporates X3 Proj. 314	X3.2-1976	1986	97.03.07	ISO 1004-77	SPARC/82-477
0018-L	Bank Check Specifications for Magnetic Ink Character Recognition	X9.13-1983				
0057-M	Character Set for Optical Character Recognition (OCR-A)	X3.17-1981	1986	97.03.01	ISO 1073/I-76	
0061-M	Character Set for Optical Character Recognition (OCR-B)	X3.49-1982	1987	97.03.01	ISO 1073/II-76	
0062-M	Character Set for Handprinting	X3.45-1982	1987			
0069-M	Optical Character Recognition (OCR) Character Positioning	X3.93M-1981	1986	97.03.02	ISO 1831-80	
0227-M	Matrix Character Sets for Optical Character Recognition	X3.111-1986	1984			SPARC/854
0228-L	OCR Font for 7 and 9 Matrix Printers (Liaison with ECMA/TC4)	-				
0254-M	Paper Used In Optical Character Recognition (OCR) Systems	X3.62-1987	6/86			X3/85-31R
0274-M	Design of OCR Forms	X3/TR-5-1982	1987			
0284-M	Optical Character Recognition (OCR) Inks	X3.86-1987	1985			
0285-M	Guideline for Optical Character Recognition (OCR) Print Quality	X3.99-1983	1988		ISO 1831-80	
0312-DT	Basic Information on OCR	X3/TR-XX-	1984			SPARC/78-136
0313-S	Bar Code Standards	-				
0477-D	ANS for Guideline for Bar Code Print Quality	-	1986			SPARC/84-320
0611-D	Optical Character Recognition (OCR) Matrix Character Sets OCR-MB	-				X3/86-1477R
	<b><u>X3B5 - DIGITAL MAGNETIC TAPE</u></b>					
0038-M	Magnetic Tape Labels and File Structure for Information Interchange	X3.27-1987	1986	97.15.1	ISO 1001-79	X3/85-1204
070-M	Unrecorded Magnetic Tape for Information Interchange (9-Track 200 and 800 CPI, NRZI, and 1600 CPI, PE)	X3.40-1983	1988	97.11.04	ISO 1864-85	
0071-M	Recorded Magnetic Tape for Information Interchange (200 CPI, NRZI) (With X3L2 see Project 237-M)	X3.14-1983	1988	97.11.02	ISO 1862-75	
0072-M	Recorded Magnetic Tape for Information Interchange (800 CPI, NRZI)	X3.22-1983	1988	97.11.03	ISO 1863-76	
0073-M	Recorded Magnetic Tape for Information Interchange (1600 CPI, PE)	X3.39-1986	6/86	97.11.05	ISO 3788-76	
0213-M	Magnetic Tape Cassettes for Information Interchange (3.81 mm, 0.150 inch) Tape at 32 bpm (800 BPI), PE	X3.48-1986	3/86	97.11.06	ISO 3407-76	SPARC/82-1053
0217-M	Magnetic Tape Cassette Label	ISO 4341-1984	1989	97.15.3	ISO 4341-78	
0221-R	Unrecorded Magnetic Tape Cartridge for Information Interchange (0.250 Inch, 1600 BPI, Phase Encoded)	X3.55-1982	12/85	97.11.10	ISO 4057-79	X3/85-1219
0233-M	Recorded Magnetic Tape for Info. Interchange (6250 CPI, Group Encoded Recording)	X3.54-1986	6/86	97.11.11	ISO 5652-83	SPARC/573
0236-L	Recorded Magnetic Tape (7-Track, 200 CPI NRZI)	-		97.11.01	ISO 1861-75	
0250-M	One-Half Inch Magnetic Tape Interchange Using a Self-Loading Cartridge	X3.85-1987	1986	97.11.12	ISO 6098-82	SPARC/591
0255-M	Recorded Magnetic Tape Cartridge for Info. Interchange, 4-Track, 0.250 Inch 630mm, 1600 BPI, 63 BPM Phase Encoded	X3.56-1986	1985	97.11.10	ISO 4057-79	SPARC/82-421
0256-W	Magnetic Tape Cassette for Info. Interchange, Dual Track Complimentary Return-To-Bias, Four-States Recording	X3.59-1981	1986	97.11.08		

52

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. COMPL. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
	<b>X3B5 - DIGITAL MAGNETIC TAPE (CONTINUED)</b>					
0271-M	Parallel Rec. Mag. Tape Cartridge for Info. Interchange, 4-Track, 0.250 Inch, 6.30 mm, 1600 BPI, 63BPPM, Phase Encoded	X3.72-1987	1986			SPARC/831
0282-M	Unrecorded Magnetic Tape Minicassette For Information Interchange, Coplanar 3.81 mm (0.150 Inch)	X3.103-1983	1988			SPARC/754
0350-M	Recorded Magnetic Tape Cartridge, 1/4 Inch 6400 BPI, 4-Track	X3.116-1986	3/86	97.11.10	DIS 8063/2	SPARC/80-564
0366-M	Unrecorded 1/4 Inch Recorded Magnetic Tape Cartridge (6400-10000 BPI)	X3.127-1987	3/86	97.11.10	DIS 8063/1	SPARC/82-420
0391-D	Recorded Magnetic Tape for Information Interchange, 0.5 in (12.7 mm) Tape, Nine Track, 3200 CPI (126 CPMM)	X3.157-1987	12/85			SPARC/83-517
0403-D	Unrecorded MagTape Cartridge for Information Interchange 0.250 Inch (6.30 mm), 10000 ftpi. (394 ftppm)	-	6/86	97.11.10	ISO 4057	SPARC/83-671
0404-M	Serial Recorded Magnetic Tape Cartridge for Information Interchange, Four and Nine Track	X3.136-1986	3/86	97.11.10	DP 8462/2	SPARC/83-698
0405-D	Unrecorded Magnetic Tape Cassette for Info. Interchange 3.81 mm (0.150 In), 252 to 394 ftppm (6400 to 10000 ftpi)	X3.164-198x	12/85			SPARC/83-690
0406-M	Serial Recorded Magnetic Tape Cassette for Info. Interchange, 0.150 in (3.81mm) 8000 bpi (315 bppm) Group Code Recording	X3.158-1987	3/86			SPARC/83-689
0472-M	Recorded Magnetic Tape Minicassette for Information Interchange, Coplanar 3.81 mm (0.150 Inch) Phase Encoded	X3.104-1983	1988			SPARC/754
0485-D	Unrecorded Magnetic Tape Cartridge for Info. Interchange 0.500 In (12.65 mm) 6000-15000 ftpi (236-590 ftppm)	-	1986			X3/84-718
0486-D	Recorded Magnetic Tape Cartridge for Info. Interchange, 0.500 In (12.65 mm), 6000-15000 bpi (236-590 bppm)	-	1986			X3/84-715
0487-D	Unrecorded Mag. Tape and Ctrtdg for Info Intchnng., 18-Track, Parallel, 12.65mm (1/2 in), 1491 cpmm (37 871 cpi)	-	3/87			X3/84-720
0488-D	Rec. Mag. Tape & Cartr. for Info. Intchg 18-Track, Parallel, 12.65mm (1/2,in), 1491 cpmm (37 871 cpi) Grp-Coded Recrdng	-	3/87			X3/84-721
0553-D	ANS for Unrecorded Magnetic Tape Mini Cartridge for Info. Interchange, 0.25 In (6.30 mm), 12500 ftpi (492 ftppm)	-	12/87			X3/85-1420
0561-D	Unrecorded Magnetic Tape and Cartridge System for Mechanical and Magnetic Interchangeability between Info. Systems	-	7/87			X3/85-1547
0562-D	Recorded Magnetic Tape Cartridge for Information Interchange 0.500 in (12.65 mm), 22-Track, Serial, 6667 bpi	-	7/87			X3/85-1548
0563-D	Unrecorded Magnetic Tape Cartridge for Information Interchange 0.500 inch (12.65 mm), 6667 ftpi (262 ftppm)	-	7/87			X3/85-1549
0565-D	Recorded Magnetic Tape Mini-Cartridge for Info. Interchange, 0.250 IN (630 mm) 12 and 24 Track, 10000 bpi (394 bppm)GCR	-	12/87			X3/85-1419
0566-W	Recorded Magnetic Tape Cartridge for Info. Interchange, 0.500 IN (12.65 mm) 24-Track, Serial 8000 bpi (315 bppm) GCR	-	6/87			X3/85-1544
0567-W	Unrecorded Magnetic Tape Cartridge for Info. Interchange, 0.500 IN (12.65 mm) 10000 ftpi (393 ftppm)	-	6/87			X3/85-1545
0568-D	Recorded Magnetic Tape Cartridge for Information Interchange, 20-Track Serial 1/2 in (12.65mm) 12 000 ftpi (472 ftppm)	-	6/87			X3/85-1546
0645-D	Unrecd. Magnetic Cart. for Information Interchnng. 0.150 in 10 000 ftpi Coercivity 500 Oersteds (44 000 meters)	-				X3/87-09-030

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>SUBSER</u>
						52
	<b>X385 - DIGITAL MAGNETIC TAPE (CONTINUED)</b>					
0646-D	Recd. Magnetic Tape Mini-Cartridge for Info. Interchg. 12-Tk. 0.150 in, 20-Tk. 0.250 in (6.30) 10 000 ftpi, MFH	-				X3/87-09-031
0647-D	Recd. Magnetic Tape Cartridge & for Info. Interchg. 1/2in, Ser. Serpentine, 22-Tk., 6667 ftpi & 48-Tk., 10 000 ftpi	-				X3/87-09-032
0649-D	Unrecd. Magnetic Tape Cartridge & for Info. Interchg. 1/2in, Ser. Serpentine, 22-Tk. 6 667 ftpi & 48-Tk. 10 000 ftpi	-				X3/87-09-033
	<b>X386 - INSTRUMENTATION TAPE</b>					
0230-W	Instrumentation Magnetic Tape	-	N/A 1988	97.12 97.12.9	ISO 3802-81 DP 8441	SPARC/77-30
0291-D	High-Density Digital Magnetic Recording (HDDR)	-	1987	97.12.7	IS 6068	SPARC/82-637
0371-D	Recording Characteristics of Instrumentation Magnetic Tape	-	1986	97.12.8	DIS 6371	SPARC/83-45, 83-361
0383-D	Characteristics of Unrecorded Instrumentation Magnetic Tape -- Interchange Practices and Recommended Test Methods	-	1986	97.12.1	DIS 1860	SPARC/83-488
0390-D	Precision Reels for Magnetic Tape Used in Interchange Instrumentation Applications	ISO 1860-198x	1986	97.12.1	DIS 1860	SPARC/83-488
0641-D	Standards Commands & Mnemonics to Be Used with IEEE-488, RS-232 & Other Remote Interfaces Instrumentation	-	1990			X3/86-1895R
0592-D	Digital Recording Based on the SMPTE D-1 Format	-				X3/86-1696
	<b>X387 - MAGNETIC DISKS</b>					
0064-M	Unrecorded Magnetic Six-Disk Pack (General, Physical, and Magnetic Characteristics)	X3.46-1983	1988	97.10.1.1	ISO 2864-74	
0065-M	Unrecorded Magnetic Eleven-Disk Pack, General Physical and Magnetic Characteristics	X3.58-1984	1989	97.10.3.1	ISO 3564-76	
0123-L	Track Format for Six Disk Pack	-		97.10.1.2	ISO 3561-76	
0124-L	Track Format for Single Disk Cartridge (Top Loaded)	-		97.10.2.3	ISO 3563-76	
0224-M	Unrecorded Single Disk Cartridge (Front Loading, 2200 BPI)	X3.52-1987	1989			
0225-M	Interchangeable Magnetic Twelve-Disk Pack (100 Megabytes)	X3.63-1981	1986	97.10.4	IOS 4337-77	
0251-M	Interchangeable Magnetic Twelve-Disk Pack (200 Megabytes)	X3.84-1981	1986	97.10.6	DIS 5653	SPARC/598
0275-M	Unrecorded Single-Disk, Double-Density Cartridge (Front Loading 2200 BPI, 200 TPI)	X3.89-1987	1986			SPARC/750-A
0277-RF	Unformatted Single-Disk Cartridge (Top Loading, 200 TPI, 4400 BPI)	X3.76-198x	1986			SPARC/750-C
0321-M	Physical, Mechanical & Magnetic Characteristics of an Unformatted 80 Megabyte Trident Pack for Use at 370 TPI, 6000BPI	X3.115-1984	1989			SPARC/78-137
0328-M	14-Inch (356 mm) Diameter Low Surface Friction Magnetic Storage Disk	X3.112-1984	1989	97.10.7	IS 6901	SPARC/79-55R
0343-M	Contact Start/Stop Storage Disk, 158361 Flux Transitions Per Track, 8.268 Inch Outer and 3.937 Inch Inner Diameters	X3.119-1984	1989		DIS 7298	SPARC/80-234 REV.
0344-M	Contact Start/Stop Storage Disk, 95840 Flux Transitions Per Track, 7.874 Inch Outer & 2.50 Inch Inner Diameters	X3.120-1984	1989		ISO 7297	SPARC/80-235R 10/10/80
0353-M	Contact Start/Stop Storage Disk: 83000 Flux Transition Per Track 130 mm (5.118 In) Outer Diam. 40 mm (1.575 In) Inner	X3.128-1985	1991		DP 7929	SPARC/81-52R REV. 3/18/81
0356-D	A Contact Start/Stop Metallic Thin Film Storage Disk, 83,333 Flux Transition Per Track, 130MM Outer Dia. & 40MM Inner Dia	X3.163-198x	1986			X3/85-607

52

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. Cmpl. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
<b><u>X3B7 - MAGNETIC DISKS</u></b>						
0360-M	5 1/4 Inch Rigid Disk Removable Cartridge	X3.155-1987	1986	97.10.11	DP 8679	SPARC/81-893R 1/16/82
0369-D	Nominal 8 inch Rigid Media Removable Cartridge	X3.156-1987	1986	97.10.12	DP 8680	SPARC/81-285R 5/14/81
0479-D	95 MM Rigid Digital Recording Disk	-	1987			SPARC/84-527
0489-L	Liaison with the Semiconductor Equipment Manufacturers Institute (SEMI) for the Development of Substrate Standards	-	N/A			X3/84-916
0492-D	100 mm Rigid Digital Recording Disc for Cartridge Applications	-	1987			SPARC/84-471
<b><u>X3B7.1 - TEST METHODS AND PROCEDURES</u></b>						
0582-S	Study Project on Test Methods and Procedures	-				X3/86-1027
<b><u>X3B8 - FLEXIBLE DISK CARTRIDGES (FDC)</u></b>						
0231-M	Single-Sided Unformatted Flexible Disk Cartridge for 6631 BPR Use	X3.73-1986	1986	97.11.9.1	ISO 5654/1-84	SPARC/541
0232-D	Flexible Disks - Recorded Characteristics	-	1986	97.11.9	ISO 5654/2-82	SPARC/818
0272-D	Flexible Disk Labels and File Structures	ISO 7665-	1987	97.15.8	ISO 7665	SPARC/80-818
0286-M	Two-Sided Unformatted 8-Inch (200 MM) Double Density Flexible Disk Cartridge (For 13262 FIPR Two-headed Application)	X3.121-1984	1989	97.11.9.3.1	ISO 7065/1-2	SPARC/77-28
0287-M	Two Sided Unformatted (200 mm) Double Density Flexible Disk Cartridge, General Physical and Magnetic Requirements	X3.121-1984	1989			SPARC/77-20
0306-R	One-Sided Single-Density Unformatted 5.25 Inch Flexible Disk Cartridge	X3.82-1980	1986	97.11.9.4.1	ISO 6596/1-2	SPARC/77-27
0322-M	Two-Sided, Double-Density, Unformatted 5.25 Inch (130 mm) 48-tpi (1.9 tpsmm) Flexible Disk Cartridge for 7958 BPR Use	X3.125-1984	1989	97.11.9.5.1	ISO 7487/1	SPARC/79-77
0354-M	One or Two-Sided Double Density Unformatted 5.25 Inch (130 mm) 96 Tracks Per Inch Flexible Disk Cartridge	X3.126-1986	1990	97.11.15		SPARC/81-313R 8/3/81
0373-D	3.5 Inch Flexible Disk Cartridge	X3.137-198x	1986			SPARC/83-423
0453-W	Unformatted 72 mm (3 Inch Nominal) Two-Sided Double Density Flexible Disk Cartridge for Information Interchange	X3.142-198X	1985			SPARC/83-731
0475-I	MMI TC97 N1368, Std. of Flexible Disk Cartridges for Data Interchange Having a Diameter Smaller than 100 MM	-		97.11.15	TC97 N1368	SPARC/84-429
0494-D	5.25 Inch High Density (130 mm) Flexible Disk Cartridge	X3.162-198x	1986		DP 8630/1, /2	X3/84-1065R
0588-D	Data Intchnng on 90mm (3.5 in) Flex Disk Cartdg Using Modified Freq Modulation Recording at 15916 ftprad, on 80 Tracks	-				X3/86-960
0589-D	90 mm (3.5 in) Flexible Disk Cartridge at 15916 ftprad	-				X3/86-1601
<b><u>X3B8.1 - TRACK FORMATS FOR FLEXIBLE DISK CARTRIDGES</u></b>						
0386-L	Liaison with all TC97 Development Projects for Recorded Characteristics of Flexible Disk Cartridges	-	N/A			SPARC/83-310
0389-D	ANS for Flexible Disk Track Form for Information Interchange	-	1986		DP 8630/2	SPARC/83-311
<b><u>X3B9 - PAPER FORMS/LAYOUTS</u></b>						
0101-R	Specification for General Purpose Paper Cards for Information Interchange	X3.11-1969	1985			
0102-M	Rectangular Holes in Twelve-Row Punched Cards	X3.21-1980	1985	97.0.2	ISO 1682-73	
0437-M	Basic Sheet Sizes and Standard Stock Sizes for Bond Paper and Index Bristols (FORMERLY X4.4-1972)	X3.151-1987	1985			N/A— Grandfather Clause
0438-M	Specifications for Single-Fly, Non-carbonized Adding Machine Paper Rolls (FORMERLY X4.8-1973)	X3.152-1987	1985			

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<b><u>X3B9 - PAPER FORMS/LAYOUTS (CONTINUED)</u></b>					
0439-D	Standard for Business Letterhead Sizes	-	1987			
0440-M	Conversion of paper Substance Weights from Ream Weights to g/m2	X3/TR-2-1982	1986			
0441-M	Paper Sizes for Single Part Continuous Business Forms	X3.96-1983	1988			
0442-D	Paper Roll Sizes	-	1988			
0443-M	Printable/Image Areas for Text and Facsimile Communication Equipment	X3.117-1984	1989			
0444-D	Quality Requirements for Paper for Continuous Forms	-	1987		DP 7552	
0445-DT	Forms Tutorial	-	1985			
0446-DT	Publications on the Advantages of North American Sizes vs. ISO A Sizes	-	1985			
0447-M	Office Machines and Business Forms Character and Line Spacing (FORMERLY X4.17-1976)	X3.150-1987	1985			N/A— Grandfather Clause
	<b><u>X3B10 - CREDIT/ID CARDS</u></b>					
0430-M	Credit Card Specifications	X4.13-1983	1988		ISO 7810	
0431-M	Magnetic Stripe Encoding on Credit Cards	X4.16-1983	1988		ISO 7811/1-5	
0432-M	Addendum to X4.16-1976, Encoding for Track 3; Liaison with X9	X4.16A-1977	1984		ISO 7812, 7813	
0433-L	Liaison with X9 - Magnetic Encoding Track 3	-				
0434-M	Interindustry Message Specifications for Credit Cards	X4.21-1981	1986		DIS 7580	
0436-M	PIN Pad Specifications	X3.118-1984	1989			
0471-I	DIS 7501, Machine Readable Passports	-		97.17.9	DIS 7501	N/A
0586-D	Identification Cards - Physical Characteristics (REF. ISO 7810)	ISO 7810-198x			ISO 7810	X3/86-1778
0590-D	Identification Cards - Recording Techniques - Part 1: Embossing	ISO 7811/1-198x			ISO 7811/1	X3/86-1778
0595-D	Identification Cards - Numbering System and Registration Procedure for Issuer Identifiers	ISO 7812-198x			ISO 7812	X3/86-1778
0609-D	Identification Cards - Financial Transaction Cards	ISO 7813-198x			ISO 7813	X3/86-1779
0633-D	Identification Cards - Recording Techniques - Part 2: Magnetic Stripe	ISO 7811/2-198x			ISO 7811/2	X3/86-1778
0634-D	Identification Cards - Recording Techniques - Part 3: Location of Embossed Characters on ID Cards	ISO 7811/3-198x			ISO 7811/3	X3/86-1778
0635-D	Identification Cards - Recording Techniques - Part 4: Location of Read-Only Magnetic Tracks - Tracks 1 & 2	ISO 7811/4-198x			ISO 7811/4	X3/86-1778
0636-D	Identification Cards - Recording Techniques - Part 5: Location of Read-Write Magnetic Track - Track 3	ISO 7811/5-198x			ISO 7811/5	X3/86-1778
	<b><u>X3B10.1 - INTEGRATED CIRCUIT CARDS</u></b>					
0372-D	Integrated Circuit Cards	-	1986	97.17.8	DIS 7816/1-3	X3/85-009R
	<b><u>X3B10.2 - REVISION OF X3.149-1986</u></b>					
0402-M	Location of Imprinted Information on a Credit Card Charge Form (FORMERLY X4.18-1977 PRIOR TO REVISION)	X3.149-1986	1991			SPARC/83-639
	<b><u>X3B10.3 - MINIMUM PHYSICAL REQUIREMENTS OF SAVINGSBOOKS</u></b>					
0459-D	Minimum Physical Requirements of Savingsbooks	-	1985		ISO 8484	SPARC/83-696 Rev. 1/19/84
	<b><u>X3B10.4 - OPTICALLY ENCODED CARD MEDIA</u></b>					
0511-D	ANS for Optically Encoded Card Media	-	1986			X3/84-1403

52

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. CHPL. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
0407-D	<u>X3B11 - OPTICAL DIGITAL DATA DISKS</u> Unrecorded Optical Media Unit for Digital Information Interchange, Nominal 200 mm (8.00 Inch) Diameter	-	1/86	97.23.3	NWI TC97 N1421	SPARC/83-536R
0408-D	Unrecorded Optical Media Unit for Digital Information Interchange, Nominal 300 mm (12.00 Inch) Diameter	-	1/86	97.23.2	NWI TC97 N1420	SPARC/83-537R
0409-D	Unrecorded Optical Media Unit for Digital Information Interchange, Nominal 120 mm (4.72 Inch) Diameter	-	1/86			SPARC/83-540R
0456-D	Unrecorded Optical Media Unit for Digital Information Interchange, Nominal 356 mm (14.00 Inch) Diameter	-	1/86	97.23.1	NWI TC97 N1419	SPARC/83-538R
0457-D	Unrecorded Optical Media Unit for Digital Information Interchange, Nominal 130 mm (5.25 Inch) Diameter	-	1/86	97.23.4	NWI TC97 N1422	SPARC/83-539R
0480-D	Recorded Characteristics of Optical Media Units for Digital Info. Interchange, Nominal 120 mm (4.72 Inch)	-	1986			SPARC/84-600
0481-D	Recorded Characteristics of Optical Media Units for Digital Info. Interchange, Nominal 130 mm (5.25 Inch)	-	1986	97.23.4	NWI TC97 N1422	SPARC/84-601
0482-D	Recorded Characteristics of Optical Media Units for Digital Info. Interchange, Nominal 200 mm (8 Inch)	-	1986	97.23.3	NWI TC97 N1421	SPARC/84-602
0483-D	Recorded Characteristics of Optical Media Units for Digital Info. Interchange, Nominal 300 mm (12 Inch)	-	1986	97.23.2	NWI TC97 N1420	SPARC/84-603
0484-D	Recorded Characteristics of Optical Media Units for Digital Info. Interchange, Nominal 356 mm (14 Inch)	-	1986	97.23.1	NWI TC97 N1419	SPARC/84-604
0524-D	File Structure and Labelling of Optical Digital Data Disks for Information Interchange	-	1988			X3/85-356
0581-D	Unrecorded Optical Media Unit for Digital Information Interchange - Nominal 90 mm (3.5 inch) Diameter	-198X	1988			X3/86-15c
0607-I	Unrecorded Reversible Optical Media Unit for Digital Information Interchange Nominal 130 mm (5.25 in) Diameter	-				X3/86-1588
<u>X3B2 - DATA BASE</u>						
0355-M	Database Language NDL	X3.133-1986	1986	97.21.3.1	SC21 N174	SPARC/81-191R
0363-M	Database Language SQL	X3.135-1986	1986	97.21.3.2	SC21 N173	SPARC/81-689R 3/4/82
0525-D	ANS for Extended Database Language SQL	-	2/86			X3/85-657R
0571-D	Embedding of SQL Statements into Programming Languages	-	2/86			X3/85-658R
0583-L	Remote Data Access (RDA) Service and Protocol	-				X3/86-1760
0594-D	Database Language SQL/Addendum 1 (Integrity Enhancement Feature)	X3.135.1-198x				X3/86-1761
0630-D	Embedding of NDL Statements into Programming Language	-				X3/86-1560
<u>X3B3 - COMPUTER GRAPHICS</u>						
0596-I	TC97 NWI N 1620, Technical Report for Info. Processing Sys - Computer Graphics - Conformity Texting of Graphics Stds.	-			ISO 1620	
0597-I	TC97 NWI N 1619, Computer Graphics - Feasibility of Formally Specifying Graphic Standards	-			ISO 1619	
0612-D	Pascal Language Binding of the Programmer's Hierarchical Interactive Graphics System (PHIGS)	-				X3/87-02-068

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CMPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
0460-D	<u>X3H3.1 - PROGRAMMERS'S HIERARCHICAL INTERACTIVE GRAPHICS SYSTEM (PHIGS)</u> Programmer's Hierarchical Interactive Graphics System (PHIGS)	X3.144-198x	1987	97.21.24	SC21 N819	SPARC/83-832 REV. 1/18/84
0346-D 0347-M 0620-I	<u>X3H3.3 - VIRTUAL DEVICE INTERFACE</u> Computer Graphics Interface (CGI) Computer Graphics Metafile (CGM) (Formerly VDM) TC97 NWI N1793, Information Processing Systems - Computer Graphics - Reference Model of Computer Graphics	X3.161-198x X3.122-1986 -	1987 1986	97.21.26 97.21.5 97.21.42	SC21/2 N1179 DIS 8632 TC97 N1793	SPARC/80-420R SPARC/80-557
0529-D 0530-D 0531-D 0532-D 0533-D 0534-D 0535-M 0543-D 0544-I 0545-I 0546-I 0559-D 0560-D	<u>X3H3.4 - LANGUAGE BINDING</u> ANS for Ada Language Binding of the Graphical Kernel System (GKS) ANS for the *Ada Language Binding of the Programmers Hierarchical Interactive Graphics Standard (PHIGS) ANS for the Pascal Language Binding of the Graphical Kernel System (GKS) ANS for the Fortran Language Binding of the Programmers Hierarchical Interactive Graphics Standard (PHIGS) C Language Binding of the Graphical Kernel System (GKS) C Language Binding of the Programmers Hierarchical Interactive Graphics System (PHIGS) Graphical Kernel System (GKS) FORTRAN Binding ANS for the Ada* Language Binding of the 3-D Extensions to GKS (*Ada is a Registered trademark of the U.S. Government) Fortran Language Binding of the 3-D Extensions to GKS Pascal Language Binding of the 3-D Extensions to GKS C Language Binding of the 3-D Extensions to GKS C Language Binding of the Computer Graphics Interface Fortran Language Binding of the Computer Graphics Interface	X3.124.3-198X - X3.124.2-198x X3.144.1-198x X3.124.4-198x - X3.124.1-1985 - - - - -	1987 1987 1987 1987 1987 1988 1990 1988 1987 1988 1988 1988 1988	97.21.7.3 97.21.27.3 97.21.7.2 97.21.27.1 97.21.7.7 97.21.27.4 97.21.7.1 97.21.7.6 97.21.7.4 97.21.7.5 97.21.7.8 97.21.26.2 97.21.26.1	DP 8651/3 SC21 N668 DIS 8651/2 SC21 N667 SC21 N669 DP 8651/1 DP 8806/1	X3/85-1166R X3/85-1168R X3/85-1169R X3/85-1172R X3/85-1173R X3/85-1175R X3/85-1167R X3/85-1171R X3/85-1170R X3/85-1174R X3/85-1559 X3/85-1560
0268-M 0547-I	<u>X3H3.5 - GRAPHICAL KERNEL SYSTEM (GKS)</u> Information Processing Systems - Computer Graphics - Graphical Kernel System Three-Dimensional Extensions to GKS (Graphical Kernel System)	X3.124-1985 -	1990	97.21.2 97.21.5.2	DIS 7942 DIS 8805	SPARC/79-52R, 81-930 X3/85-963R
0552-D	<u>X3H3.6 - WINDOW MANAGEMENT</u> Window Management	-	1989			X3/85-1531
0336-D	<u>X3H4 - INFORMATION RESOURCE &amp; DICTIONARY</u> Information Resource Dictionary System (IRDS)	X3.138-198x	1986	97.21.06	NWI TC97 1243	SPARC/80-147
0570-DT	<u>X3H4.1 - IRDS REFERENCE MODEL</u> Technical Report, Reference Model for Information Resource Dictionary System (IRDS)	-	1987			X3/85-1110R

52

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. Cmpl. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
0557-DT	<u>X3M4.2 - IRDS EXTERNAL SOFTWARE INTERFACE</u> Technical Report on Integration of External Software Environments with the IRDS	-	1987			X3/85-1113R
0569-D	Information Resource Dictionary System (IRDS) Software Interface	-	1988			X3/85-1115R
0212-M 0297-MT	<u>X3J1 - PL/I</u> Programming Language PL/I Technical Report for Real Time Subset of Full PL/I, X3.53-1976	X3.53-1987 X3/TR-7-1985	1986 1990	97.22.5	ISO 6160	SPARC/841 X3/84-961
0296-R	<u>X3J1.3 - GENERAL PURPOSE SUBSET</u> PL/I General Purpose Subset	X3.74-198x	1986	97.22.6	ISO 6522	SPARC/82-810
0215-RF 0352-M 0584-D	<u>X3J2 - BASIC</u> Programming Language Minimal BASIC Programming Language Full BASIC Addendum to Programming Language Full BASIC, Modules and Individual Character Input	X3.60-1978 X3.113-1987 -	1985		DP 6373	SPARC/509 SPARC/81-51R X3/86-1546
0067-R	<u>X3J3 - FORTRAN</u> Programming Language FORTRAN	X3.9-198x	1987	97.05.02	ISO 1539-80	SPARC/78-33
0021-D 0022-M 0585-D 0640-I	<u>X3J4 - COBOL</u> COBOL Information Bulletins (CIB NO. 23) Programming Language COBOL Addendum to ANSI X3.23-1985, Programming Language COBOL Correction Addendum ISO 1985 for Programming Language COBOL	- X3.23-1985 X3.23A-198x -	1990	97.05.01	ISO 1989-78	SPARC/81-257 X3/86-413
0055-M 0315-MT 0316-MT 0361-DT	<u>X3J7 - APT</u> Programming Language APT APT Language - Postprocessor Interface Modules APT Language - Expository Remarks Concerning X3.37-1980 Tutorial for X3.37, Revision 3 of Programming Language APT	X3.37-1987 X3/TR-4-1982 X3/TR-3-1982 X3/TR-X-198x	1986 1987 1987 1986	97.09.01-09	ISO 3592-78	SPARC/84-   SPARC/84-480
	<u>X3J7.1 - PROCESSOR LANGUAGES</u> <u>X3J7.2 - POSTPROCESSOR LANGUAGES</u> <u>X3J7.3 - LATEX LANGUAGE</u> <u>X3J7.4 - ROBOTICS LANGUAGE</u>					
0317-M	<u>X3J9 - PASCAL</u> Programming Language PASCAL (Note: Complete Designation is ANSI/IEEE770X3.97-1983)	X3.97-1983	1988	97.05.10	DIS 7185	SPARC/79-111
0345-D	<u>X3J9.1 - PASCAL EXTENSIONS</u> Extended Programming Language PASCAL	X3.160-198x	1986			X3/85-1900
0331-D 0577-D	<u>X3J10 - APL</u> Programming Language APL ANS for Advancements in the APL Language	X3.123-198X -	1986 1991	97.05.11	DP 8485	SPARC/79-349 X3/86-527
0381-D	<u>X3J11- PROGRAMMING LANGUAGE C</u> Programming Language C	X3.159-198x	1985			SPARC/83-79R 3/2/83
0507-D	<u>X3J12 - DIBOL</u> Programming Language DIBOL	X3.165-198x	1987			X3/84-994R

P. 343



<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>COMPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>ISO-3 REF.</u> <u>NUMBER</u>
0574-D	<u>X3J13 - COMMON LISP</u> COMMON LISP	-	1/88			86-344
0610-D	<u>X3J14 - FORTH</u> Programming Language Forth	-	1988			X3/86-1777
0016-M	<u>X3K1 - COMPUTER DOCUMENTATION</u> Guide for Technical Documentation of Computer Projects	X3/TR-6-1982	1987			
0264-L	Computer Configuration Charts	-		97.5.5		
0266-L	Specification of Single-Hit Decision Tables	-		97.7.7	DIS 5806	
0299-L	<u>X3K1 - COMPUTER DOCUMENTATION (CONTINUED)</u> Symbols & Conventions for Program Flow, Program Networking, Data Flow & Computer Configuration	-		97.7.8	DIS 5807	
0506-L	Guidelines for the Documentation of Computer-Based Systems	-		97.7.3	DP 6592	N/A
0516-D	Documentation Standard for Small Computer Applications	-	1987	97.07.03	SC7 N337	X3/85-112
0517-D	Logical Flow of Activities in the Life of an Automated System	-	1986			X3/85-111
0026-D	<u>X3K5 - VOCABULARY FOR INFORMATION PROCESSING SYSTEMS</u> American National Dictionary for Information Processing Systems (ANDIPS)	X3/TR-1-1982	3/88			SPARC/84-581
0027-L	ISO Vocabulary of Data Processing	-		97.1.1-.20	ISO 2382/1-XVI	
	Section 01: Fundamental Terms			97.1.1	ISO 2382/1-84	
	Section 02: Mathematics & Logic. Arith. & Logic Oper.			97.1.2	ISO 2382/2-76	
	Section 03: Equipment Technology			97.1.3	ISO 2382/3-76	
	Section 04: Organization of Data			97.1.4	ISO 2382/4-74	
	Section 05: Representation of Data			97.1.5	ISO 2382/5-74	
	Section 06: Preparation of Handling Data			97.1.6	ISO 2382/6-74	
	Section 07: Digital Computer Programs			97.1.7	ISO 2382/7-	
	Section 08: Control, Integrity, and Security			97.1.8	ISO 2382/8-	
	Section 09: Data Communication			97.1.9	ISO 2382/9-79	
	Section 10: Operating Techniques & Facilities			97.1.10	ISO 2382/10-76	
	Section 11: Control, Input-Output & Arithmetic Equipment			97.1.11	ISO 2382/11-76	
	Section 12: Storage Techniques & Data Media			97.1.12	ISO 2382/12-79	
	Section 13: Computer Graphics and Micrographics			97.1.13	ISO 2382/13-84	
	Section 14: Reliability, Maintainability & Availability			97.1.14	DIS 2382/14-	
	Section 15: Programming Languages			97.1.15	DIS 2382/15-	
	Section 16: Information Theory			97.1.16	ISO 2382/16-78	
	Section 17: Data Bases			97.1.17		
	Section 18: Open Systems & Distributed Data Networks			97.1.18	DP 2382/18	
	Section 19: Analogue Computing			97.1.19	ISO 2382/19-80	
	Section 20: Systems Development			97.1.20	DP 2382/20	
	Section 21: Interface Terminology			97.1.21	DP 2382/21	
0398-L	Liaison with ISO TC97 SC1, DIS 5138, ISO Office Machines Vocabulary	-			ISO 5138	
	Section 01: Dictation Equipment				ISO 5138/1	
	Section 02: Duplicators				ISO 5138/2	
	Section 03: Addressing Machines				DIS 5138/3	
	Section 04: Letter Opening Machines				ISO 5138/4	
	Section 05: Letter Folding Machines				ISO 5138/5	
	Section 06: Calculators				DIS 5138/6	
	Section 07: Postal Franking Machines				DIS 5138/7	
	Section 08: Document Copying Machines				DIS 5138/8	
	Section 09: Typewriters				ISO 5138/9-84	
	Section 10: Word Processing Equipment				DP 5138/10	
	Section 11: Document Inserting Machines				DIS 5138/11	
0448-D	Glossary of Word Processing, Definition of Terms and Functions	-				

52

52

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<u>X3L2 - CODES AND CHARACTER SETS</u>					
0006-M	Graphic Representation of the Control Characters of the American National Code for Information Interchange	X3.32-1973	1986	97.02.01	ISO 2047-81	
0007-L	Rules for the Definition of 4-Bit Subsets (TC97/SC2 and ECMA 21)	-		97.02.10	ISO 963-73	
0012-M	Information Processing—Coded Character Sets—7-Bit American National Standard Code for Info. Interchange (7-bit ASCII)	X3.4-1986	1986	97.02.01	ISO 646-83	SPARC/83-542
0013-RF	USA Sponsorship Procedures for ISO Registration According to ISO 2375	X3.83-198x	1985	97.02.04	ISO 2375-80	
0103-RF	Hollerith Punched Card Code	X3.26-198x	1985	97.02.15	ISO 6586-80	
0105-R	Code Extension Techniques for Use with the 7-Bit Coded Character Set for the ANS Code for Information Interchange	X3.41-1974	1986	97.02.02	ISO 2022-82	SPARC/83-393
0106-D	USA Candidates for Registry	-	N/A 1985	97.02.12	ISO 1113-79	
0107-M	Perforated Tape Code for Information Interchange	X3.6-1973				
0216-R	Magnetic Tape Cassette Code (For X3B5; REF. X3.48-1977, PROJ. 213-R)	-	3/86	97.02.04	ISO 3275-74	SPARC/82-1053
0237-M	Recorded Magnetic Tape for Information Interchange (200 CPI, NRZI) (Liaison with X3B5 for Coding; Ref. Proj. 71)	X3.14-1983	1988	97.02.05	ISO 962-74	
0239-L	Transformation of Data Between Telex Code and 7-Bit Code	-	N/A	97.02.07	ISO 6936-83	
0240-L	Coding of Character Sets for OCR & MICR	-	N/A	97.02.09	ISO 2033-83	
0257-M	Codes for Magnetic Tape Cartridge (0.250 Inch) (For X3B5; Ref. Proj. 255)	X3.56-1986	1985	97.11.10	ISO 4057-79	SPARC/82-421
0294-L	Codes for Flexible Disks (For X3B8/X3L5)	-	N/A	97.11.09	DIS 6863	
0304-M	Hexadecimal Input/Output to Microprocessors Using 5-Bit and 7-Bit Teleprinters	X3.95-1982	1987		ISO 6936-82	SPARC/77-93
0349-M	Coded Character Set for Use with X3.96-1983 Text Information Interchange In Page Image Format (Ref. Proj. 450)	-	1988	97.02.14	ISO	SPARC/80-663 6937/1-2-1
0351-M	Coded Character Sets for Use with X4.23 and X4.22 Keyboard Arrangements for Alphanumeric Machines	X3.114-1984	1989			SPARC/80-
0387-D	Control Function Coding for X3V1 Basic Processable Text Interchange Format and Text Processing Functions	-	N/A	97.02.13		SPARC/83-548
0388-D	Additional Graphic Character Sets for Use with ASCII	-	1986	97.02.14	ISO 6937	SPARC/83-546
0392-D	7-Bit and 8-Bit ASCII Supplemental Multilingual Graphic Character Set (ASCII Multilingual Set)	X3.134.2-198X	1985	97.02.20	NWI TC97 N1415	SPARC/83-547
0397-D	X3L2 Project for X3B3 Computer Graphics Metafile (CGM) and Computer Graphics Interface (CGI) Functions	-	1986	97.02.17	TC97 N1418	SPARC/83-549
0466-I	TC97 NWI N1285, 7-Bit Coded Character Set for the Arabic Language	-	1986	97.02.19	TC97 N1285	SPARC/84-385
0495-D	8-Bit ASCII - Structure and Rules	X3.134.1-198X	1986		TC97 N1498	X3/84-1080
0509-I	NWI TC97 N1416, Coding of Audio Information, Particularly Synthesized Sound, as Part of Interchangeable Documents	-		97.02.21	TC97 N1416	X3/84-926
0514-D	ANS for Alternate Controls for Character Imaging Devices	-	1985			X3/84-1691
0536-S	Identifying Videotex Requirements to be Met by Message Handling Systems Under Dev. in TC97/SC18 & X3V1 (MOTIS) & CCITT	-	12/86			X3/85-1207
0537-S	Draft Joint X3L2/X3V1 Project for Identifying Data Link Protocol Requirements for Videotex	-	12/86			X3/85-1208
0538-S	Joint X3L2/X3V1 Project for Identifying Session Layer Protocol Requirements for Videotex	-	12/86			X3/85-1209

P. 344

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<b><u>X3L2 - CODES AND CHARACTER SETS (CONTINUED)</u></b>					
0539-S	Joint X3L2/X3V1 Project for Identifying Presentation Layer Protocol Requirements for Videotex	-	4/86			X3/85-1210
0540-S	Joint X3L2/X3V1 Project for Identifying Common Application Layer Protocol Requirements for Videotex	-	12/86			X3/85-1211
0554-D	Extending ANS X3.110-1983, NAPLPS, to Accommodate Office Systems Requirements For Videotex; see also X3 Proj. 359	-	7/86			X3/85-1206
0555-DT	Technical Report for Specifying Guidelines for Implementors of X3.110-1983	-	5/86			X3/85-1214
0564-D	Extending ANS X3.110-1983, Videotex/ Teletext Presen. Level Protocol Syntax, to Include Photographic Image Coding	-	6/86			X3/85-1212
	<b><u>X3L2.1 - VIDEOTEX/TELETEXT</u></b>					
0359-M	Videotex/Teletext Presentation Layer Protocol Syntax (North American PLPS)	X3.110-1983	1988	97.02.14	ISO 6937/1-2	SPARC/81-684
0572-DT	Technical Report for the NAPLPS Verification Test Package	-	1986			X3/85-1213
	<b><u>X3L2.2 - ADDITIONAL CONTROL FUNCTIONS FOR X3.64</u></b>					
0004-R	Control Codes for 8-Bit Sets	X3.64-1979	1986	97.02.08	ISO 6429-83	SPARC/83-543 83-544
	<b><u>X3L2.3 - TWO-BYTE GRAPHIC CHARACTER SET</u></b>					
0396-D	Two-Byte Graphic Character Set for Processing and Interchange	-	1985	97.02.18		SPARC/83-545
	<b><u>X3L8 - DATA REPRESENTATION</u></b>					
0043-W	Guide for Standardization of Representation of Data Elements	X3TR-X-198x	1986	97.14.4		
0045-M	Representation for U.S. Customary, SI and other Units to be used in Systems with Limited Character Sets	X3.50-1986	1986	97.14.2	ISO 2955-83	X3/84-1418
0083-M	Representation of Calendar Date and Ordinal Date for Information Interchange	X3.30-1985	1990			
0084-M	Representation of Local Time of Day for Information Interchange	X3.43-1986	1990	97.14.3.1	ISO 3307-75	X3/84-1414
0085-M	Representations of Universal Time, Local Time Differentials, and U.S. Time Zone References for Information Interchange	X3.51-1986	1986	97.14.3.2	ISO 4031-78	X3/84-1419
0086-W	Identification of Individuals	(X3 Project Suspended)				
0087-D	Structure for Identification of Organizations	-		97.14.5	ISO 6523-84	
0088-W	Identifiers of Accounts	-	1986	97.14.5	DIS 6523	
0090-R	Identification of the States, the District of Columbia, and the Outlying Areas of the U.S. for Info. Interchange	X3.38-198x	1986			X3/85-806
0091-R	Structure for the Identification of the Counties of the U.S. for Information Interchange	X3.31-198x	1986			X3/84-1413
0092-R	Structure for the Identification of Named Populated Places and Related Entities of the States of the U.S.	X3.47-198x	1986			X3/84-1417
0093-M	Representation of Geographic Point Locations for Information Interchange	X3.61-1986	1986	97.14.6	ISO 6709-83	X3/84-1420
0095-W	Representation of Mailing and Shipping Addresses (Liaison with TC154/SC1)	-	N/A		TC154/SC1	
0096-L	Identification of Countries (ANSC Z39/SC27)	Z39.27-1976			ISO 3166-81	
0097-L	Identification of Sub-Divisions of Countries	-		TC46/WG2	ISO 3166-81	

52

52

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
<u>X3L8 - DATA REPRESENTATION (CONTINUED)</u>						
0241-W	Identification of Continents	-	N/A	97.14.9	DP 4827	
0242-L	Representation of Human Sexes (Liaison with TC97/SC14)	-		97.14.10	ISO 5218-77	
0243-L	Representation of Human Blood Type (Liaison with TC97/SC14)	-		97.14.11		
0244-W	Representation of Human Blood Type Classifications (Liaison with TC97/SC14)	-	N/A	97.14.13		
0258-W	Representation of Classifications of Occupations (Liaison with TC97/SC14)	-	N/A	97.14.8	DP 4828	
0259-L	Check Characters (Liaison with TC97/SC14)	-		97.14.15	ISO 7064-83	
0608-D	Groupings of Geopolitical Entities of the World	-				X3/86-1949R
0510-M	ANS for Codes for Identification of Hydrologic Units in the U.S. and the Caribbean Outlying Areas	X3.145-1986				X3/84-1421
<u>X3L8.5 - ATTRIBUTES OF DATA ELEMENTS</u>						
0400-DT	Technical Report for a Guideline for the Development of Attributes of Data Elements	-	1986			SPARC/83-585
<u>X3L8.6 - CLASSIFICATION OF DATA ELEMENTS</u>						
0399-DT	Technical Report for a Guideline for the Classification of Data Elements	-				SPARC/83-583
<u>X3L8.7 - MNEMONIC CODES FOR DATA ELEMENTS</u>						
0455-D	Generation of Mnemonic Codes for Data Elements and Data Items	-	1988			SPARC/83-584
<u>X3S3 - DATA COMMUNICATIONS</u>						
0110-M	Synchronous Signaling Rates for Data Transmission	X3.1-1987	1986			
0113-L	Signal Quality at Interface Between DTE and Synchronous DCE for Serial Data Transmission (Liaison with EIA TR-30)	X3.24-1968	N/A	97.06.25		
0116-L	Interface Between DTE and DCE (Liaison with EIA TR-30 on DTE/DCE Interface Definition, EIA RS-232D)	-	N/A			
0117-L	Interface Between ACU and DTE (Liaison with EIA TR-30 on EIA RS-366)	-	N/A			
0120-L	Concentration and Multiplexing Systems (Liaison with CCITT re. Concentration & Multiplex. in CCITT Access Arrangements)	-	N/A			
0121-L	Interface Between Connecting Arrangements and DTE (Liaison with EIA TR-30)	-	N/A			
0122-L	Connector Pin Allocations for Use with High Speed DTE (Liaison with EIA TR-30 on Connector Pin Assignment)	-	N/A	97.06.09	ISO 2593	
0245-L	Fault Isolation Methods and Remote Test - Public Data Networks (Liaison with CCITT SGD on the Subject re PDNs)	-	N/A	97.06.05		
0246-L	Elec. & Mech. Charac. of Interfaces (IC Techn. (EIA RS 422 & 423) 25-Pin, 37 Pin 15-Pin DTE/DCE Connector and Pin Assign.	-	N/A	97.06.09	ISO 2110,4902-3	
0260-L	Data Transmission over Telephone Type Facilities - Liaison with CCITT, COMKVII (Active Liaison through CCITT SG D)	-	N/A	97.06.22		
0261-L	Other TC97/SC6 Liaison Activities	-	N/A	97.06.23		
0280-M	Determination of Performance of Data Communications Systems that Use Bit- Oriented Control Procedures	X3.79-1987	1986			

P. 346

<u>X3 PROJ.</u> <u>NO/TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CMPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<u>X3S3 - DATA COMMUNICATIONS (CONTINUED)</u>					
0288-D	Start/Stop Signal Quality Between DTE & Non-Synchronous DCE (Liaison with EIA TR 30 on EIA RS-303-79)	-	1986	97.06.25	DIS 7480	
0289-L	Electrical Characteristics of Balanced Voltage Digital Interface Circuits (EIA SP-1220, REV. OF RS-422)	-	N/A	97.06.25		
0290-L	Electrical Characteristics of Unbalanced Voltage Digital Interface Circuits (EIA-1221, REV. OF RS-423)	-	N/A	97.06.25		
0298-L	DTE's Functional and Electrical Specifications (Liaison with EIA TR-30)	-	N/A	97.06.25		
0324-S	OSI Reference Model, Physical Layer	-	1987	97.06.30		
0332-M	IPS - OSI - Connection Oriented Transport Layer Protocol Specification (See also CCITT X.214 and X.224)	X3.140-1986	1986	97.06.35	ISO 8072 & 8073	SPARC/79-330R
0368-L	X3S3 Liaison Project with IEEE Project 802 on Local Area Networks	-	N/A	97.06.16	DP 8802/2,3,4	SPARC/82-978
0462-D	Protocol Providing Connectionless Transport Service Using Connectionless or Connection-Oriented Network Service	-	1987	97.16.14	TC97 N1254	SPARC/84-367
0493-D	Transport Service Definition to Provide Connectionless-mode Data Transmission	-	1987			SPARC/84-366 REV.
0598-I	TC97 NWI N 1606, Procedures for Testing Conformance to ISO 8073, Transport Protocol	-			ISO 1606	
0622-I	TC97 NWI N1794, Data Link Layer Management	-		97.06.44	TC97 N1794	
0625-I	TC97 N1797, Local Area Networks - Logical Link Control, Type 3 Operation - Acknowledged Connectionless Service	-		97.06.43.02.02	TC97 N1797	
0629-I	TC97 N1802, LAN Communication Interface Connectors	-		97.06.47	TC97 N1802	
0643-DT	Technical Report on Network Layer Routing Architecture	-				X3/87-08-108
0644-D	End System to Intermediate System Routing Information Exchange Protocol for Use in Conjunction with ISO 8473	-				X3/87-08-109
0648-D	End System to Intermediate System Routing Information Exchange Protocol for Use in Conjunction with ISO 8208	-				X3/87-08-110
	<u>X3S3.1 - DATA COMMUNICATIONS PLANNING</u>					
0248-L	Data Transmission Vocabulary	-	N/A	97.06.14	ISO 2382/09	
	<u>X3S3.3 - NETWORK LAYER</u>					
0047-M	Structure for Formatting Message Headings for Information Interchange Using ASCII for Data Comm. Sys. Control	X3.57-1986	1986			
0111-M	Bit Sequencing of ASCII in Serial-By-Bit Data Transmission	X3.15-1983	1988	97.06.01	ISO 1177	
0112-M	Character Structure and Character Parity Sense for Serial-By-Bit Data Communication in ASCII	X3.16-1983	1988	97.06.01	ISO 1155/1177	
0114-M	Character Structure and Character Parity Sense for Parallel-By-Bit Data Communication in ASCII	X3.25-1983	1988	97.06.01		
0281-D	Code Independent Message Heading Format (From 47)	-	1986			
0326-S	OSI Reference Model, Network Layer	-	1986	97.06.32	DIS 8348	
0365-D	ANS for Open Systems Interconnection-- Reference Model Internetwork Protocol of the Network Layer	-	1987	97.06.16	DP 7777 & 8473	SPARC/82-134
0549-D	Information Processing Systems - Data Communications - Network Layer Addressing	-	1987	97.06.32.05		X3/85-375R

52

52

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. CNFL. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
	<b>X3S3.3 - NETWORK LAYER (CONTINUED)</b>					
0550-D	Information Processing Systems - Data Communications - Network Layer Service Definition	-	1986	97.06.32.01,.02		X3/85-377R
0551-DT	Information Processing Systems - Data Communications - Technical Report on Network Layer Architecture	-	1988	97.06.32.03		X3/85-376R
0623-I	TC97 NWI N1795, Local Area Networks Logical Link Control - Flow Control	-		97.06.42.02.01	TC97 N1795	
0627-I	TC97 N1799, Provision of the Underlying Service Assumed by ISO 8473 Over Point-to-Point Subnetworks which provide OSI	-		97.06.32.04.03	TC97 N1799	
0628-I	TC97 N1801, OSI Network Layer - Intermediate System Functions	-		97.06.46	TC97 N1801	
	<b>X3S3.4 - CONTROL PROCEDURES</b>					
0048-M	Procedures for Use of the Communication Control Characters of the ASCII Code in Specified Data Communication Links	X3.28-1986	1986	97.06.01	ISO 1745-75	X3/84-1470
0049-R	Advanced Data Communication Control Procedures (ADCCP)	X3.66-1979	1986	97.06.16	ISO 3309 & 4335	X3/84-1127R
0325-S	OSI Reference Model, Data Link Layer	-	1987	97.06.31		
0624-I	TC97 NWI N1796, Asynchronous Start/Stop Transmission Operation of HDLC Protocols	-		97.06.16.01.03	TC97 N1796	
	<b>X3S3.5 - COMMUNICATION SYSTEMS PERFORMANCE</b>					
0028-RF	Determination of the Performance of Data Communication Systems	X3.44-198X	1986			
0319-M	Data Communications Systems and Services - Measurement Methods for User Oriented Performance Evaluation	X3.141-1986	1986			SPARC/78-121
0320-M	Data Communication User Oriented Performance Parameters	X3.102-1983	1988			SPARC/78-1
	<b>X3S3.7 - PUBLIC DATA NETWORK ACCESS</b>					
0223-D	General Purpose Interface Between DTE & DCE for Synchronous Operation on Public Data Networks (Liaison with CCITT X.21)	X3.69-198X	N/A	97.06.11.21		
0278-D	Network Characteristics Including User Classes of Service Facilities for Public Data Networks	-	1986	97.06.13		
0279-M	Interface Between DTE & DCE for Packet Mode Operation with Packet Switch Data Communications Networks (CCITT X.25)	X3.100-1983	1988	97.06.11		
0364-D	Standard Interface for Data Terminal Equipment Operating in the Packet Mode (See also CCITT X.25)	-	1986	97.06.11	DP 8208	SPARC/82-274
0508-I	NWI TC97 N1433, Design of Procedures for Testing & Conformance to Data Communications Protocol Based on CCITT X.25	-	N/A	97.06.38	TC97 N1433	X3/84-1167
0626-I	TC97 N1798, Provision of OSI Network Service over X.25 Permanent Virtual Circuits	-		97.06.45	TC97 N1798	
	<b>X3T1 - DATA ENCRYPTION</b>					
0293-M	Data Encryption Algorithms	X3.92-1987	1986			SPARC/77-16
0295-M	Data Link Encryption	X3.105-1983	1988			SPARC/77-96
0308-S	Data Encryption	-				
0323-M	Modes of Operation for the Data Encryption Algorithms	X3.106-1983	1988			
0339-D	Presentation (Level 6) Encryption and Decryption	-				SPARC/80-209
0340-D	Encryption and Decryption at Transport Level 4	-				SPARC/80-210

52

<u>X3 PROJ.</u> <u>ID/TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>COMPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
0040-M	<u>X3T2 - DATA INTERCHANGE</u> Information Processing - Specification for a Data Descriptive File for Information Interchange	ISO 8211-1986	1991	97.15.6	ISO 8211	SPARC/79-514
0104-RF	Representation of Numeric Values in Character Strings for Information Interchange	X3.42-1975	1986	97.15.4	ISO 6093.2	
0520-I	Information Processing - OSI - Specification of Abstract Syntax Notation One (ASN.1)	-		97.21.17.03	DIS 8824	N/A
0521-I	Information Processing - OSI - Specification of Basic Encoding Rules For Abstract Syntax Notation One (ASN.1)	-		97.21.17.04	DIS 8825	N/A
0558-D	Presentation Transfer Syntax and Notation	-	10/86			X3/85-954
0593-D	Common Language-Independent Data Types	-				X3/87-428R
0602-D	Language Independent Procedure Calls	-	6/89	97.22.17		X3/87-429R
	<u>X3T5 - OPEN SYSTEMS INTERCONNECTION</u>					
0202-DT	Operating Systems Command and Response Language	-	1985			SPARC/79-20, X3/86-365R
0300-L	Open Systems Interconnection	-		97.16.08..09		SPARC/77-112
0410-I	NWI TC97 N1216, OSI Security Architecture	-		97.16.11		SPARC/84-032
0411-I	NWI TC97 N1217, OSI Naming and Addressing	-		97.16.12		SPARC/84-032
0458-S	Study Project for OSI Protocol Conformance Test Standardization	-	1985			SPARC/84-168
0461-I	NWI TC97 N1253, Addendum to the Transport Service Standard for Connectionless Mode Data Transmission	-		97.16.13		SPARC/84-206
0463-I	NWI TC97 N 1278, Formal Description Techniques	-		97.16.15		SPARC/84-312
0464-I	NWI TC97 N1265, OSI Registration Authority Framework	-		97.16.16		SPARC/84-314
0465-I	NWI TC97 N1266, OSI Common Application Service Elements and Protocol	-		97.16.17		SPARC/84-313
0526-I	NWI TC97 N1524, OSI Management Information Services	-		97.21.28	TC97 N1524	X3/85-723
0527-I	NWI TC97 N1525, OSI Directory Services and Protocols	-		97.21.29	TC97 N1525	X3/85-722
0613-I	TC97 NWI N1790, Basic Reference Model of Open Distributed Processing	-			TC97 N1790	
0614-I	TC97 NWI N1792, Conformance Test Suites for FTAM	-	N/A	97.21.41	TC97 N1792	
0615-I	TC97 NWI N1785, Conformance Test Cases For Session, Presentation and Common Application Protocols	-	N/A	97.21.35	TC97 N1785	
0617-I	TC97 NWI N1786, Connectionless Session Protocol to Provide Connectionless-Mode Session Service	-		97.21.36	TC97 N1786	
0618-I	TC97 NWI N1788, Addendum to the Presentation Service Standard for Connectionless Mode Data Transmission	-		97.21.38	TC97 N1788	
0619-I	TC97 NWI N1789, Connectionless Presentation Protocol to Provide Connectionless-Mode Presentation	-		97.21.39	TC97 N1789	
0621-I	TC97 NWI N1791, Terminal Management	-		97.21.40	TC97 N1791	
0631-I	NWI TC97 N1742, Transaction-Mode Application Service Element and Protocol for OSI	-	N/A		TC97 N1742	
0642-D	Application Layer Service Definition Supported by Presentation Connectionless Mode Transmission	-				X3/87-08-002
0650-D	Application Layer Protocol Definition Supported by Presentation Connectionless Mode Transmission	-				X3/87-08-001

<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. CNFL. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
0327-S	<u>X3T5.1 - OSI ARCHITECTURE</u> Development of the X3 Master Plan OSI Reference Model (Aligned with CCITT X.200)	-		97.16.1.8, 97.16.1.9	ISO 7498	SPARC/79-330R
0348-D	<u>X3T5.4 - OSI MANAGEMENT PROTOCOLS</u> OSI Management Protocols	-		97.16.07		SPARC/80-446R
0333-M	<u>X3T5.5 - APPLICATION &amp; PRESENTATION LAYERS</u> Open Systems Interconnection - Basic Connection Oriented Session Protocol Specification	X3.153-1987	1985	97.16.03		SPARC/79-331 REV.
0334-D	Application Layer Protocols	-		97.16.04, .05,.06		SPARC/79-333 REV. 1/7/80
0335-D	Presentation Layer Services & Protocols	-		97.16.10		SPARC/79-332 REV. 1/7/80
0249-L	<u>X3T9 - I/O INTERFACE</u> Interface Between Computing Systems and Industrial Processes	-		97.13.4-5		
0374-L	Liaison with IEEE Project 802 on Local Area Networks	-				SPARC/83-34
0376-L	Liaison with ECMA TC24 on Local Area Networks	-				SPARC/83-164
0467-M	Intelligent Peripheral Interface, Logical Device Specific Command Sets for Magnetic Disk Drives	X3.130-1986	1986			SPARC/84-221
0468-D	Intelligent Peripheral Interface, Logical Device Generic Command Set	-	1986			SPARC/84-222
0496-M	Intelligent Peripheral Interface - Logical Device Generic Command Set for Optical and Magnetic Disks	X3.132-1987	1986			X3/84-1090
0503-D	Fiber Distributed Data Interface (FDDI) Station Management (SMT) Standard	-	1987			X3/84-1093
0504-D	Intelligent Peripheral Interface - Logical Device Generic Command Set for Communications	-	1986			X3/84-109
0505-M	Intelligent Peripheral Interface - Logical Device Generic Command Set for Magnetic Tape	X3.147-1986	1986			X3/84-1091
0541-D	ANS for Fiber Distributed Data Interface (FDDI) Physical Layer, Medium Dependent (PMD)	X3.166-198x	1987			X3/85-985
0587-D	Enhanced Small Device Interface (ESDI)	-				X3/86-1673
0591-D	Intelligent Peripheral Interface, Logical Device Specific Command Set for Magnetic Tapes	-				X3/86-1687
0651-D	Fiber Distributed Data Interface (FDDI) Physical Medium Dependent Single-Mode (PMD-SM) & Assignment to X3T9	-				X3/87-09-022
0375-R	<u>X3T9.2 - LOWER LEVEL INTERFACE</u> Small Computer Systems Interface (SCSI)	X3.131-198x	1986	97.13.10	TC97 N1431	SPARC/82-878
0052-R	<u>X3T9.3 - DEVICE LEVEL INTERFACE</u> Interfaces Between Flexible Disk Cartridges and Their Host Controllers	X3.80-198x	1986	97.13.10	TC97 N1431	SPARC/83-360, REV.
0053-M	Storage Module Interfaces	X3.91M-1987	1986	97.13.7, .10	TC97 N1431	X3/84-1449
0054-L	Small Computer to Peripheral Bus Interface, Data Transfer Between Computer and Peripheral	-		97.13.9	DP 7069	
0329-M	Interfaces Between Rigid Disk Drives and Hosts	X3.101-1984	1989	97.13.10	TC97 N1431	SPARC/79-342
0370-R	Intelligent Peripheral Interface Physical Level	X3.129-198x		97.13.10	TC97 N1431	X3/86-1688



<u>X3 PROJ. NO./TYPE</u>	<u>TITLE</u>	<u>STD. DESIG.</u>	<u>EST. CPL. DATE</u>	<u>ISO PROJ. DESIG.</u>	<u>ISO DOC. NO.</u>	<u>SD-3 REF. NUMBER</u>
	<b><u>X3T9.5 - LOCAL DISTRIBUTED DATA INTERFACE</u></b>					
0337-D	Physical Layer Interface for Local Distributed Data Interfaces to a Non-Branching Coaxial Cable Bus	X3.108-198x	1986			SPARC/80-145 REV. 3/19/80
0338-D	Data Link Layer Protocol for Local Distributed Data Interfaces	-	1986			SPARC/80-146 REV. 3/19/80 SPARC/81-704
0357-D	Physical Layer Protocol for Local Distributed Data Interface	-	1986			
0377-D	Local Distributed Data Interface (LDDI) Network Layer Protocol	-	1987			SPARC/82-1056
0379-D	Fiber Distributed Data Interface (FDDI) Physical Layer	X3.148-198x	1987			SPARC/82-1058
0380-M	Fiber Distributed Data Interface (FDDI) Token Ring Media Access Control (MAC)	X3.139-1987	1986			SPARC/82-1059
0382-D	Fiber Distributed Data Interface (FDDI) Network Layer Protocol	-	1987			SPARC/82-1060
0556-D	ANS for Local Distributed Data Interface (LDDI) Star-Wired Physical Interface Sublayer	X3.167-198x	1987	97.13.01		X3/85-986
0573-D	Fiber Distributed Data Interface (FDDI) Hybrid Ring Control (HRC)	-	1987	97.13.01		X3/85-1903R
	<b><u>X3T9.6 - CARTRIDGE TAPE DRIVES</u></b>					
0378-M	Device Level Interface for Streaming Cartridge and Cassette Tape Drives	X3.146-1986	1987			X3/85-0029
0401-D	Enhanced Device Level Interface for Cartridge Tape Drives	-	1986			X3/85-0030
	<b><u>X3V1 - TEXT: OFFICE &amp; PUBLISHING SYSTEMS</u></b>					
0576-D	SGML Document Interchange Format (SDIF)	-	12/86 1987		DIS 8879	X3/86-715R X3/86-1035R
0580-	Print Image for Interchange - Office Systems	-				
	<b><u>X3V1.1 - USER REQUIREMENTS: M.S.T.</u></b>					
0412-I	Text Preparation and Interchange Equipment: Basic and Optimal Requirements	-		97.18.18		SPARC/84-033
0414-I	Text Preparation and Interchange Equipment: Classification & Terminology	-		97.18.20		SPARC/84-033
0417-I	Text Preparation and Interchange Equipment: Minimum Requirements for Text Presentation	-		97.18.23		SPARC/84-033
0491-D	Progression of Presentation/Rendition Capabilities Relative to User Requirements	-	1985			X3/84-684
0497-L	User Requirements for Text Preparation, Interchange and Presentation	-		97.18.07	SC18 N274 Rev.	N/A
0498-L	User Requirements for Text Preparation and Interchange	-		97.18.07.01		N/A
0499-L	Reference Model for Text Preparation and Interchange	-		97.18.07.02		N/A
0500-L	Message-Oriented Text Interchange System User Requirements	-		97.18.07.03		N/A
	<b><u>X3V1.3 - DOCUMENT ARCHITECTURE</u></b>					
0384-D	Office Document Architecture	-		97.18.10.01	DP 8613/1	SPARC/83-108
0385-D	Office Document Architecture - Document Description	-		97.18.10.03	DP 8613/3	SPARC/83-107
0478-D	Office Document Interchange Format	-	1985	97.18.10.04	DP 8613/4	SPARC/84-357
0501-I	Text Structures	-		97.18.10	DP 8613	N/A
0522-D	ANS for Text Interchange on Magnetic Media	-	6/86			X3/85-557

52

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CHPL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
0358-D	<u>X3V1.4 - TEXT INTERCHANGE</u> Message Protocol Standards -- Procedure for Communication of Prepared Text	-	1986	97.18.11	DP 8505, 8506	SPARC/81-495R 8/3/81
0476-D	Communication Access Requirements for Office Systems	-	1986			X3/84-680
0490-D	MOTIS (Message-Oriented Text Interchange Systems) Naming Convention and Directory Services	-	1987	97.18.11.1..2		X3/84-681
0393-D	<u>X3V1.5 - CONTENT ARCHITECTURE</u> Text Imaging Capabilities	-		97.18.12.01	DP 8613/5-10	SPARC/83-104R 7/25/83
0394-D	Positioning of Text on Hard Copy Devices	-		97.18.12.02	DP 8564	SPARC/83-105R 7/25/83
0395-D	Basic Processable Text Interchange Format	-			DP 8613/6	SPARC/83-106R 7/25/83
0449-L	Liaison with X3B9 Project 443-M on X3.117-1984, Printable/Image Areas for Text & Facsimile Communication Equipment	-		97.18.13		
0450-M	Text Information Interchange in Page Image Format	X3.98-1983	1988			
0502-I	(With X3L2; see Proj. 349-D) Characteristics of Systems Elements for the Presentation of a Text	-		97.18.12	DP 8613/5-10	N/A
0203-D	<u>X3V1.8 - TEXT DESCRIPTION &amp; PROCESSING LANGUAGES</u> Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML) (AKA-DIS 8879)	X3.143-198x	10/85	97.18.15	DP 8879/6	SPARC/79-08
0542-D	ANS for Generalized Music Representation for Information Processing	-	12/86			X3/85-964
0600-I	TC97 NWI N 1608, Description and Identification of Character Fonts	-			ISO 1608	
0413-I	<u>X3V1.9 - USER SYSTEMS INTERFACE &amp; SYMBOLS</u> NWI TC97 N1210, Text Preparation and Interchange Equipment: Test Charts and Text Patterns	-		97.18.19	SC18 N387, 388	SPARC/84-033
0415-I	NWI TC97 N1212, Text Preparation and Interchange Equipment: Graphic Symbols	-		97.18.08, .21		SPARC/84-033
0416-I	NWI TC97 N1213, Text Preparation and Interchange Equipment: Minimum Information to be Specified	-		97.18.22	SC18 N389, 390	SPARC/84-033
0418-I	NWI TC97 N1215, Layout and Operation of the Keyboard for Multiple Latin Alphabet Languages	-		97.18.24	SC18 N414	SPARC/84-033
0424-R	Keyboard Arrangement for Alphanumeric Machines	X3.154-198x	3/86			X3/85-1272
0425-M	Alternate Keyboard Arrangement for Alphanumeric Machines	X4.22-1983	1988			
0451-D	Classification of Vocabulary of Word Processing for Document Preparation	-	1985	97.18.08, .09	SC18 N55, 56	
0515-I	TC97 NWI N1450, Functionality and Layout of Function Keys	-		97.18.25	NWI TC97 N1450	X3/84-1572
0578-S	User System Interfaces for Standards Related to Text Processing: Office and Publishing Systems	-				X3/86-888R
0601-I	TC97 NWI N 1607, Document Interchange Format for Storage Media	-			ISO 1607	
0632-I	Operation Through a Switched Telephone Network, a Circuit Switched Data Network or an Integrated Ser. Digital Network	-		97.06.48	ISO/TC97 N1912	

52

<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CNFL.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
	<u>X3V1.9 - USER SYSTEMS INTERFACE &amp; SYMBOLS (CONTINUED)</u>					
0637-I	Protocol Identification in the Network Layer	-		97.06.49	ISO/TC97 N1919	
0638-I	Local Area Networks - MAC Sublayer Interconnection (MAC Bridging)	-		97.06.43.01.01	ISO/TC97 N1920	
	<u>X3V1.10 - FONT AND CHARACTER INFORMATION</u>					
0575-D	ANS for Font and Character Information Interchange	-	6/87			X3/86-716-5
	<u>SPARC/DBSSG - DATA BASE SYSTEMS STUDY GROUP</u>					
0226-S	Data Base Management Systems	-	1987	97.21.25		
0528-I	NWI TC97 N1526, Reference Model for DBMS Standards	-		97.21.30	TC97 N1526	X3/85-724
	<u>SSISG - SOFTWARE SYSTEMS INTERFACE STUDY GROUP</u>					
0599-I	Software Systems Interface	-				
	<u>SPARC - STANDARDS PLANNING AND REQUIREMENTS COMMITTEE</u>					
0076-M	One-Inch Perforated Paper Tape for Information Interchange	X3.18-1982	1987	97.0.3	ISO 1154-75	
0077-M	Eleven-Sixteenths Inch Perforated Paper Tape for Information Interchange	X3.19-1982	1987			
0078-M	Take-up Reels for One Inch Perforated Tape for Information Interchange	X3.20-1982	1987	97.0.6	ISO 3692-76	
0079-M	Specifications for Properties of Unpunched Oiled Paper Perforator Tape	X3.29-1984	1989			
0080-M	Interchange Rolls of Perforated Tape for Information Interchange	X3.34-1984	1989			
0081-W	Flow Chart Symbols and Their Usage in Information Processing	X3.5-1970	1985	97.7.1-2,4	ISO 1028 & 2636	
0211-M	Computer Program Abstracts	X3.88-1987	1986			
0218-M	Representation of Vertical Carriage Positioning Characters in Information Interchange	X3.78-1987	1986			
0253-M	Programming Language - Programming Aid for Numerically Controlled Manufacturing	X3.94-1985	1990			
0263-L	Programming Language for Industrial Process (PLIP)	-		97.5.5		SPARC/649
0265-L	Program Design	-		97.7.3		
0267-S	Long Range Planning for Programming Language Standards	-				
0269-L	Programming Language MUMPS (MUMPS Development Committee)	-				
0342-L	SPARC Liaison Project with DOD High Order Language Group on Ada*	-				X3/80-307
0422-M	Minimum Markings to Appear on Containers Used for Printing Ribbons	X4.19-1985	1990			
0423-M	Office Machines & Printing Machines Used for Information Processing, Widths of Fabric Ribbons on Spools	X4.20-1985	1990			
0426-M	10-Key Keyboard for Adding and Calculating Machines	X4.6-1984	1989			
0427-M	Minimum Requirements for Office-Type Dictating Equipment	X4.9-1984	1989			
0428-M	Remote Dictation Through an Intercommunications Switching System	X4.10-1984	1989			
0452-S	Programming Languages Study	-				SPARC/83-55R
	<u>SC7 TAG - U.S. TAG TO TC97/SC7</u>					
0469-I	NWI TC97 N1248, Guidelines for Software Development Methods	-		97.07.18	TC97 N1248, 1445	SPARC/83-710
0604-I	TC97 NWI N1627, Standard Diagrams for Software Development Models	-				



<u>X3 PROJ.</u> <u>NO./TYPE</u>	<u>TITLE</u>	<u>STD.</u> <u>DESIG.</u>	<u>EST.</u> <u>CONF.</u> <u>DATE</u>	<u>ISO</u> <u>PROJ.</u> <u>DESIG.</u>	<u>ISO</u> <u>DOC.</u> <u>NO.</u>	<u>SD-3 REF.</u> <u>NUMBER</u>
0513-I	<u>SC21 TAG - U.S. TAG TO TC97/SC21</u> TC97 NWI N1448, A Survey of DBMS Related Standardization Activities	-	N/A	97.21.25	NWI N1448	X3/84-1550
0548-I	TC97 NWI N1555, Information Processing - Remote Database Access Service and Protocol	-		97.21.31	TC97 N1555	X3/85-1146
0473-I	<u>SC22 TAG - U.S. TAG TO TC97/SC22</u> NWI TC97 N1315, Guidelines for Preparation of Programming Language Standards	-		97.22.13		SPARC/84-175
0474-I	NWI TC97 N1316, Binding Techniques for Programming Languages	-		97.22.14		SPARC/84-176
0512-L	U.S. TAG to TC97/SC22/WG12, Conformance and Validation	-				N/A
0523-I	NWI TC97 N1519, Conformity Requirements and Testing in Programming Language Standards	-		97.22.15		X3/85-608
0605-I	TC97 NWI N 1773, Portable Operating Sys. Interface for Computer Environments, POSIX	-			ISO 1773	
0606-I	TC97 NWI N 1602, Specification of Computer Programming Language Modula 2	-			ISO 1602	
0616-I	TC97 N1594, Programming Language ALGOL '68	-			TC97 NWI N1594	

X3/SD-7

53

December 1987

ACCREDITED STANDARDS COMMITTEE\*  
X3-INFORMATION PROCESSING SYSTEMS

---

MEETING SCHEDULE &  
CALENDAR

---

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

SECRETARIAT:

Computer and Business Equipment Manufacturers Association



## X3 Standing Documents

This document is one of a series, developed by X3 and the X3 Secretariat, which provides a "data base" of information on Accredited Standards Committee X3 - Information Processing Systems. Each document is updated periodically on an individual basis.

The series is intended to serve several purposes:

- o To describe X3 and its program to inquirers
- o To inform committee members of the organization and operation of X3
- o To provide a system of orderly administration incorporating the procedures required by ANSI together with supplements approved by the X3 Secretariat, for the guidance of X3 officers, members, subgroups and the Secretariat staff.

The series of Standing Documents consists of the following:

X3/SD-0	Informational Brochure - September 1985
X3/SD-1	Master Plan - May 1987
X3/SD-2	Organization & Procedures - October 1985
X3/SD-3	Project Proposal Guide - May 1987
X3/SD-4	Projects Manual - December 1987
X3/SD-5	Standards Criteria - September 1984
X3/SD-6	Membership and Officers - December 1987
X3/SD-7	Meeting Schedule and Calendar - December 1987
X3/SD-9	Policy and Guidelines - (to be issued)
X3/SD-10	X3 Subgroup Annual Report Format - June 1987

### SD-7, MEETINGS SCHEDULE & CALENDAR

X3/SD-7 provides the current scheduled meeting dates and their locations which have been reported to the X3 Secretariat. It is suggested that anyone attending a committee meeting for the first time contact the committee chair (see X3/SD-6) and confirm that there are no changes in the schedule.

Corrections and suggestions for improvement will be welcomed, and should be addressed to:

X3 Secretariat/CBEMA  
 Attn: Nadine Morgan  
 311 First Street, NW  
 Suite 500  
 Washington, DC 20001-2178

p. 356

SD-7 -- MEETING SCHEDULE AND CALENDAR 1987/1988

53

XSTC	1988									
	JAN	FEB	MAR	APR	MAY	JUNE	JULY	AUG	SEPT	OCT
X3		9-10				14-15				12-13
SD-7 Tutorials		08				13				11
TECH ED TR.				28						
JTC1				11-15						
JTC1 AG				20-22						
JTC1 TAG		10-11				15-16			19-20	13-14
JTC1 TAG /AC	21-22		29-30		25-26					
SMC	26		28		18		20		27	
SPARC	12-14		22-25		3-5		12-14		13-15	
SPARC/DBSSG										
SPARC/SSISG										
SPC	28-29									
FF		1 & 8								
SWG/FS TAG				6						
X3A1			09-11						TBA	
X3A1.1										
X3A1.2										
X3A1.3										
X3B5		16-18			10-12					
X3B6	15-16		23-24			15-16				06-07
X3B7	12-13				11-12				14-15	
X3B8										
X3B8.1										
X3B9										
X3B10										
X3B10.1										
X3B10.2										
X3B10.3										
X3B10.4										
X3B11										
X3H2		08-10		11-14				22-25		
X3H3										
X3H3.1										
X3H3.2										
X3H3.3										
X3H3.4										
X3H3.5										
X3H3.6										
X3H4		08-11		19-22						
X3H4.1										
X3J1										
X3J1.3										
X3J2	25-26							08-12	24-25	
X3J3		08-12			09-13					
X3J4	25-29				10-13		19-22		03-06	
X3J7										
X3J7.1										
X3J7.2										
X3J7.3										
X3J7.4										
X3J9			08-11			07-10			13-16	
X3J9.1										
X3J10										
X3J11										
X3J12										
X3J13										
X3J14										
X3K1										
X3K5	12-15		22-25		17-20	6-10	19-22		27-30	
X3L2		29-02				20-24			12-16	
X3L2.1		29-30				20-22			12-13	
X3L2.2		29-30				20-22			12-13	

XSTC	1988									
	JAN	FEB	MAR	APR	MAY	JUNE	JULY	AUG	SEPT	OCT
X3L2.3		29-30				20-22			12-13	
X3L8		01-03				06-08				
X3L8.4										
X3L8.5										
X3L8.6										
X3L8.7										
X3S3	12		1-2		24-25		19-21	30		
X3S3.1										
X3S3.2										
X3S3.3	11-15									
X3S3.4										
X3S3.5										
X3S3.7		22-26								
X3T1										
X3T2	27-29			27-29			27-29			TEA
X3T5		22-25		19-20		7-8				
X3T5.1		23-24								
X3T5.4		23-24								
X3T5.5		23-24							15-19	
X3T9		22-26		25-29		20-24				
X3T9.2										
X3T9.3										
X3T9.5										
X3T9.6										
X3V1			14-18	25-29		6-10			19-23	
X3V1.1										
X3V1.3										
X3V1.4										
X3V1.5										
X3V1.8										
X3V1.9										
ANSI Conference			15-16							
ANSI/ISSB		18								
SC21 TAG										
SC22 TAG										

53

SD-7 — MEETING SCHEDULE AND CALENDAR 1988/89

XSTC	1988			1989				
	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUNE
X3				07-08				13-14
SD-2 Tutorial				06				6-9
JTC1				08-09				14-15
JTC1 TAG								
SMC	09						TEA	
X3A1								
X3J2	24-25							
X3J4	03-06							
X3J9					07-10			06-09
X3L2		05-09						
X3L2.1		05-06						
X3L2.2		05-06						
X3L2.3		05-06						

P. 358



JANUARY 1988 DATES AND LOCATIONS

X3S3.3	11-15	Santa Clara, CA
X3S3	12	Washington, DC-CBEMA
X3B7	12-13	Santa Ana, CA
SPARC	12-14	San Francisco, CA
X3K5	12-15	Washington, DC-CBEMA
X3B6	15-16	Anaheim, CA
JTC1/AC	21-22	Washington, DC-CBEMA
X3J2	25-26	San Francisco, CA
X3J4	25-29	La Jolla, CA
SMC	26	Washington, DC-CBEMA
X3T2	27-29	Los Angeles, CA
SPC	28-29	Washington, DC-CBEMA

FEBRUARY 1988 DATES AND LOCATIONS

SD-2 Tutorial	08	Monterey, CA
FF	01	Washington, DC-CBEMA
FF	08	Monterey, CA
X3L8	01-03	Washington, DC-CBEMA
X3H2	8-10	Orlando, FL
X3H4	08-11	Washington, DC
X3J3	08-12	New Orleans, LA
X3	9-10	Monterey, CA
JTC1 TAG	10-11	Monterey, CA
X3B5	16-18	Riviera Beach, FL
ISSB	18	New York, NY
X3T5	22-25	Washington, DC-CBEMA
X3S3.7	22-26	Palo Alto, CA
X3T9	22-26	TBA
X3T5.1	23-24	Washington, DC-CBEMA
X3T5.4	23-24	Washington, DC-CBEMA
X3T5.5	23-24	Washington, DC-CBEMA
X3L2.1	29-30	Dallas, TX
X3L2.2	29-30	Dallas, TX
X3L2.3	29-30	Dallas, TX
X3L2	29-02*	Dallas, TX

MARCH 1988 DATES AND LOCATIONS

X3S3	01-02	Washington, DC-GSA
X3L2	—02	Dallas, TX
X3J9	08-11	Newport Beach, CA
X3A1	09-11	New Orleans, LA
X3V1	14-18	Golden, NBI
ANSI	15-16	New York, NY
SPARC	22-25	Washington, DC-CBEMA
X3K5	22-25	Washington, DC-CBEMA
X3B6	23-24	Laurel, MD
SMC	28	Washington, DC-CBEMA
JTC1/AC	29-30	Washington, DC-CBEMA

APRIL 1988 DATES AND LOCATIONS

SWG/PS TAG	06	Washington, DC-CBEMA
X3H2	11-14	Pasadena, CA
X3T5	19-20	Washington, DC-CBEMA
X3H4	19-22	San Francisco, CA
JTC1 AG	20-22	Washington, DC-CBEMA
X3T9	25-29	TBA
X3V1	25-29	SC18 Plenary, Hague
X3T2	27-29	Washington, DC-CBEMA
TECH ED	28	Washington, DC-CBEMA

MAY 1988 DATES AND LOCATIONS

SPARC	03-05	Scottsdale, AZ
X3J3	09-13	Urbana, IL
X3B5	10-12	Annapolis, MD
X3J4	10-13	Litchfield, CT
X3B7	11-12	San Francisco, CA
X3K5	17-20	Washington, DC-CBEMA
SMC	18	New York, NY
X3S3	24-25	Washington, DC-CBEMA
JTC1/AC	25-26	Washington, DC-CBEMA

JUNE 1988 DATES AND LOCATIONS

X3L8	06-08	Washington, DC-CBEMA
X3V1	06-10	Boston, DEC
X3K5	06-10	Berlin, Germany-DIN
X3T5	07-08	Washington, DC-CBEMA
X3J9	07-10	Minneapolis, MN
SD-2 Tutorials	13	Washington, DC-CBEMA
X3	14-15	Washington, DC-CBEMA
JTC1 TAG	15-16	Washington, DC-CBEMA
X3B6	15-16	San Mateo, CA
X3L2.1	20-22	Boston, MA
X3L2.2	20-22	Boston, MA
X3L2.3	20-22	Boston, MA
X3L2	20-24	Boston, MA
X3T9	20-24	TBA

JULY 1988 DATES AND LOCATIONS

SPARC	12-14	Philadelphia, PA
X3S3	19-21	Boulder, CO
X3J4	19-22	U.K.
X3K5	19-22	Washington, DC-CBEMA
SMC	20	Washington, DC-CBEMA
X3T2	27-29	Seattle, WA

AUGUST 1988 DATES AND LOCATIONS

X3J3	08-12	Estes Park, CO
X3T9	15-19	TBA
X3H2	22-25	Boulder, CO
X3S3	30	Washington, DC-CBEMA

SEPTEMBER 1988 DATES AND LOCATIONS

X3A1	TBA	Myrtle Beach, SC
X3H2	TBA	San Francisco
X3T2	TBA	Washington, DC-CBEMA
X3L2.1	12-13	Raleigh, NC
X3L2.2	12-13	Raleigh, NC
X3L2.3	12-13	Raleigh, NC
X3L2	12-16	Raleigh, NC
X3J9	13-16	Framingham, MA
SPARC	13-15	Merrimack, NH
X3B7	14-15	Las Vegas, NV
JTC1/AC	19-20	Washington, DC-CBEMA
X3V1	19-23	Atlanta, Bell Smith
X3K5	27-30	Washington, DC-CBEMA

OCTOBER 1988 DATES AND LOCATIONS

X3T9	17-21	TBA
X3L8	03-05	Washington, DC-CBEMA
X3J4	03-06	Carmel, CA
X3B6	06-07	Warminster, PA
SD-2 Tutorial	11	Washington, DC-CBEMA
X3	12-13	Washington, DC-CBEMA
JTC1 TAG	13-14	Washington, DC-CBEMA
X3J2	24-25	Annapolis, MD
X3T2	26-28	Pennsylvania, PA

NOVEMBER 1988 DATES AND LOCATIONS

SMC	09	Washington, DC-CBEMA
X3J14	10-11	Washington, DC-CBEMA
X3H2	14-16	Long Island, NY
SPARC	15-18	Washington, DC-CBEMA
X3K5	15-18	Washington, DC-CBEMA
X3S3	29-30	Washington, DC-CBEMA

DECEMBER 1988 DATES AND LOCATIONS

X3L2.1	05-06	Tampa, FL
X3L2.2	05-06	Tampa, FL
X3L2.3	05-06	Tampa, FL
X3L2	05-09	Tampa, FL
X3T9	05-09	TBA
X3V1	05-09	Raleigh, NC-IBM
X3J9	06-09	Los Angeles, CA

1989 DATES AND LOCATIONS

X3	Feb. 7-8	Phoenix, AZ
JTC1 TAG	Feb. 8-9	Phoenix, AZ
X3A1	Mar. TBA	Monterey, CA
X3J9	Mar. 7-10	Pacific Grove, CA
X3J9	Jun. 6-9	Radford, VA
JTC1	Jun. 6-9	Paris, France
X3	Jun. 13-14	Washington, DC-CBEMA
JTC1 TAG	Jun. 14-15	Washington, DC-CBEMA

X3 1988 MEETING SCHEDULE

53

February 09-10	Monterey, CA
June 14-15	Washington, DC-CBEMA
October 12-13	Washington, DC-CBEMA

X3 1989 Advance Planning:

February 07-08	Phoenix, AZ
June 13-14	Washington, DC-CBEMA
October 11-12	Washington, DC-CBEMA

JTC1 TAG 1988 MEETING SCHEDULE

January 10-11	Monterey, CA
June 15-16	Washington, DC-CBEMA
October 13-14	Washington, DC-CBEMA

JTC1 TAG 1989 Advance Planning:

February 08-09	Phoenix, AZ
June 14-15	Washington, DC-CBEMA
October 12-13	Washington, DC-CBEMA

JTC1 TAG AC 1988 MEETING SCHEDULE

January 21-22	Washington, DC-CBEMA
March 22-25	Washington, DC-CBEMA
May 25-26	Washington, DC-CBEMA
September 19-20	Washington, DC-CBEMA

SPARC 1988 MEETING SCHEDULE

January 12-14	San Francisco, CA
March 22-25	Washington, DC-CBEMA
May 03-05	Scottsdale, AZ
July 12-14	Philadelphia, PA
September 13-15	Merrimack, NH

SMC 1988 TENTATIVE MEETING SCHEDULE

January 26	Washington, DC-CBEMA
March 28	Washington, DC-CBEMA
May 18	New York, NY
July 20	Washington, DC-CBEMA
September 27	Washington, DC-CBEMA
November 9	Washington, DC-CBEMA

SPC 1988 MEETING SCHEDULE

January 28-29	Washington, DC-CBEMA
---------------	----------------------

FF 1988 MEETING SCHEDULE

February 1	Washington, DC-CBEMA
February 8	Monterey, CA

SWG/FS TAG 1988 MEETING SCHEDULE

April 6	Washington, DC-CBEMA
---------	----------------------

<u>Meeting</u>	<u>YeMonDay</u>	<u>Place</u>	<u>City</u>	<u>Ref</u>
<u>ISO/IEC JTC 1 - INFORMATION TECHNOLOGY</u>				
JTC 1 Plenary	87/11/17-20	New Otani Hotel	Tokyo	JTC 1 N 1
Taxonomy Group	87/11/9-13	Ottawa	Canada	JTC 1 N 70 Rv
Taxonomy Group	88/05/10-13		Tokyo	JTC 1 N 70 Rv
SG on Funct. Stds.	88/05/16-18		Tokyo	JTC 1 N 70 Rv
SG on Funct. Stds.	89/Jan/Feb	to be confirmed	Copenhagen	JTC 1 N 70 Rv
SG on Funct. Stds.	89/Oct/Nov	to be confirmed	Munich	JTC 1 N 70 Rv

SC 1 - VOCABULARY

JTC 1 SC01 Plenary	88/06/06-10	To be determined	Berlin	JTC 1 N 30
JTC 1 SC01 Plenary	89/00		Japan	97/1 N 1060
JTC 1 SC01 WG04	87/11/2-6		Paris	JTC 1 N 30
JTC 1 SC01 WG05	87/11/2-6		Munich	JTC 1 N 30
JTC 1 SC01 WG06	87/11/16-20		Paris	JTC 1 N 30
JTC 1 SC01 WG06	87/12		Wash. DC	1 N 1060

SC - 2 CHARACTER SETS AND INFORMATION CODING

JTC 1 SC02 Plenary	88/10/17-21	to be determined		JTC 1 N 31
JTC 1 SC02 WG01	89/03/14-18		Paris	JTC 1 N 31
JTC 1 SC02 WG02	88 Spring		Boston	JTC 1 N 31
JTC 1 SC02 WG02	88 Summer		London	JTC 1 N 31

Meeting	Year	Day	Place	City	Ref
---------	------	-----	-------	------	-----

53

SC.6 - TELECOMMUNICATIONS AND INFORMATION EXCHANGE BETWEEN SYSTEMS

JTC 1 SC06	Planary	88/02/03-04	St. Pierre Park Hotel	Guernsey, UK	JTC 1 N 32
JTC 1 SC06	WGO1 ConTest	87/11/02-05	DN	Berlin	6 N 4813 Rec.19
JTC 1 SC06	WG01	88/01/25-02/02	St. Pierre Park Hotel	Guernsey, UK	JTC 1 N 32
JTC 1 SC06	WG01	88/09/10		Denmark	JTC 1 N 32
JTC 1 SC06	WG02	88/01/25-02/02	St. Pierre Park Hotel	Guernsey, UK	JTC 1 N 32
JTC 1 SC06	WG02	88/09/10		USA	JTC 1 N 32
JTC 1 SC06	WG03	88/01/25-02-02	St. Pierre Park Hotel	Guernsey, UK	JTC 1 N 32
JTC 1 SC06	WG03	88/09/10		Germany FR	JTC 1 N 32
JTC 1 SC06	WG04	88/09/10		Japan	JTC 1 N 32
JTC 1 SC06	WG04 ConTest	87/11/16-20	AFNOR	Paris	6/4 N 283
JTC 1 SC06	WG04	88/01/25-02/02	St. Pierre Park Hotel	Guernsey, UK	JTC 1 N 32

SC.7 - SOFTWARE DEVELOPMENT AND SYSTEM DOCUMENTATION

JTC 1 SC07	Planary	88/08	DeIR	Netherlands	JTC 1 N 33
JTC 1 SC07	WG02	88/2/02-05		Amsterdam	
JTC 1 SC07	WG04	88/4/25-27		New York	
JTC 1 SC07	WG05	87/11/16-18	Montreal	Canada	7 N 541

SC.11 - FLEXIBLE MAGNETIC MEDIA FOR DIGITAL DATA INTERCHANGE

JTC 1 SC11		89		Switz.	JTC 1 N 34
JTC 1 SC11		90		USA	JTC 1 N 34
JTC 1 SC11		91		France	JTC 1 N 34

p. 362

53

<u>Meeting</u>	<u>YeMonDay</u>	<u>Place</u>	<u>City</u>	<u>Ref</u>
----------------	-----------------	--------------	-------------	------------

SC.13 - INTERCONNECTION OF EQUIPMENT

JTC 1 SC13	Plenary	88/10/03-07	Tokyo	
------------	---------	-------------	-------	--

SC.14 - REPRESENTATIONS OF DATA ELEMENTS

JTC 1 SC14	Plenary	88	USA or China	TX dd 87/11/05
------------	---------	----	--------------	----------------

JTC 1 SC14	Plenary	89	Belgium or UK	TX dd 87/11/05
------------	---------	----	---------------	----------------

SC.17 - IDENTIFICATION AND CREDIT CARDS

JTC 1 SC017	WG04	88/02/29-03/04	San Francisco	JTC 1 N 38
-------------	------	----------------	---------------	------------

JTC 1 SC017	WG04	88/08/20-25	Copenhagen	JTC 1 N 38
-------------	------	-------------	------------	------------

SC.18 - TEXT AND OFFICE SYSTEMS

JTC 1 SC18	Plenary & HOD/C	88/04/25-29	The Hague	JTC 1 N 39
------------	-----------------	-------------	-----------	------------

JTC 1 SC18	Plenary & HOD/C	89/Apr/May	AFNOR	Paris (Tentative)
------------	-----------------	------------	-------	-------------------

JTC 1 SC18	Plenary & HOD/C	90 or 91 Apr/May		Germany (Tentative)
------------	-----------------	------------------	--	---------------------

JTC 1 SC18	WG01	88/01/25-29	USA	JTC 1 N 39
------------	------	-------------	-----	------------

JTC 1 SC18	WG01	88/04/18-22	Europe	JTC 1 N 39
------------	------	-------------	--------	------------

JTC 1 SC18	WG03	87/11/02-11	AFNOR	Paris JTC 1 N 39
------------	------	-------------	-------	------------------

JTC 1 SC18	WG04	88/02/01-05	Paris	18 N 1253
------------	------	-------------	-------	-----------

JTC 1 SC18	WG04	88/09/12-18	Spain or USA	18 N 1253
------------	------	-------------	--------------	-----------

JTC 1 SC18	WG05	87/11/09-13	AFNOR	Paris JTC 1 N 39
------------	------	-------------	-------	------------------

JTC 1 SC18	WG05	88/01/11-15	DOC	Ottawa JTC 1 N 39
------------	------	-------------	-----	-------------------

JTC 1 SC18	WG05	88/05/02-06	WANG	Brussels 18 N 1052
------------	------	-------------	------	--------------------

JTC 1 SC18	WG05	88/10/03-07	JISC	Tokyo 18 N 1052
------------	------	-------------	------	-----------------

JTC 1 SC18	WG08	88/01/18-22	Tentative	Los Angles, CA
------------	------	-------------	-----------	----------------

P. 363

<u>Meeting</u>	<u>Year/Day</u>	<u>Place</u>	<u>City</u>	<u>Ref</u>
JTC 1 SC18 WG08	88/04/18-22	BSI	London	18 N 1108
JTC 1 SC18 WG08	88/10/17-21		Paris (Tentative)	18 N 1108
JTC 1 SC18 WG08 SWG on SPDL	88/06/13-17			
JTC 1 SC18 WG09	88/03/07-11		Spain	

53

SC 20 - DATA CRYPTOGRAPHIC TECHNIQUES

JTC 1 SC20 Plenary	88/04/14-15		London, UK	JTC 1 N 40
JTC 1 SC20 WG01	88/04		London, UK	JTC 1 N 40
JTC 1 SC20 WG01	88/Autumn		Turin, Italy	JTC 1 N 40
JTC 1 SC20 WG02	88/April		London, UK	JTC 1 N 40
JTC 1 SC20 WG03	88/April		London, UK	JTC 1 N 40
JTC 1 SC20 WG03	88/Autumn		Turin, Italy	JTC 1 N 40

SC 21 - INFORMATION TRANSFER AND MANAGEMENT FOR OPEN SYSTEMS INTERCONNECTION

JTC 1 SC21 Plenary	88/03/15-16	Quality Inn	Wash., DC	JTC 1 N 41
JTC 1 SC21 HOD/C	88/03/11-14	CBEMA	Wash., DC	21 N 1901
JTC 1 SC21 Planning	87/11/9-13	AFNOR	Paris	21 N 1901
JTC 1 SC21 WG01	88/02/29/03-08	McLean	VA, USA	JTC 1 N 41
JTC 1 SC21 WG01	88/11/28/12-14		Sydney Australia	JTC 1 N 41
JTC 1 SC21 WG01 Open distributed processing	87/11/02-08		Cambridge	21 N 2021

P. 364

53

<u>Meeting</u>	<u>YeMonDay</u>	<u>Place</u>	<u>City</u>	<u>Ref</u>
JTC 1 SC21 WG01, Q48.1, Q48.2	87/12/07-11		Stirling	21 N 2021
JTC 1 SC21 WG03	88/02-29/03-04		Galthers- burg, MD	JTC 1 N 41
JTC 1 SC21 WG03	88/11/28/12-14		Sydney Australia	JTC 1 N 41
JTC 1 SC21 WG04	88-03-03/10		Tysons Cor. VA, USA	JTC 1 N 41
JTC 1 SC21 WG04	88/11/28/12-14		Sydney Australia	JTC 1 N 41
JTC 1 SC21 WG04	88/03/02-10	OMNICOM	Tysons Cor. VA, USA	21 N 1901
JTC 1 SC21 WG04 Direc. CCITT	87/11/09-18		Gloucester	21 N 2156
JTC 1 SC21 SWG Reg. Auths.	87/11/09-18	BSI	London	21 N 1919
JTC 1 SC21 WG04 Manag. Info. Services	88/01			
JTC 1 SC21 WGO5	88/02/29/03-08		Crystal City, VA	JTC 1 N 41
JTC 1 SC21 WGO5	88/11-28/12-14		Sydney Australia	JTC 1 N 41
JTC 1 SC21 WGO5	88/03/01-08	COS	McLean, VA	21N 1901
JTC 1 SC21 WGO5 OSCRL	87/12/3-6		Dublin	21/5 N 275
JTC 1 SC21 WGO5 POSIX/ OSCRL SC 22/ WG n	88/01		Hagen or London	21 N 2082
JTC 1 SC21 WGO5 VT Basic Class	88/02/22-28		New Orleans	21 N 2082
JTC 1 SC21 WGO6	88/02/29/03/08		Wash. DC	JTC 1 N 41
JTC 1 SC21 WGO6	88/11/28/12-14		Sydney Australia	JTC 1 N 41
JTC 1 SC21 WGO6	88/02/29- 88/03/08	Quality Inn Capital Hill	Wash. DC	21 N 2082
JTC 1 SC21 WGO6 ASN.1	87/11/12-14		Gloucester	21 N 1989

53

<u>Meeting</u>	<u>YeMonDay</u>	<u>Place</u>	<u>City</u>	<u>Ref</u>
JTC 1 SC21 WGO8 Reg.Auth.	87/11/16-20	AFNOR	Paris	21 N 1989
JTC 1 SC21 WGO8 Sess. LOTOS	87/Nov/Dec	CNICE	Paris	21 N 1989
JTC 1 SC21 WGO8 Unlimited	88/02/22-26	ANSI	NYC	21 N 1989

SC.22 - APPLICATION SYSTEMS ENVIRONMENTS AND PROGRAMMING LANGUAGES

JTC 1 SC22 Plenary	89/09			
JTC 1 SC22 AG	88/10		Tokyo	JTC 1 N 42
JTC 1 SC22 WG02	88/02/02-05		Amsterdam	JTC 1 N 42
JTC 1 SC22 WG04	88/10/03		London	JTC 1 N 42
JTC 1 SC22 WG09	87/12/07-08		Boston	
JTC 1 SC22 WG12	88/04/06-08		Italy	
JTC 1 SC22 WG13	88/01/11-15	Nice	France	
JTC 1 SC22 WG14	87/11/11-12		Amsterdam	
JTC 1 SC22 WG13	88/01/11-15	Nice	France	JTC 1 N 42
JTC 1 SC22 WG16	88/02/24-25	AFNOR	Paris	JTC 1 N 42

SC.23 - OPTICAL DIGITAL DATA DISKS

JTC 1 SC23 Plenary	88/11/29-12-02		Amsterdam Netherlands	23 N 170
--------------------	----------------	--	--------------------------	----------

SC.24 - COMPUTER GRAPHICS

JTC 1 SC24 Plenary	87/12/01-03	DN	Berlin	24 N 1
JTC 1 SC24 Lang. Binding	87/11/3-10	Stuttgart	FRG	24 N 3

SC.25 - INFORMATION TECHNOLOGY EQUIPMENT

JTC 1 SC25	88/05/2-5		Stockholm	JTC 1 N 45
JTC 1 SC25 WG P&R	87/12/15		Munich	JTC 1 N 45
JTC 1 SC25 WG01	88/04	Tentative		JTC 1 N 45



	1988	1989	1990
New Year's Day	Friday/January 1	Sunday/January 1	Monday/January 1
Martin Luther King Jr.'s Birthday	Friday/January 15	Sunday/January 15	Monday/January 15
M.L. King Jr.'s Birthday Obs.	Monday/January 18	Monday/January 16	Monday/January 15
Abraham Lincoln's Birthday	Friday/February 12	Sunday/February 12	Monday/February 12
St. Valentine's Day	Sunday/February 14	Tuesday/February 14	Wednesday/February 14
Presidents' Day	Monday/February 15	Monday/February 20	Monday/February 19
Ash Wednesday	Wednesday/February 17	Wednesday/February 8	Wednesday/February 28
George Washington's Birthday	Monday/February 22	Wednesday/February 22	Thursday/February 22
St. Patrick's Day	Thursday/March 17	Friday/March 17	Saturday/March 17
Palm Sunday	Sunday/March 27	Sunday/March 19	Sunday/April 8
Good Friday	Friday/April 1	Friday/March 24	Friday/April 13
Passover	Saturday/April 2	Thursday/April 20	Tuesday/April 10
Easter	Sunday/April 3	Sunday/March 26	Sunday/April 15
Mother's Day	Sunday/May 8	Sunday/May 14	Sunday/May 13
Armed Forces Day	Saturday/May 21	Saturday/May 20	Saturday/May 19
Victoria Day (Canada)	Monday/May 23	Monday/May 22	Monday/May 21
Memorial Day Observance	Monday/May 30	Monday/May 29	Monday/May 28
Traditional Memorial Day	Monday/May 30	Tuesday/May 30	Wednesday/May 30
Flag Day	Tuesday/June 14	Wednesday/June 14	Thursday/June 14
Father's Day	Sunday/June 19	Sunday/June 18	Sunday/June 17
Dominion Day (Canada)	Friday/July 1	Saturday/July 1	Sunday/July 1
Independence Day	Monday/July 4	Tuesday/July 4	Wednesday/July 4
Labor Day	Monday/September 5	Monday/September 4	Monday/September 3
Rosh Hashanah	Monday/September 12	Saturday/September 30	Thursday/September 20
Yom Kippur	Wednesday/September 21	Monday/October 9	Saturday/September 29
Columbus Day Observance	Monday/October 10	Monday/October 9	Monday/October 8
Columbus Day	Wednesday/October 12	Thursday/October 12	Friday/October 12
Thanksgiving (Canada)	Monday/October 10	Monday/October 9	Monday/October 8
United Nations Day	Monday/October 24	Tuesday/October 24	Wednesday/October 24
Halloween	Monday/October 31	Tuesday/October 31	Wednesday/October 31
Election Day	Tuesday/November 8	Tuesday/November 7	Tuesday/November 6
Veterans Day	Friday/November 11	Saturday/November 11	Sunday/November 11
Thanksgiving	Thursday/November 24	Thursday/November 23	Thursday/November 22
Hanukkah	Sunday/December 4	Saturday/December 23	Wednesday/December 12
Christmas	Sunday/December 25	Monday/December 25	Tuesday/December 25

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

Doc. No.:

X3/88-015-X.I.S

JT/88-10

January 4, 1988

Date:

Project:

Ref. Doc.:

Reply to:

107(\*) JCA-9

54

**To: Members, X3, IAC, SPARC, SMC  
Officers X3TC's, TG's & SPARC/SG's  
Members, JTC1 TAG and AC**

**Subject: Transmittal of December 1987 SD-6, Membership & Officers**

Enclosed is the December 1987 SD-6. We have attached an SD-6 change form so that you may notify us of any changes as soon as possible. Please locate your name and check the address, phone number, and membership status that our records reflect. If there is an error please let us know about it by completing and returning the attached change form.

You will also notice that we have included membership lists for the JTC1 TAG and the JTC1 TAG Advisory Committee. These membership lists will be reflected in the X3/SD-6 until the next meeting of the JTC1 TAG and the next publication date of the Standing Documents. After February, we will issue independent JTC1 TAG Standing Documents.

We appreciate your help and any suggestions you may wish to offer.

Sincerely,



Catherine A. Kachurik  
Director, X3 Secretariat

**Enclosures: December 1987 SD-6  
December 1987 SD-6 Change Form**

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

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

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

# DECEMBER 1987 SD-6 CHANGE FORM

PLEASE MAKE THE FOLLOWING CHANGE(S) TO THE  
DECEMBER 1987 SD-6:

PAGE \_\_\_\_

CHANGE FROM:

-----  
-----  
-----

TO:

-----  
-----  
-----

**PLEASE RETURN THIS FORM TO:**

**CARESSA WILLIAMS  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, NW  
SUITE 500  
WASHINGTON, DC 20001**

**or call 202-737-8888 ext. 54**

X3/SD-6  
December 1987

54

ACCREDITED STANDARDS COMMITTEE\*  
X3-INFORMATION PROCESSING SYSTEMS

---

MEMBERSHIP & OFFICERS

---

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

Secretariat:  
Computer & Business Equipment Manufacturers Association



# X3 Standing Documents

54

This document is one of a series, developed by X3 and the X3 Secretariat, which provides a "data base" of information on Accredited Standards Committee X3 - Information Processing Systems. Each document is updated periodically on an individual basis.

The series is intended to serve several purposes:

- o To describe X3 and its program to inquirers
- o To inform committee members of the organization and operation of X3
- o To provide a system of orderly administration incorporating the procedures required by ANSI together with supplements approved by the X3 Secretariat, for the guidance of X3 officers, members, subgroups and the Secretariat staff.

The series of Standing Documents consists of the following:

X3/SD-0	Information Brochure - 1986
X3/SD-1	Master Plan - May 1987
X3/SD-2	Organization & Procedures - October 1985
X3/SD-3	Project Proposal Guide - May 1987
X3/SD-4	Projects Manual - December 1987
X3/SD-5	Standards Criteria - September 1984
X3/SD-6	Membership and Officers - December 1987
X3/SD-7	Meeting Schedule and Calendar - December 1987
X3/SD-9	Policy & Guidelines - September 1984 (to be issued)
X3/SD-10	X3 Subgroup Annual Report - October 1985

## X3/SD-6 Membership and Officers

X3/SD-6 provides the current, approved organizational membership of Accredited Standards Committee X3, together with the name, mailing address and telephone number of Principal and Alternate Representatives. Also listed are representatives of other organizations which have requested liaison status, together with ex-officio and individual Observers. Finally, the membership of the X3 Standing Committees and the officers of the Technical Committees are listed.

Corrections and suggestions for improvement will be welcomed, and should be addressed to:

X3 Secretariat/CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001-2178

# TABLE OF CONTENTS

54

## Accredited Standards Committee X3

Producer Members .....	1
Consumer Members .....	2
General Interest Members .....	3
Observers .....	4

## X3 Standing Committees

Secretariat Management Committee (SMC) .....	5
Standards Planning & Requirements Committee (X3/SPARC) .....	6

## Joint Technical Committee TAG Membership .....

JTC1 TAG Advisory Committee .....	10
-----------------------------------	----

## X3 Technical Committee Officers

X3 - Information Processing Systems .....	11
A - Character Recognition .....	11
B - Media .....	12
H & J - Programming Languages.....	12-13
K - Documentation .....	14
L - Data Representation .....	14
S - Data Communication .....	14
T & V - Systems Technology .....	15

Technical Committee Officer Appointment Dates.....	16-18
--	-------

Alphabetical Listing of Officers' Mailing Addresses .....	19-23
---	-------

SPARC & IAC Liaisons .....	24
----------------------------	----

## X3 Organizational Charts

General .....	25
Detailed .....	26

## ISO/IEC/JTC1

International Organizational Chart.....	27
Names and Addresses of Chairmen of U.S. Held Secretariats for ISO/IEC/JTC1 Subcommittees.....	28
Names and Addresses of Working Group Convenors.....	28

## X3 MEMBERSHIP

**CHAIRMAN**  
RICHARD GIBSON  
AT&T  
ROOM 5A 211  
ROUTE 202 & 206N  
BEDMINSTER NJ 07921  
201-234-3795

**VICE CHAIRMAN**  
DONALD C. LOUGHRY  
HEWLETT-PACKARD  
INFO. NETWORKS DIVISION  
19420 HOMESTEAD RD. MS 43UX  
CUPERTINO CA 95014-0606  
408-257-7000 X2454

**ADMIN. SECRETARY**  
CATHERINE A. KACHURIK  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

**RECORDING SECRETARY**  
GWENDY J. PHILLIPS  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

# PRODUCERS

54

**3M COMPANY**  
PAUL D. JAHNKE (P)  
3M COMPANY  
3M CENTER, BLDG. 236-GL-19  
ST. PAUL MN 55144  
612/736-0117

**AMP INCORPORATED**  
EDWARD KELLY (P)  
AMP INCORPORATED  
M/S 210-01  
P. O. BOX 3608  
HARRISBURG PA 17105-3608  
717-561-6153

RONALD LLOYD (A)  
AMP INCORPORATED  
P.O. BOX 3608, M/S 210-01  
HARRISBURG, PA 17105  
717-564-0100

**APPLE COMPUTER, INC.**  
KARL KIMBALL (P)  
APPLE COMPUTER, INC.  
20525 MARIANI AVENUE M/S 22Y  
CUPERTINO, CA 95014  
408-973-2403

MICHAEL J. LAWLER (A)  
APPLE COMPUTER, INC.  
20525 MARIANI AVENUE  
CUPERTINO, CA 95014  
408-973-4671

**AT&T**  
PAUL D. BARTOLI (P)  
AT&T  
CRAWFORD CORNERS ROAD  
ROOM 1M-311  
HOLMDEL, NJ 07733  
201-949-5965

THOMAS F. FROST (A)  
AT&T  
ROUTE 202-206 NORTH  
ROOM 5A210  
BEDMINSTER, NJ 07921  
201-234-8750

**CONTROL DATA CORPORATION**  
ERNEST L. FOGLE (P)  
CONTROL DATA CORPORATION  
901 EAST 78TH STREET, BMW03M  
BLOOMINGTON MN 55420-1334  
612-853-4080

KEITH A. LUCKE (A)  
CONTROL DATA CORPORATION  
901 EAST 78TH STREET, BMW03M  
BLOOMINGTON MN 55420-1334  
612/853-4380

**DATA GENERAL CORPORATION**  
LEE SCHILLER (P)  
DATA GENERAL CORPORATION  
62 T. W. ALEXANDER DRIVE  
RES. TRIANGLE PK., NC 27709  
919-248-5807

LYMAN CHAPIN (A)  
DATA GENERAL CORPORATION  
M/S D112  
4400 COMPUTER DRIVE  
WESTBORO, MA 01580  
617-870-6056

**DIGITAL EQUIPMENT CORPORATION**  
GARY S. ROBINSON (P)  
DIGITAL EQUIPMENT CORPORATION  
146 MAIN STREET, ML012B/E51  
MAYNARD MA 01754-2572  
617/493-4094

DELBERT L. SHOEMAKER (A)  
DIGITAL EQUIPMENT CORPORATION  
1331 PENNSYLVANIA AVE. N.W.  
50TH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

**HEWLETT-PACKARD**  
DONALD C. LOUGHRY (P)  
HEWLETT-PACKARD  
INFORMATION NETWORKS DIVISION  
19420 HOMESTEAD RD. MS 43UX  
CUPERTINO CA 95014-0606  
408-257-7000 X2454

**HONEYWELL BULL**  
DAVID M. TAYLOR (P)  
HONEYWELL BULL  
3800 W. 80TH STREET  
MM70-407  
MINNEAPOLIS, MN 55431  
612-896-3790

**IBM CORPORATION**  
MARY ANNE GRAY (P)  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE NY 10571-2597  
914/697-7224

ROBERT H. FOLLETT (A)  
IBM CORPORATION  
643/RCTR/9E  
6705 ROCKLEDGE DRIVE  
BETHESDA, MD 20817  
301-564-2108

**MOORE BUSINESS FORMS**  
D. H. ODDY (P)  
MOORE BUSINESS FORMS  
MOORE RESEARCH DIVISION  
300 LANG BOULEVARD  
GRAND ISLAND, NY 14072-1697  
716/773-0378

**NCR CORPORATION**  
WILLIAM E. SNYDER (P)  
NCR CORPORATION  
WHQ-5E  
DAYTON, OH 45479  
513-445-1986

A. R. DANIELS (A)  
NCR CORPORATION  
WHQ-5E  
DAYTON OH 45479  
513/445-1310

**PRIME COMPUTER, INC.**  
STEPHEN AZARIAN (P)  
PRIME COMPUTER, INC.  
M/S 10B-10  
500 OLD CONNECTICUT PATH  
FRAMINGHAM, MA 01701  
617-879-2960 X3123

**SCIENTIFIC COMP. SYS. CORP.**  
JAMES A. BAKER (P)  
SCIENTIFIC COMP. SYS. CORP.  
C/O 131 AVENIDA DRIVE  
BERKELEY, CA 94708  
415-548-3557

**UNISYS**  
MARVIN W. BASS (P)  
UNISYS  
M/S 2G4, P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215/542-3319

JEAN G. SMITH (A)  
UNISYS  
M/S E11-121  
P.O. BOX 500  
BLUE BELL, PA 19424-0001

**WANG LABORATORIES, INC.**  
I. J. CINECOE (P)  
WANG LABORATORIES, INC.  
M/S 014 A1B  
ONE INDUSTRIAL AVENUE  
LOWELL, MA 01851-5161  
617-967-5514

SARAH WAGNER (A)  
WANG LABORATORIES, INC.  
M/S 1389  
ONE INDUSTRIAL AVENUE  
LOWELL, MA 01851-5161  
617-967-6366

**XEROX CORPORATION**  
ROY PIERCE (P)  
XEROX CORPORATION  
1301 RIDGEVIEW  
M/S 111  
LEWISVILLE TX 75067  
214-420-2804

CUBE

THOMAS EASTERDAY (P)  
CUBE  
10 W. 106TH STREET  
P.O. BOX 527  
INDIANAPOLIS, IN 46206  
317/846-4211

DONALD MILLER (A)  
CUBE  
WOLPOFF & ABRAMSON  
11140 ROCKVILLE PIKE  
ROCKVILLE MD 20852-3164  
301/468-0303 X 41

DECUS

JAMES EBRIGHT (P)  
DECUS  
C/O SOFTWARE RESULTS CORP.  
2807 SILVER DRIVE  
COLUMBUS, OH 43211-1081  
614-267-2203

EASTMAN KODAK

GARY HAINES (P)  
EASTMAN KODAK  
PHOTOGRAPHIC TECHNOLOGY DIV.  
1700 DEWEY AVENUE, BLDG. 69  
ROCHESTER, NY 14650  
716-477-3597

CHARLETON C. BARD (A)  
EASTMAN KODAK  
C/O 74 CORNWALL LANE  
ROCHESTER, NY 14617  
716/722-5432

GENERAL ELECTRIC COMPANY

RICHARD W. SIGNOR (P)  
GENERAL ELECTRIC COMPANY  
BLDG. 30EW  
1285 BOSTON AVENUE  
BRIDGEPORT, CT 06601-2385  
203-382-3610

WILLIAM R. KRUESI (A)  
GENERAL ELECTRIC COMPANY  
MAIL DROP W1E  
3135 EASTON TURNPIKE  
FAIRFIELD CT 06432-1008  
203/373-2402

GENERAL SERVICES ADMN.

WILLIAM C. RINEHULS (P)  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD VA 22153-2532  
202/566-1180

LARRY L. JACKSON (A)  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD, VA 22153-2532  
202/566-0194

GUIDE INTERNATIONAL

FRANK KIRSHENBAUM (P)  
GUIDE INTERNATIONAL  
AMERICAN MANAGEMENT SYSTEMS  
31 MEADOWOOD DRIVE  
JERICO NY 11753-2833  
212/618-0300

SANDRA SCHWARTZ ABRAHAM (A)  
GUIDE INTERNATIONAL  
25 HORNOR LANE  
PRINCETON, NJ 08540

LAWRENCE BERKELEY LABORATORY

DAVID F. STEVENS (P)  
LAWRENCE BERKELEY LABORATORY  
BUILDING 50B, ROOM 2258  
1 CYCLOTRON ROAD  
BERKELEY CA 94720  
415/486-7344

ROBERT L. FINK (A)  
LAWRENCE BERKELEY LABORATORY  
UCLBL  
M/S 50B-2258  
BERKELEY, CA 94720  
415-486-5692

MAP/TOP

JAMES D. CONVERSE (P)  
MAP/TOP  
C/O EASTMAN KODAK COMPANY  
1099 JAY STREET  
ROCHESTER, NY 14650  
716-464-5705

MIKE KAMINSKI (A)

MAP/TOP  
C/O G.M. ADVANCED ENGINEERING  
30300 MOUND ROAD  
WARREN, MI 48090-9040

NATIONAL COMMUNICATIONS SYSTEM

DENNIS BODSON (P)  
NATIONAL COMMUNICATIONS SYSTEM  
8TH & SO. COURTHOUSE ROAD  
ARLINGTON, VA 22204-2198  
202-692-2124

GEORGE W. WHITE (A)  
NATIONAL COMMUNICATIONS SYSTEM  
8TH & SO. COURTHOUSE ROAD  
ARLINGTON VA 22204-2198  
202/692-2124

RAILINC CORPORATION

MONCURE N. LYON (P)  
RAILINC CORPORATION  
50 F STREET, NW  
WASHINGTON, DC 20001  
202-639-5542

RECOGNITION TECH. USERS ASSN.

HERBERT F. SCHANTZ (P)  
RECOGNITION TECH. USERS ASSN.  
HLS ASSOCIATES  
1701 CREEKSIDE DRIVE, STE. 100  
SOUTHLAKE, TX 76092  
817-481-3062

G. W. WEITZEL (A)  
RECOGNITION TECH. USERS ASSN.  
GRAHAM MAGNETICS  
6625 INDUSTRIAL PARK BLVD.  
NO. RICHLAND HILLS TX 76118  
817/281-9450

SHARE, INC.

THOMAS B. STEEL (P)  
SHARE, INC.  
C/O HOFSPRA UNIVERSITY  
DEPARTMENT OF COMPUTER SCIENCE  
HEMPSTEAD, NY 11550  
516-560-5556

ROBERT P. RANNIE (A)  
SHARE, INC.  
DEPT. OF COMPUTER SCIENCES  
NORTHERN ILLINOIS UNIVERSITY  
DE KALB, IL 60115  
815-753-0423

TRAVELERS INSURANCE COMPANIES

JOSEPH T. BROPHY (P)  
TRAVELERS INSURANCE COMPANIES  
1 TOWER SQUARE  
HARTFORD CT 06115-1599  
203/277-3122

U.S. DEPT. OF DEFENSE

FRED VIRTUE (P)  
U.S. DEPT. OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON DC 20330-5190  
202/695-0499

BELKIS LEONG-HONG (A)  
U.S. DEPT. OF DEFENSE  
OASD (C), MS. IRMS.  
ROOM 1C 535 PENTAGON  
WASHINGTON, DC 20301-1100  
202/695-4470

VIM INCORPORATED

CHRIS TANNER (P)  
VIM INCORPORATED  
CHALK RIVER NUCLEAR LABS  
CHALK RIVER, ONT KOJ 1J0  
CANADA  
613-584-3311 X 4053

M. SPARKS (A)  
VIM INCORPORATED  
C/O UNISYS  
1500 PERIMETER PKWY. STE 400  
HUNTSVILLE, AL 35806-1686  
205-837-7610

WINTERGREEN INFORMATION SYS.

JOHN L. WHEELER (P)  
WINTERGREEN INFORMATION SYS.  
CARRIAGE HOUSE COMMONS  
SUITE 347, 159 WEST MAIN ST.  
WEBSTER, NY 14580  
716-671-4087



# GENERAL INTEREST

54

AICCP

THOMAS M. KURIHARA (P)  
2058 CARRHILL ROAD  
VIENNA, VA 22180  
202-366-9717

AMERICAN LIBRARY ASSOCIATION

PAUL E. PETERS (P)  
AMERICAN LIBRARY ASSOCIATION  
NEW YORK PUBLIC LIBRARY  
5TH AVE. & 42ND ST., RM. 213  
NEW YORK, NY 10018  
212-930-0720

AMERICAN NUCLEAR SOCIETY

GERALDINE C. MAIN (P)  
AMERICAN NUCLEAR SOCIETY  
BCS RICHLAND, INC.  
P. O. BOX 300  
RICHLAND WA 99352-0300  
509-376-2287

SALLY HARTZELL (A)

AMERICAN NUCLEAR SOCIETY  
C/O POWER COMPUTING COMPANY  
25 VAN NESS AVENUE, SUITE 550  
SAN FRANCISCO, CA 94102  
415-626-7273

DATA PROCESSING MNGMT.  
ASSOC.

WARD ARRINGTON (P)  
DATA PROCESSING MNGMT. ASSOC.  
241 SHORE DRIVE EAST  
MIAMI, FL 33133  
305-593-4015

WALLACE R. MCPHERSON, JR. (A)  
DATA PROCESSING MNGMT. ASSOC.  
ROB #3, ROOM 4682  
400 MARYLAND AVENUE, S.W.  
WASHINGTON, DC 20202  
202-245-0361

IEEE

SAVA I. SHERR (P)  
IEEE  
P.O. BOX 20757  
MIDTOWN STATION  
NEW YORK, NY 10129  
201-662-2029

H. WOOD (A)

IEEE  
C/O NATIONAL BUREAU OF STDS.  
TECHNOLOGY BLDG., ROOM B154  
GAITHERSBURG, MD 20899-0999  
301-975-3240

NATIONAL BUREAU OF STANDARDS

ROBERT E. ROUNTREE (P)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG MD 20899-0999  
301-975-2827

MICHAEL HOGAN (A)

NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A61  
GAITHERSBURG MD 20899  
301-975-2926

OMNICOM, INC.

HAROLD C. FOLTS (P)  
OMNICOM, INC.  
115 PARK STREET, S.E.  
VIENNA, VA 22180  
703/281-1135

CHERYL C. SLOBODIAN (A)

OMNICOM, INC.  
115 PARK STREET, S.E.  
VIENNA, VA 22180  
703-281-1135

VISA U.S.A.

JEAN T. MCKENNA (P)  
VISA U.S.A.  
P. O. BOX 8999  
SAN FRANCISCO CA 94128  
415-570-3422

PATTY GREENHALGH (A)

VISA U.S.A.  
P.O. BOX 8999  
SAN FRANCISCO, CA 94128  
415-570-3424

## X3 MEMBERSHIP

54

### EX-OFFICIO MEMBERS - WITHOUT VOTE:

CHAIRPERSON X3/SPARC  
CHAIRPERSON SMC  
CHAIRPERSON X3/SPC  
CHAIRPERSON X3/TC'S  
CHAIRPERSON JTC1 TAG  
CHAIRPERSON JTC1 TAG AC

### EX-OFFICIO OBSERVERS

MEMBERS OF SMC  
MEMBERS OF X3/SPARC  
MEMBERS OF X3/SPC  
OFFICERS X3/TC'S, SC'S & SPARC/SG'S

### CHAIRPERSON OF ANSI ISSB

JAMES H. BURROWS (L)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B160  
GAITHERSBURG, MD 20899-0999  
301-975-2822

### LIAISON OBSERVERS

#### AMERICAN NAT'L STANDARDS INSTITUTE

FRANCES SCHROTTER (L)  
ANSI  
1430 BROADWAY  
NEW YORK NY 10018-3308  
212-642-4934

#### CODASYL/PROGRAMMING LANGUAGE COMMITTEE

IAN PROKOP (L)  
MCGRAW-HILL, INC.  
29 HARTWELL AVE.  
LEXINGTON, MA 02173  
617/863-5100

#### ASC T1 - TELECOMMUNICATIONS

ALVIN LAI (L)  
EXCHANGE CARRIERS STDS. ASSOC.  
5430 GROSVENOR LANE  
BETHESDA, MD 20814-2122  
301-564-4505

#### X9 - FINANCIAL SERVICES

CYNTHIA L. FULLER (L)  
AMERICAN BANKERS ASSOCIATION  
1120 CONNECTICUT AVENUE, N. W.  
WASHINGTON DC 20036-3973  
202-663-5000

#### X12 - BUSINESS DATA INTERCHANGE

TOM JONES (L)  
WESTERN DATACOM  
5083 MARKET STREET  
YOUNGSTOWN, OH 44512  
216-788-6583

#### ANSC 739 - LIBRARY & INFORMATION SCIENCES AND RELATED PUBLISHING PRACTICES

RAY DENNENBERG (L)  
LIBRARY OF CONGRESS  
NETWORK DEVELOPMENT OFFICE  
WASHINGTON DC 20540-0001  
202/287-5894

#### HUMAN FACTORS SOCIETY

PAUL REED (L)  
HUMAN FACTORS SOCIETY  
C/O AT&T BELL LABORATORIES  
184 LIBERTY CORNER ROAD  
WARREN, NJ 07060  
201-580-5618

#### X3 LIAISON TO ANSI PLANNING PANEL ON INDUSTRIAL AUTOMATION

LEROY RODGERS (L)  
DIGITAL EQUIPMENT CORPORATION  
CONTINENTAL BLVD., MK1-2/L6  
MERRIMACK, NH 03054  
603-884-8318

#### X3 LIAISON TO IEEE COMPUTER SOCIETY STANDARDS ACTIVITIES BOARD

GARY S. ROBINSON (L)  
DIGITAL EQUIPMENT CORPORATION  
146 MAIN STREET, ML012B/E51  
MAYNARD MA 01754-2572  
617/493-4094

#### X3 LIAISON TO CCITT USNC

EDWARD KELLY (L)  
AMP INCORPORATED  
M/S 210-01  
P. O. BOX 3608  
HARRISBURG PA 17105-3608  
717-561-6153

RICHARD GIBSON (L)  
AT&T  
ROOM 5A 211  
ROUTE 202 & 206N  
BEDMINSTER NJ 07921  
201-234-3795

#### OBSERVERS

WALTER G. FREDRICKSON (O)  
HARRIS CORPORATION  
1025 WEST NASA BLDV. MS/14  
MELBOURNE, FL 32919  
305-727-9100

#### STUART M. GARLAND (O) AT&T

TELETYPE CORPORATION  
5555 WEST TOUHY AVENUE  
SKOKIE IL 60077-3235  
312/982-3596

PIERRE L'ALLIER (O)  
CONCURRENT COMPUTER CORP.  
227 BATH ROAD  
SLOUGH  
BERKSHIRE SL1 4AX ENGLAND

KENNETH MAGEL (O)  
ASSN. FOR COMPUTING MACHINERY  
300 MINARD  
NORTH DAKOTA STATE UNIVERSITY  
FARGO, ND 58105

GERARD A. RAINVILLE, JR. (O)  
NATIONAL SECURITY AGENCY  
P.O. BOX 11  
ANNAPOLIS JUNCTION, MD 20701  
301-776-1684

D. L. SEIGAL (O)  
AMERICAN EXPRESS  
TRAVEL RELATED SERVICES  
1647 E. MORTEN AVENUE  
PHOENIX, AZ 85020  
602-371-3637

SELMA ZINKER (O)  
TANDEM COMPUTERS  
CORPORATE INFO. CENTER  
19333 VALLCO PARKWAY  
CUPERTINO, CA 95014-2599  
408-725-6343

# SECRETARIAT MANAGEMENT COMMITTEE (SMC)

54

## CHAIRMAN

DELBERT L. SHOEMAKER (P)  
DIGITAL EQUIPMENT CORPORATION  
1331 PENNSYLVANIA AVE. N.W.  
SIXTH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

JANUARY 1989

## VICE CHAIRMAN

EDWARD KELLY (P)  
AMP INCORPORATED  
M/S 210-01  
P. O. BOX 3608  
HARRISBURG PA 17105-3608  
717-561-6153

JANUARY 1989

## SECRETARY

PATRICIA A. STEINER  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

CHARLETON C. BARD (P)  
EASTMAN KODAK  
C/O 74 CORNWALL LANE  
ROCHESTER, NY 14617  
716/722-5432

JANUARY 1989

MARVIN W. BASS (P)  
UNISYS  
M/S 2G4  
P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215/542-3319

JANUARY 1988

MARY ANNE GRAY (P)  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE NY 10577-2597  
914/697-7224

JANUARY 1988

BELKIS LEONG-HONG (P)  
U.S. DEPT. OF DEFENSE  
OASD (C), MS, IRMS.  
ROOM 1C 535 PENTAGON  
WASHINGTON, DC 20301-1100  
202/695-4470

JANUARY 1990

PAUL E. PETERS (P)  
AMERICAN LIBRARY ASSOCIATION  
NEW YORK PUBLIC LIBRARY  
5TH AVE. & 42ND ST., RM. 213  
NEW YORK, NY 10018  
212-930-0720

JANUARY 1990

WILLIAM C. RINEHULS (P)  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD VA 22153-2532  
202/566-1180

JANUARY 1990

THOMAS B. STEEL (P)  
SHARE, INC.  
C/O HOFSPRA UNIVERSITY  
DEPARTMENT OF COMPUTER SCIENCE  
HEMPSTEAD, NY 11550  
516-560-5556

JANUARY 1988

## EX-OFFICIO

RICHARD GIBSON  
AT&T  
ROOM 5A-211  
ROUTE 202 & 206N  
BEDMINSTER, NJ 07921  
201-234-3795

GARY HAINES  
EASTMAN KODAK  
PHOTO. TECH. DIV.  
1700 DEWEY AVENUE  
BUILDING 69  
ROCHESTER, NY 14650  
716-477-3597

CATHERINE A. KACHURIK  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

WILLIAM HANRAHAN  
CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

P. 377

# STANDARDS PLANNING AND REQUIREMENTS COMMITTEE (SPARC)

54

## CHAIRMAN

WILLIAM C. RINEHULS  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD VA 22153-2532  
202/566-1180

## VICE CHAIRMAN

LEROY RODGERS (P)  
DIGITAL EQUIPMENT CORPORATION  
CONTINENTAL BLVD., MK1-2/L6  
MERRIMACK, NH 03054  
603-884-8318

## SECRETARY

GWENDY J. PHILLIPS  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

MARVIN W. BASS (P)  
UNISYS  
M/S 2G4  
P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215/542-3319

MARGARET K. BUTLER (P)  
ARGONNE NATIONAL LABORATORY  
9700 SOUTH CASS AVENUE  
BUILDING 201  
ARGONNE, IL 60439-4801  
312/972-7172

THOMAS EASTERDAY (P)  
CUBE  
10 W. 106TH STREET  
P.O. BOX 527  
INDIANAPOLIS, IN 46206  
317-846-4211

ERNEST L. FOGLE (P)  
CONTROL DATA CORPORATION  
901 EAST 78TH STREET  
ROOM BMW03M  
BLOOMINGTON, MN 55420  
612/853-6937

ROBERT H. FOLLETT (P)  
IBM CORPORATION  
643/RCTR/9E  
6705 ROCKLEDGE DRIVE  
BETHESDA, MD 20817  
301-564-2108

MARY ANNE GRAY (A)  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE NY 10577-2597  
914/697-7224

GARY HAINES (P)  
EASTMAN KODAK  
PHOTOGRAPHIC TECHNOLOGY DIV.  
1700 DEWEY AVENUE, BLDG. 69  
ROCHESTER, NY 14650  
716-477-3597

KARL KIMBALL (P)  
APPLE COMPUTER, INC.  
20525 MARIANI AVENUE  
CUPERTINO, CA 95014  
408-973-4671

THOMAS M. KURIHARA (P)  
2058 CARRHILL ROAD  
VIENNA, VA 22180  
202-366-9717

WILLIAM P. LAPLANT, JR. (P)  
P.O. BOX 2130  
ARLINGTON, VA 22202-0130  
301-763-3905

KEITH A. LUCKE (A)  
CONTROL DATA CORPORATION  
901 EAST 78TH STREET, BMW03M  
BLOOMINGTON MN 55420-1334  
612/853-4380

THOMAS J. MCNAMARA (P)  
41 SUMMIT AVENUE  
WOLLASTON, MA 02170  
617-479-8400

STEVE OKSALA (A)  
UNISYS  
M/S 2G4, P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215/542-3319

ROBERT E. ROUNTREE (A)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG MD 20899-0999  
301-975-2827

JAMES RYLAND (P)  
SOCIAL SECURITY ADMINISTRATION  
ROOM 530 COMPUTER CENTER  
6401 SECURITY BOULEVARD  
BALTIMORE MD 21235-0001  
301-594-6005

ROY G. SALTMAN (P)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A266  
GAITHERSBURG, MD 20899-0999  
301-975-3376

MARVIN SCHLENOFF (A)  
SOCIAL SECURITY ADMINISTRATION  
ROOM 530 COMPUTER CENTER  
6401 SECURITY BOULEVARD  
BALTIMORE, MD 21235-7999

DELBERT L. SHOEMAKER (A)  
DIGITAL EQUIPMENT CORPORATION  
1331 PENNSYLVANIA AVE. N.W.  
SIXTH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

DOROTHY STAPLETON (A)  
U.S. DEPT. OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON DC 20330-5190  
202/695-0499

FRED VIRTUE (P)  
U.S. DEPT. OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON DC 20330-5190  
202/695-0499

## STUDY GROUP CHAIRMEN

SOFTWARE SYSTEMS INTERFACE  
STUDY GROUP  
ROBERT H. FOLLETT (L)  
IBM CORPORATION  
643/RCTR/9E  
6705 ROCKLEDGE DRIVE  
BETHESDA, MD 20817  
301-564-2108

DATA BASE SYSTEMS STUDY GROUP  
EDWARD L. STULL (L)  
GTE GOVERNMENT SYSTEMS  
1700 RESEARCH BOULEVARD  
ROOM 3095  
ROCKVILLE, MD 20850  
301-294-8649

**PLEASE NOTE:**

Until the next meeting of JTC1 TAG and the next publication date of the Standing Documents, we will show the JTC1 TAG membership combined with the X3/Standing Document 6. After February, we will issue independent JTC1 TAG Standing Documents.

# JTC1 TAG MEMBERSHIP LIST

54

## CHAIRMAN

JOSEPH DEBLAST  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE, NY 10577-2597  
914-697-7280

## VICE CHAIRMAN

ROBERT E. ROUNTREE  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG, MD 20899-0999  
301-975-2827

## ADMINISTRATIVE SECRETARY

CATHERINE A. KACHURIK  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

## RECORDING SECRETARY

GWENDY J. PHILLIPS  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

## 3M COMPANY

PAUL D. JAHNKE (P)  
3M COMPANY  
3M CENTER  
BLDG. 236-GL-19  
ST. PAUL, MN 55144  
612-736-0117

## \*AICCP

THOMAS M. KURIHARA (P)  
AICCP  
C/O 2058 CARRHILL ROAD  
VIENNA, VA 22180  
202-366-9717

## ALLEN-BRADLEY

RONALD H. REIMER (P)  
ALLEN-BRADLEY  
1201 SOUTH SECOND STREET  
MILWAUKEE, WI 53204  
414-382-2227

## AMERICAN LIBRARY ASSOCIATION

PAUL E. PETERS (P)  
AMERICAN LIBRARY ASSOCIATION  
NEW YORK PUBLIC LIBRARY  
5TH AVENUE & 42ND ST., RM. 213  
NEW YORK, NY 10018  
212-930-0720

## AMERICAN NUCLEAR SOCIETY

GERALDINE C. MAIN (P)  
AMERICAN NUCLEAR SOCIETY  
BCS RICHLAND, INC.  
P.O. BOX 300  
RICHLAND, WA 99352-0300  
509-376-2287

SALLY HARTZELL (A)  
AMERICAN NUCLEAR SOCIETY  
C/O POWER COMPUTING COMPANY  
25 VAN NESS AVENUE, STE. 550  
SAN FRANCISCO, CA 94102  
415-626-7273

## AMP INCORPORATED

EDWARD KELLY (P)  
AMP INCORPORATED  
M/S 210-01  
P.O. BOX 3608  
HARRISBURG, PA 17105-3608  
717-561-6153

## RONALD LLOYD (A)

AMP INCORPORATED  
P.O. BOX 3608  
M/S 210-01  
HARRISBURG, PA 17105  
717-564-0100

## APPLE COMPUTER, INC.

KARL KIMBALL (P)  
APPLE COMPUTER, INC.  
20525 MARIANI AVENUE  
M/S 22Y  
CUPERTINO, CA 95014  
408-973-2403

## MICHAEL J. LAWLER (A)

APPLE COMPUTER, INC.  
20525 MARIANI AVENUE  
CUPERTINO, CA 95014  
408-973-4671

## AT&T

PAUL D. BARTOLI (P)  
AT&T  
CRAWFORD CORNERS ROAD  
ROOM 1M-311  
HOLMDEL, NJ 07733  
201-949-5965

## THOMAS F. FROST (A)

AT&T  
ROUTE 202-206 NORTH  
ROOM 5A210  
BEDMINSTER, NJ 07921  
201-234-8750

## CONTROL DATA CORPORATION

CHARLES E. COOPER (P)  
CONTROL DATA CORPORATION  
901 EAST 78TH STREET, BMW03M  
BLOOMINGTON, MN 55420-1334  
612-853-4080

## KEITH A. LUCKE (A)

CONTROL DATA CORPORATION  
901 EAST 78TH STREET, BMW03M  
BLOOMINGTON, MN 55420-1334  
612-853-4380

## CUBE

THOMAS EASTERDAY (P)  
CUBE  
10 W. 106TH STREET  
P.O. BOX 527  
INDIANAPOLIS, IN 46206  
317-846-4211

## DONALD MILLER (A)

CUBE  
WOLPOFF & ABRAMSON  
11140 ROCKVILLE PIKE  
ROCKVILLE, MD 20852-3164  
301-468-0303 X41

## DATA GENERAL CORPORATION

LYMAN CHAPIN (P)  
DATA GENERAL CORPORATION  
M/S D112  
4400 COMPUTER DRIVE  
WESTBORO, MA 01580  
617-870-6056

## LEE SCHILLER (A)

DATA GENERAL CORPORATION  
62 T. W. ALEXANDER DRIVE  
RES. TRIANGLE PK., NC 27709  
919-248-5807

## DATA PROCESSING MNGMT. ASSOC.

WARD ARRINGTON (P)  
DATA PROCESSING MNGMT. ASSOC.  
241 SHORE DRIVE EAST  
MIAMI, FL 33133  
305-593-4015

## WALLACE R. MCPHERSON, JR. (A)

DATA PROCESSING MNGMT. ASSOC.  
ROB #3, ROOM 4682  
400 MARYLAND AVENUE, S.W.  
WASHINGTON, DC 20202  
202-245-0361

## DECUS

JAMES EBRIGHT (P)  
DECUS  
C/O SOFTWARE RESULTS CORP.  
2887 SILVER DIRVE  
COLUMBUS, OH 43211-1081  
614-267-2203

## DIGITAL EQUIPMENT CORPORATION

GARY S. ROBINSON (P)  
DIGITAL EQUIPMENT CORPORATION  
146 MAIN STREET  
M/S ML012B/E51  
MAYNARD, MA 01754-2572  
617-493-4094

DELBERT L. SHOEMAKER (A)  
DIGITAL EQUIPMENT CORPORATION  
1331 PENNSYLVANIA AVE., N.W.  
SIXTH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

# JTC1 TAG MEMBERSHIP (CONTINUED)

54

## EASTMAN KODAK

GARY HAINES (P)  
EASTMAN KODAK  
PHOTOGRAPHIC TECHNOLOGY DIV.  
1700 DEWEY AVENUE, BLDG. 69  
ROCHESTER, NY 14650  
716-477-3597

CHARLES C. BARD (A)  
EASTMAN KODAK  
C/O 74 CORNWALL LANE  
ROCHESTER, NY 14617  
716-722-5432

## ELECTRONIC INDUSTRIES ASSOCIATION

ED KELLY (P)  
AMP INCORPORATED  
M/S 210-01  
P.O. BOX 3608  
HARRISBURG, PA 17105-3608  
717-561-6153

OLE GOLUBIATNIKOV (A)  
GENERAL ELECTRIC COMPANY  
FARRELL ROAD, PLANT 1, RM. D-6  
SYRACUSE, NY 13221  
315-456-4744

## EXCHANGE CARRIERS STDS. ASSOC.

MARTIN T. SULLIVAN (P)  
EXCHANGE CARRIERS STDS. ASSOC.  
C/O BELLCORE  
331 NEWMAN SPRINGS ROAD  
BOX 7020, ROOM 1F-321  
RED BANK, NJ 07701-7020  
201-758-2233

ALVIN LAI (A)  
EXCHANGE CARRIERS STDS. ASSOC.  
5430 GROSVENOR LANE  
BETHESDA, MD 20814-2122  
301-564-4505

## GENERAL ELECTRIC COMPANY

RICHARD W. SIGNOR (P)  
GENERAL ELECTRIC COMPANY  
BUILDING 30EW  
1285 BOSTON AVENUE  
BRIDGEPORT, CT 06601-2385  
203-382-3610

WILLIAM R. KRUEST (A)  
GENERAL ELECTRIC COMPANY  
MAIL DROP W1E  
3135 EASTON TURNPIKE  
FAIRFIELD, CT 06432-1008  
203-373-2402

## GENERAL SERVICES ADMIN.

WILLIAM C. RINEHULS (P)  
GENERAL SERVICES ADMIN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD, VA 22153-2532  
202-566-1180

LARRY L. JACKSON (A)  
GENERAL SERVICES ADMIN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD, VA 22153-02532  
202-566-0194

GUIDE INTERNATIONAL  
FRANK KIRSCHENBAUM (P)  
GUIDE INTERNATIONAL  
AMERICAN MANAGEMENT SYSTEMS  
31 MEADOWOOD DRIVE  
JERICHO, NY 11753-2833  
212-618-03000

SANDRA SCHWARTZ ABRHAM (A)  
GUIDE INTERNATIONAL  
25 HONOR LANE  
PRINCETON, NJ 08540

## HEWLETT-PACKARD

DONALD C. LOUGHRY (P)  
HEWLETT-PACKARD  
INFORMATION NETWORKS DIVISION  
19420 HOMESTEAD RD., M/S 43UX  
CUPERTINO, CA 95014-0606  
408-257-7000

## HONEYWELL BULL

DAVID M. TAYLOR (P)  
HONEYWELL BULL  
3800 W. 80TH STREET  
M/S MM70-407  
MINNEAPOLIS, MN 55431  
612-896-3790

## HUMAN FACTORS SOCIETY

PAUL REED (P)  
HUMAN FACTORS SOCIETY  
C/O AT&T BELL LABORATORIES  
184 LIBERTY CORNER ROAD  
WARREN, NJ 07060  
201-580-5618

## IBM CORPORATION

MARY ANNE GRAY (P)  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE, NY 10577-2597  
914-697-7224

## ROBERT H. FOLLETT (A)

IBM CORPORATION  
643/RCTR/9E  
6705 ROCKLEDGE DRIVE  
BETHESDA, MD 20817  
301-564-2108

## IEEE

SAVA I. SHERR (P)  
IEEE  
C/O P. O. BOX 20757  
MIDTOWN STATION  
NEW YORK, NY 10129  
201-662-2029

## ANDREW W. SALEM (A)

IEEE  
345 EAST 47TH STREET  
NEW YORK, NY 10017  
212-705-7966

## HELEN M. WOOD (A2)

IEEE  
C/O NATIONAL BUREAU OF STDS.  
B154 TECHNOLOGY  
GAITHERSBURG, MD 20899  
301-975-3240

## LASERDRIVE LTD.

ALAN EBRIGHT (P)  
LASERDRIVE LTD.  
1101 SPACE PARK DRIVE  
SANTA CLARA, CA 95054  
408-970-3600

## LAWRENCE BERKELEY LABORATORY

DAVID F. STEVENS (P)  
LAWRENCE BERKELEY LABORATORY  
BUILDING 50B, ROOM 2258  
1 CYCLOTRON ROAD  
BERKELEY, CA 94720  
415-486-7344

ROBERT L. FINK (A)  
LAWRENCE BERKELEY LABORATORY  
UCLBL, M/S 50B-2258  
BERKELEY, CA 94720  
415-486-9892

## MAP/TOP

JAMES D. CONVERSE (P)  
MAP/TOP  
C/O EASTMAN KODAK COMPANY  
1099 JAY STREET  
ROCHESTER, NY 14650  
716-464-5705

## MIKE KAMINSKI (A)

MAP/TOP  
C/O G. M. ADVANCED ENGINEERING  
30300 MOUND ROAD  
WARREN, MI 48090-9040

## MOORE BUSINESS FORMS

D. H. ODDY (P)  
MOORE BUSINESS FORMS  
MOORE RESEARCH DIVISION  
300 LANG BOULEVARD  
GRAND ISLAND, NY 14072-1697  
716-773-0378

## NATIONAL BUREAU OF STANDARDS

ROBERT E. ROUNTREE (P)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG, MD 20899-0999  
301-975-2827

## MICHAEL HOGAN (A)

NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A61  
GAITHERSBURG, MD 20899  
301-975-2926

## NATIONAL COMMUNICATIONS SYST

DENNIS BODSON (P)  
NATIONAL COMMUNICATIONS SYSTEM  
8TH & SO. COURTHOUSE ROAD  
ARLINGTON, VA 22204-2198  
202-692-2124

## DONALD WILSON (A)

NATIONAL COMMUNICATIONS SYSTEM  
8TH & SOUTH COURTHOUSE ROAD  
ARLINGTON, VA 22204-2198

## NCR CORPORATION

WILLIAM E. SNYDER (P)  
NCR CORPORATION  
WHQ-5E  
DAYTON, OH 45479  
513-445-1986

P. 381

# JTC1 TAG MEMBERSHIP (CONTINUED)

54

A. R. DANIELS (A)  
NCR CORPORATION  
WHQ-5E  
DAYTON, OH 45479  
513-445-1310

NISO  
PAT HARRIS (P)  
NISO  
NATIONAL BUREAU OF STANDARDS  
ADMIN. 101-LIBRARY, E-106  
GAITHERSBURG, MD 20899

OMNICOM, INC.  
HAROLD C. FOLTS (P)  
OMNICOM, INC.  
115 PARK STREET, S.E.  
VIENNA, VA 22180

CATHERINE HOWELLS (A)  
OMNICOM, INC.  
501 CHURCH STREET, N.E.  
SUITE 304  
VIENNA, VA 22180  
703-281-1135

CHERYL C. SLOBODIAN (A2)  
OMNICOM, INC.  
115 PARK STREET, S.E.  
VIENNA, VA 22180  
703-281-1135

PRIME COMPUTER, INC.  
STEPHEN AZARIAN (P)  
PRIME COMPUTER, INC.  
M/S 10B-10  
500 OLD CONNECTICUT PATH  
FRAMINGHAM, MA 01701  
617-879-2960 X3123

RAILINC CORPORATION  
MONCURE N. LYON (P)  
RAILINC CORPORATION  
50 F STREET, N.W.  
WASHINGTON, DC 20001  
202-639-5542

RECOGNITION TECH. USERS ASSN.  
HERBERT F. SCHANTZ (P)  
RECOGNITION TECH. USERS ASSN.  
HLS ASSOCIATES  
1701 CREEKSIDE DRIVE, STE. 100  
SOUTHLAKE, TX 76092  
817-481-3062

G. W. WETZEL (A)  
RECOGNITION TECH. USERS ASSN.  
GRAHAM MAGENTICS  
6625 INDUSTRIAL PARK BLVD.  
NO. RICHLAND HILLS, TX 76118  
817-281-9450

SCIENTIFIC COMP. SYS. CORP.  
JAMES A. BAKER (P)  
SCIENTIFIC COMP. SYS. CORP.  
C/O 131 AVENIDA DRIVE  
BERKELEY, CA 94708  
415-548-3557

SHARE, INC.  
THOMAS B. STEEL (P)  
SHARE, INC.  
C/O HOPSPRA UNIVERSITY  
DEPARTMENT OF COMPUTER SCIENCE  
HEMPSTEAD, NY 11550  
516-560-5556

ROBERT P. RANNIE (A)  
SHARE, INC.  
DEPT. OF COMPUTER SCIENCES  
NORTHERN ILLINOIS UNIVERSITY  
DE KALB, IL 60115  
815-753-0423

TRAVELERS INSURANCE COMPANIES  
JOSEPH T. BROPHY (P)  
TRAVELERS INSURANCE COMPANIES  
1 TOWER SQUARE  
HARTFORD, CT 06115-1599  
203-277-3122

U.S. DEPARTMENT OF DEFENSE  
FRED VIRTUE (P)  
U.S. DEPARTMENT OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON, DC 20330-5190  
202-695-0499

FRANK KIRSCHNER (A)  
U.S. DEPARTMENT OF DEFENSE  
HQ USAF/SCTT  
WASHINGTON, DC 20330-5190

UNISYS  
MARVIN W. BASS (P)  
UNISYS  
M/S 2G4  
P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215-542-3319

JEAN G. SMITH (A)  
UNISYS  
M/S E11-121  
P. O. BOX 500  
BLUE BELL, PA 19424-0001

VIM INCORPORATED  
CHRIS TANNER (P)  
VIM INCORPORATED  
CHALK RIVER NUCLEAR LABS  
CHALK RIVER, ONT K0J1J0  
CANADA  
613-584-3311 X2865

MADLINE SPARKS (A)  
VIM INCORPORATED  
C/O UNISYS  
1500 PERIMETER PKWY., STE. 400  
HUNTSVILLE, AL 35806-1686  
205-837-7610

VISA U.S.A.  
JEAN T. MCKENNA (P)  
VISA U.S.A.  
P. O. BOX 8999  
SAN FRANCISCO, CA 94128  
415-570-3422

PATTY GREENHALGH (A)  
VISA U.S.A.  
P. O. BOX 8999  
SAN FRANCISCO, CA 94128  
415-570-3424

WANG LABORATORIES, INC.  
J. I. CINECOE (P)  
WANG LABORATORIES, INC.  
M/S 014 A1B  
ONE INDUSTRIAL AVENUE  
LOWELL, MA 01851-5161  
617-967-5514

SARAH WAGNER (A)  
WANG LABORATORIES, INC.  
M/S 1389  
ONE INDUSTRIAL AVENUE  
LOWELL, MA 01851-5161  
617-967-6366

WINTERGREEN INFORMATION SYS.  
JOHN L. WHEELER (P)  
WINTERGREEN INFORMATION SYS.  
CARRIAGE HOUSE COMMONS  
159 WEST MAIN STREET, STE. 347  
WEBSTER, NY 14580  
716-671-4087

XEROX CORPORATION  
ROY PIERCE (P)  
XEROX CORPORATION  
1301 RIDGEVIEW  
LOUISVILLE, TX 75067  
214-420-2804

## EX-OFFICIO & LIAISON

AMERICAN NATIONAL STANDARDS INSTITUTE  
FRANCES SCHROTTER  
ANSI  
1430 BROADWAY  
NEW YORK, NY 10018-3308  
212-642-4934



# JTC1 TAG ADVISORY COMMITTEE (JTC1 TAG AC)

54

## ACTING CHAIR

GARY S. ROBINSON (P)  
DIGITAL EQUIPMENT CORPORATION  
146 MAIN STREET, ML012B/E51  
MAYNARD MA 01754-2572  
617/493-4094

## SECRETARY

CATHERINE A. KACHURIK  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202/737-8888

CHARLETON C. BARD (A)  
EASTMAN KODAK  
C/O 74 CORNWALL LANE  
ROCHESTER, NY 14617  
716/772-5432

MARVIN W. BASS (A)  
UNISYS  
M/S 2G4  
P. O. BOX 500  
BLUE BELL, PA 19424-0001  
215/542-3319

JAMES CONVERSE (A)  
EASTMAN KODAK  
1099 JAY STREET  
ROCHESTER, NY 14630  
716-464-5705

RICHARD GIBSON (P)  
AT&T  
ROOM 5A 211  
ROUTE 202 & 206N  
BEDMINSTER NJ 07921  
201-234-3795

MARY ANNE GRAY (P)  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE NY 10577-2597  
914/697-7224

GARY HAINES (P)  
EASTMAN KODAK  
PHOTOGRAPHIC TECHNOLOGY DIV.  
1700 DEWEY AVENUE, BLDG. 69  
ROCHESTER, NY 14650  
716-477-3597

EDWARD KELLY (P)  
AMP INCORPORATED  
M/S 210-01  
P. O. BOX 3608  
HARRISBURG PA 17105-3608  
717-561-6153

KARL KIMBALL (P)  
APPLE COMPUTER, INC.  
20525 MARIANI AVENUE M/S 22Y  
CUPERTINO, CA 95014  
408-073-2403

FRANK KIRSHENBAUM (P)  
GUIDE INTERNATIONAL  
AMERICAN MANAGEMENT SYSTEMS  
31 MEADOWOOD DRIVE  
JERICHO NY 11753-2833  
212/618-0300

ROBERT E. ROUNTREE (P)  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG MD 20899-0999  
301-973-2827

DELBERT L. SHOEMAKER (A)  
DIGITAL EQUIPMENT CORPORATION  
1331 PENNSYLVANIA AVE. N.W.  
SIXTH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

DOROTHY STAPLETON (A)  
U.S. DEPT. OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON DC 20330-5190  
202/695-0499

FRED VIRTUE (P)  
U.S. DEPT. OF DEFENSE  
HQ USAF/SCTT  
ROOM 5C1083, PENTAGON  
WASHINGTON DC 20330-5190  
202/695-0499

SPARC CHAIRMAN  
WILLIAM C. RINEHULS (XO)  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD VA 22153-2532  
202/566-1180

ANSI  
FRANCES SCHROTTER (XO)  
ANSI  
1430 BROADWAY  
NEW YORK NY 10018-3308  
212-642-4934

47B TAG ADMINISTRATOR  
ROBERT DAVIS (XO)  
TAG ADMINISTRATOR  
47B  
22685 SUMMIT ROAD  
LOS GATOS, CA 95030

TC83 TAG ADMINISTRATOR  
SAVA I. SHERR (XO)  
P.O. BOX 20757  
MIDTOWN STATION  
NEW YORK, NY 10129  
201-662-2029

# X3 TECHNICAL COMMITTEE OFFICERS CHART

54

<u>X3TC</u>	<u>**CHAIR</u>	<u>**VICE CHAIR</u>	<u>**INT'L REP.</u>	<u>VOCAB. REP.</u>	<u>SECRETARY</u>	<u>***INT'L</u>
<u>X3 - INFORMATION PROCESSING SYSTEMS</u>						
	RICHARD GIBSON AT&T 201-234-3795	DON LOUGHRY HEWLETT-PACKARD 408-257-7000 Extension 2454	N/A	N/A	CATHIE KACHURIK X3 SECRETARIAT 202-737-8888	JTC1
<u>SPARC- STANDARDS PLANNING AND REQUIREMENTS COMMITTEE</u>						
	W.C. RINEHULS GEN. SER. ADMN. 202-566-1180	LEROY RODGERS DEC 603-884-8318	N/A	N/A	GWENDY PHILLIPS X3 SECRETARIAT 202-737-8888	
<u>SPARC STUDY GROUPS</u>						
<u>DBSSG</u> Database Sys. Study Group	EDWARD STULL GTE GVT. SYS. 301-294-8649	NANCY McDONALD COMP. TECH. PLNG. 813-968-2660	ELIZABETH FONG NBS 301-975-3250	EDWARD STULL GTE GVT. SYS. 301-294-8649	ELIZABETH FONG NBS 301-975-3250	SC21
<u>SSISG</u> Softwr.Sys. Intfce. Study Group	*BOB FOLLETT IBM 301-564-2108	VACANT	VACANT	VACANT	VACANT	
<u>SMC - SECRETARIAT MANAGEMENT COMMITTEE</u>						
	DEL SHOEMAKER DEC 202-383-5622	ED KELLY AMP 717-561-6153	N/A	N/A	PATTI STEINER X3 SECRETARIAT 202-737-8888	
<u>SMC - STRATEGIC PLANNING COMMITTEE</u>						
	*DONALD LOUGHRY HEWLETT-PACKARD 408-257-7000 Extension 2454	VACANT	N/A	N/A	CATHERINE A. KACHURIK X3 SECRETARIAT 202-737-8888	
<u>JTC1 TAG - TECHNICAL ADVISORY GROUP TO ANSI, MEMBER BODY OF JTC1, INFORMATION TECHNOLOGY</u>						
	JOE DEBLASI IBM CORP. 914-697-7280	ROBERT ROUNTREE NBS 301-975-2827	N/A	N/A	CATHIE KACHURIK X3 SECRETARIAT 202-737-8888	
<u>JTC1 TAG AC - JTC1 TAG ADVISORY COMMITTEE</u>						
	*GARY ROBINSON DEC 617-493-4094	VACANT	N/A	N/A	CATHIE KACHURIK X3 SECRETARIAT 202-737-8888	
<u>X3 SUBCOMMITTEES</u>						
<u>SC21 TAG</u>	*J. ASCHENBRENNER IBM CORP. 919-254-0299	JON BECKER UNISYS 215-341-4643	N/A	N/A	VACANT	SC21
<u>SC22 TAG</u>	BOB FOLLETT IBM 301-564-2108	*MABEL VICKERS NBS 301-975-3277	N/A	N/A	VACANT	SC22
<u>X3 TECHNICAL COMMITTEES</u>						
<u>A - RECOGNITION</u>						
<u>X3A1</u> OCR & MICR	CARL KNOEDEL STD.REG.CO 513-443-1072	ROBERT BLOSS UNISYS 313-451-4298	JOHN McDONNELL RECOG. EQUIP. 214-579-5800	CARL KNOEDEL STD. REG. CO. 513-443-1072	CARL KNOEDEL STD. REG. CO. 513-443-1072	
<u>X3A1.1</u> FONT DESIGN	PATRICK TRAGLIA IBM CORP. 607-752-3357	VACANT	N/A	N/A	VACANT	
<u>X3A1.2</u> OCR SUPPL. & FORMS	CARL KNOEDEL STD. REG. CO. 513-443-1072	VACANT	N/A	N/A	VACANT	
<u>X3A1.3</u> IMAGE DEF. & MEASUREMENT	C.E. BISS PHO. SCI. CORP. 716-265-1600	VACANT	N/A	N/A	VACANT	

NOTE: Where the SMC appointment ballot is not complete or volunteers have not submitted the necessary documentation, the officer position has been listed as ACTING (\*) or "VACANT"

\*\* SMC elected officer

\*\*\* Corresponding International Standards Committee

XYTC	**CHAIR	**VICE CHAIR	**INT'L REP.	VOCAB. REP.	SECRETARY	**INT'L
<b>B - MEDIA</b>						
X375 DIGITAL REC. TAPE	R. STEINWURFNER AT&T 201-386-7053	SAM CHEATHAM STOR. TECH. CORP. 303-673-3359	MICHAEL HOGAN NBS 301-975-2926	ARNOLD ROCCATI EG&G WASH. ANAL. 301-840-3277	VACANT	SC11/SC15
X386 INSTRUMENTA. TAPE	BILL POLAND NASA/GODDARD 301-286-8592	C. DON WRIGHT LOCKHEED 201-234-9484	STAN REYNOLDS SAND. NATL LABS. 505-844-3518	VACANT	VACANT	JTCL/WG2
X387 REC. DISKS	BILL CARLSON MAXTOR 408-942-1700	VACANT	BILL CARLSON MAXTOR 408-942-1700	PAUL JAHNKE 3M COMPANY 612-736-0117	VACANT	
X388 FLDK. DISK CARTRIDGES	JAMES BARNES BARNES ASSOC. 713-374-5674	WILLIAM PROCTOR MEMOREX CORP. 408-957-0684	MICHAEL HOGAN NBS 301-975-2926	RONALD YOUNG KEY 408-947-8700	CLOYD TOLBERT COMPAQ COMP. 713-370-0670	SC11/SC15
X388.1 TEK. FORMS FOR FDC'S	CLOYD TOLBERT COMPAQ COMP. 713-370-0670	VACANT	N/A	N/A	VACANT	SC11/SC15
X389 PAPER FORMS/LAYOUTS	DELMER ODDY MRE. BUS. FORMS 716-773-8378	VACANT	VACANT	VACANT	DURHAM SEELEY STD. REG. CO. 716-271-3400 X2407	
X3B10 CREDIT/ID CARDS	R. KITCHENER LOGICARD, SYS. 914-273-8734	ALICE DROOGAN MSTRCARD INT'L 212-649-5358	DAVID SEIGAL AMERICAN EXP. 602-371-3637	VACANT	JOHN STEARNS DATACARD 612-933-1223	SC17
X3B10.1 INTEGRATED CIRCUIT CARDS	VACANT	VACANT	N/A	N/A	JIM JOHNSON DATACARD 612-933-1223	SC17/WG4
X3B10.2 REV. OF X3.149	VACANT	VACANT	N/A	N/A	VACANT	SC17
X3B10.3 MEM. PHTS. REQ./SAVINGSBOOKS	VACANT	VACANT	N/A	N/A	VACANT	SC17
X3B10.4 OPT. ENCOD. CARD MEDIA	ROBERT CALLEN DREXLER TECH. 415-969-4428	VACANT	N/A	N/A	VACANT	SC17
X3B11 OPT. DIG. DATA DISKS	J. ZAJACZKOWSKI CHEROKEE DATA 303-449-1239	ROBERT BENDER PENTAX TEKNOI. 303-460-1610	MICHAEL DEESE CHEROKEE DATA 303-530-0618	J.A. WESTENBROEK DUPONT 302-999-5354	SHARON ORIEL DOW CHEMICAL 517-636-2173	SC23/SC15
<b>B &amp; J - PROGRAMMING LANGUAGES</b>						
X3B2 DATABASE	DONALD DEUTSCH G.E. 301-340-4580	CAROL D. JOYCE RELA. TECH. INC. 415-769-1400	LEONARD GALLAGHER NBS 301-975-3251	BERNARD KOCIS NARDAC 202-433-4390	MICHAEL GORMAN WHITEMARSH INFO. 301-249-1142	SC21/WG3
X3B3 COMPUTER GRA.	PETER BONO P.R. BONO ASSOC. 203-464-9350	JOHN MCCONNELL DEC 603-884-2285	JANET CHIN CHIN ASSOC. 415-843-9384	MADELEINE SPARKS UNISYS 205-837-7610	JAMES MICHENER APOLLO COMP. 617-256-6600 X5275	SC24
X3B3.1 PLIGS	DAVID BAILEY CALCOMP/SANDERS 603-885-8141	DEBBIE CAHN BOEING	N/A	N/A	VACANT	SC24
X3B3.2 FORMAL DESCR. VALID & TESTING.	*GEORGE CARSON GSC ASSOCIATES 213-978-9351	VACANT	N/A	N/A	VACANT	SC24
X3B3.3 VIRTUAL DEVICE INTERFACE	KARLA VECCHIET METHEUS CORP. 503-640-8000	THOMAS POWERS DEC 617-493-2704	N/A	N/A	VACANT	SC24
X3B3.4 LANGUAGE BINDING	MADELEINE SPARKS UNISYS 205-837-7610	G. CUTHBERT SOFTWARE TECH. 305-723-3999	N/A	N/A	DAVE LARSON HEWLETT-PACKARD 303-226-3800	SC24

54

X3TC	**CHAIR	**VICE CHAIR	**INT'L REP.	VOCAB. REP.	SECRETARY	***INT'L
<b>H &amp; J - PROGRAMMING LANGUAGES (CONTINUED)</b>						
X3B3.5 GKS	MARY MILLER LANDMARK GRAPH. 713-531-4080	*JOHN MCCORMELL DEC 603-884-2285	N/A	N/A	VACANT	SC24
X3B3.6 WINDOW MANAGEMENT	GEORGES GRINSTEIN UNIV. OF LOWELL 617-452-5000	VACANT	N/A	N/A	VACANT	SC24
X3B4 INFO. RESOURCE & DICTIONARY	ANTHONY WINKLER 703-848-2750	DAVID CARPENTER PANSOPHIC 312-357-5950	DAVID THOMAS MSP, INC. 617-863-5800	DON MCCAFFREY 703-569-2390	DON MCCAFFREY 703-569-2390	SC21
X3B4.1 IRDS REF. MODEL	VACANT	VACANT	N/A	N/A	VACANT	SC21
X3B4.2 IRDS EXTER. SOFTWARE INTERFC.	VACANT	VACANT	N/A	N/A	VACANT	SC21
X3J1 PL/I	JOHN KLENSIN MIT 617-253-1355	CHARLES NYLANDER DEC 603-881-2081	*K.R. LUND IBM CORP.	ARTHUR COSTON APPLD. INFO. SYS. 919-942-7801	VACANT	SC22
X3J1.3 GEN PURP. SUBSET	VACANT	VACANT	N/A	N/A	VACANT	SC22
X3J2 BASIC	JAMES HARLE U.S. NAVAL ACAD.	ANDREW KLOSSNER TEKTRONIX, INC.	VACANT	JOHN CUGINI NBS 301-975-3248	VACANT	SC22/WG8
X3J3 FORTRAN	JEANNE ADAMS NCR 303-497-1275	JERROLD WAGENER AMOCO PROD. CO. 918-664-3415	ANDREW JOHNSON PRIME 617-879-2960 X4045	KURT HIRCHERT SUPERCOMPUTING	J.K. REID AERE HARWELL	SC22/WG5
X3J4 COBOL	DON SCHRICHER WANG LABS. 617-967-7628	BRUCE SINCLAIR RYAN MCFARLAND 512-343-1010	PEGGY BEARD NCR CORP. 619-693-5730	VICKI ECKELS TEXAS INST. 512-250-6512	PEGGY BEARD NCR CORP. 619-693-5730	SC22/WG4
X3J7 APT	WICKHAM LOH IBM CORP. 213-312-5926	C. W. WILSON MARTIN MARIETTA	TOM BRENGAN BOEING	VACANT	VACANT	ISO/TC184/
X3J7.1 PROCESSOR LANGS	VACANT	VACANT	N/A	N/A	VACANT	ISO/TC184/SC3
X3J7.2 POSTPRCSR LANG.	VACANT	VACANT	N/A	N/A	VACANT	ISO/TC184/SC3
X3J7.3 LATHE LANG.	VACANT	VACANT	N/A	N/A	VACANT	ISO/TC184/SC3
X3J7.4 ROBOTICS LANG.	VACANT	VACANT	N/A	N/A	VACANT	ISO/TC184/SC3
X3J9 PASCAL	K. ZEMROWSKI TRW 703-876-8056	MICHAEL HAGERTY RADIONICS 408-757-8877	PAULA SCHWARTZ RES. LIBR. GRP. 415-329-3534	WES MUNSIL APSE, INC. 602-962-6815	V. POLMARCZNY NCR CORP. 614-439-0516	SC22/WG2
X3J10 APL	VACANT	G. H. POSTER SYRACUSE UNIV. 315-423-4375	VACANT	PETER LUSTER 703-525-8001	VACANT	SC22/WG3
X3J11 C LANGUAGE	JIM BRODIE J. BRODIE ASSOC. 602-961-0032	*THOMAS PLUM PLUM HALL INC. 609-927-3770	P.J. FLAUGHER WHITESMITHS, LTD. 617-692-7800	ANDREW JOHNSON PRIME 617-879-2960 4045	P.J. FLAUGHER WHITESMITHS, LTD. 617-692-7800	SC22/WG14
X3J12 DIBOL	ELI SEKLANKA TEC COMPUTER SYS. 617-964-3890	KENNETH LIDSTER D.I.S.C. INC. 916-635-7300	VACANT	K. SCHILLING MCBA 818-242-9600	KENNETH BEERS DEC 603-884-5537	
X3J13 COMMON LISP	*ROBERT MATHIS 703-425-5923	VACANT	VACANT	VACANT	VACANT	SC22/WG16
X3J14 FORTRAN	*VACANT	RAY DUNCAN LAB MICROSYS 213-306-7412	VACANT	VACANT	VACANT	

P. 386

SYTC	**CHAIR	**VICE CHAIR	**INT'L REP.	VOCAB. REP.	SECRETARY	**INT'L
<b>K - DOCUMENTATION</b>						
X3K1	JOHN HACKNEY INTSTATE. ELEC. 714-635-7210	VACANT	RICHARD WERLING 703-820-4034	VACANT	VACANT	
X3K5	JOHN WOOD IBM CORP. 919-254-0182	HELMUT THIESS 202-265-9439	RICHARD BATEY UNISYS 215-542-2709	N/A	ROY MULLINAX HOUS. & URBAN 703-755-5607	SC1
<b>L - DATA REPRESENTATION</b>						
X3L2	DONALD THELEN AT&T TECH. 201-644-3728	VACANT	MACKLIN BISHOP IBM CORP. 512-823-2005	JOHN RUSSELL CONTROL DATA 612-853-5414	RICHARD MYERS UNISYS	SC2
X3L2.1	DONALD THELEN AT&T 201-644-3728	VACANT	N/A	N/A	VACANT	SC2
X3L2.2	VACANT	VACANT	N/A	N/A	VACANT	SC2
X3L2.3	*VACANT	VACANT	N/A	N/A	VACANT	SC2/WG2
X3L8	BILL KERWORTHY DEPT. OF DEFENSE 202-694-3361	DUANE MARQUIS NTIA 301-224-4300	KAREN KIRKBRIDE 202-746-0797	BILL KERWORTHY DEPT. OF DEFENSE 202-694-3361	VACANT	SC14
X3L8.4	HENRY TOM NBS 301-975-3271	VACANT	N/A	N/A	VACANT	SC14
X3L8.5	VACANT	VACANT	N/A	N/A	VACANT	SC14
X3L8.6	KAREN KIRKBRIDE 202-746-0797	ROGER PAYNE U.S. GEO SURVEY 703-648-4544	N/A	N/A	VACANT	SC14
X3L8.7	DUANE MARQUIS NTIA 301-224-4300	MARY SUPPERS DEF. INTEL. AGY. 202-373-3007	N/A	N/A	VACANT	SC14
<b>S - DATA COMMUNICATION</b>						
X3S3	W. F. EMMONS IBM CORP. 919-254-0294	*VACANT	JOHN WHEELER WINTERGREEN I.S. 716-671-4087	VACANT	VACANT	SC6
X3S3.1	W. F. EMMONS IBM CORP. 919-254-0294	VACANT	N/A	N/A	VACANT	SC6
X3S3.2	VACANT	VACANT	N/A	N/A	VACANT	SC6
X3S3.3	LYNN CHAPIN DATA GENERAL 617-870-6056	DAVID FISCIHELLO UNISYS 215-341-4642	N/A	N/A	PAUL TSUCHIYA MITRE CORP. 703-883-7352	SC6/WG2
X3S3.4	*DAVID CARLSON AT&T 201-949-7503	VACANT	N/A	N/A	VACANT	SC6
X3S3.5	NEAL SEITZ DEPT. OF COMM. 303-497-3106	VACANT	N/A	N/A	VACANT	SC6
X3S3.7	FRED BURG AT&T 201-949-0919	RANDY SPUSTA BELL COMM. RES.	N/A	N/A	J. PODVOJSKY MITRE CORP. 617-271-2155	SC6

X3TC	**CHAIR	**VICE CHAIR	**INT'L REP.	VOCAB. REP.	SECRETARY	**INT'L
<b>F &amp; V - SYSTEMS TECHNOLOGY</b>						
X3T1 DATA ENCRYPTION	STEPHEN LEVIN ITT DCD 201-284-4504	GEOFFREY TURNER BANK OF AMERICA 415-675-4204	ROBERT ELANDER IBM CORP. 914-385-6692	STEPHEN LEVIN ITT DCD 201-284-4504	VACANT	SC20
X3T2 DATA INTER- CHANGE	L. J. GALLAGHER NBS 301-975-3251	MIKE BLACKLEDGE SANDIA NAT'L. LAB. 505-846-6014	MAURICE SMITH ALLIED-BENDIX 816-997-3590	VACANT	VACANT	SC21/WG6 SC22/WG11
X3T5 OSI	JERROLD POLEY EDS CORP. 313-524-8416	J. ASCHENBRENNER IBM CORP. 919-254-0299	J. ASCHENBRENNER IBM CORP. 919-254-0299	VACANT	VACANT	SC21/WG6
X3T5.1 OSI ARCHT.	JOHN DAY CODEX CORP. 617-364-2000	JOHN GURZICK MCI 703-442-5363	N/A	N/A	VACANT	SC21/WG1
X3T5.4 OSI MGMT. PROTOCOLS	WILL COLLINS CODEX CORP. 617-364-2000	MARK KLERER AT&T 201-949-8645	N/A	N/A	VACANT	SC21/WG4
X3T5.5 APPLI. & PRES. LAYERS	L. L. HOLLIS IBM CORP. 919-254-0292	WAYNE DAVISON RES. LIBR. GRP. 415-328-0920	N/A	N/A	VACANT	SC21/WG5
X3T9 I/O INTERFACE	DEL SHOEMAKER DEC 202-383-5622	BILL BURR NBS 301-975-2914	GENE MILLIGAN CONTROL DATA 405-324-3070	ARNOLD ROCCATI EG&G WASH. AN. 301-840-3277	BILL BURR NBS 301-975-2914	SC13
X3T9.2 LOWER LEVEL INTERFACE	JOHN LOHMEYER NCR CORP. 316-688-8000	I. DAL ALLAN ENDL 408-867-6630	N/A	N/A	VACANT	SC13
X3T9.3 DEVICE LEVEL INTERFACE	VACANT	VACANT	N/A	N/A	VACANT	SC13
X3T9.5 LOCAL DIST. DATA INTERFACE	GENE MILLIGAN CONTROL DATA 405-324-3070	FLOYD ROSS UNISYS 215-341-1542	N/A	N/A	VACANT	SC13
X3T9.6 CENTRIDGE TAPE DRIVES	LOUIS DOMSHY ARCHIVE CORP. 714-641-0279	VACANT	N/A	N/A	VACANT	SC13
X3V1 TEXT: OFFICE & FUB. SYS.	L. M. COLLINS IBM CORP. 214-556-4392	ROY PIERCE XEROX CORP. 214-420-2804	MACKLIN BISHOP IBM CORP. 512-823-2005	STEVEN SCHRIER BOOZ, ALLEN, HAMIL. 202-767-4975	CHARLES REEVES ASSOC. OF INFO. 615-574-2342	SC18
X3V1.1 USER REQ.: R.S.T.	W. S. HOBGOOD IBM CORP.	VACANT	N/A	N/A	JONATHAN GRUDIN MCC 512-338-3615	SC18/WG1
X3V1.3 DOCUMENT ARCHITECTURE	ROY PIERCE XEROX CORP. 214-420-2804	HERMAN SILBIGER AT&T 201-898-2360	N/A	N/A	JAMES MASON OAK RIDGE LAB. 615-574-6973	SC18/WG3
X3V1.4 TEXT INTERCHNG	R. H. CHRISTIE CONTROL DATA 612-482-6689	CAROL MOLLEN NCR CORP. 612-638-7777	RONALD HARVEY HONEYWELL 602-861-4711	CAROL MOLLEN NCR CORP. 612-638-7777	ROBERT SAMUELL BELLSOUTH SER. 404-529-7246	SC18/WG4
X3V1.5 CONTENT ARCHT.	VACANT	VACANT	N/A	N/A	LYNNE ROSENTHAL NBS 301-975-3353	SC18/WG5
X3V1.8 TEXT DESCR. & PROC. LANGS.	BILL DAVIS INTERNAL REV. 202-566-6533	LAWRENCE BECK GRUMMAN DATA 516-682-8478	N/A	N/A	VACANT	SC18/WG8
X3V1.9 USER SYS. INTF. & SYMBOLS	CATHERINE SWYDER OAK RIDGE 615-574-5351	HELEN WALKER IBM CORP. 512-838-7861	N/A	N/A	VACANT	SC18/WG9
X3V1.10 FONT RESRS.	VACANT	VACANT	N/A	N/A	VACANT	SC18 -15-

X3TC	OFF. POS.	NAME	DATE	TICKLER	TERM
			EFFECTV. YY/MM/DD	NOTIF. YY/MM/DD	EXPIRATION YY/MM/DD
X3	CH	RICHARD GIBSON	03/01/87	N/A	N/A
X3	VC	DONALD LOUGHRY	03/10/87	N/A	N/A
JTC1TAG	CH	JOSEPH DEBLASI	11/02/87	08/02/90	11/02/90
JTC1TAG	VC	ROBERT MOUNTREE	12/28/87	09/28/90	12/28/90
AC	CH	VACANT			
AC	VC	VACANT			
SMC	CH	DELBERT SHOEMAKER	03/17/87	01/17/90	03/17/90
SMC	VC	EDWARD KELLY	03/17/87	01/17/90	03/17/90
SPARC	CH	WILLIAM RINEHULS	04/12/85	01/01/88	04/12/88
SPARC	VC	LEROY RODGERS	10/01/85	07/01/88	10/01/88
DBSSG	CH	EDWARD STULL	00/00/00	00/00/00	00/00/00
DBSSG	VC	NANCY McDONALD	06/25/85	03/25/88	06/25/88
DBSSG	IR	ELIZABETH FONG	09/18/85	04/18/88	09/18/88
SC21 TAG	CH	J. ASCHENBRENNER	11/30/84	06/30/87	11/30/87
SC21 TAG	VC	JON BECKER	07/27/87	04/27/90	07/27/90
SC22 TAG	CH	ROBERT FOLLETT	07/27/87	04/27/90	07/27/90
X3A1	CH	CARL KNOEDEL	06/22/87	03/22/90	06/22/90
X3A1	VC	ROBERT BLOSS	09/06/84	04/05/87	09/06/87
X3A1	IR	JOHN McDONNELL	09/01/84	04/01/87	09/01/87
X3A1.1	CH	PATRICK TRAGLIA	03/20/85	01/20/88	03/20/88
X3A1.2	CH	CARL KNOEDEL	09/10/84	04/10/87	09/10/87
X3A1.3	CH	C. BISS	04/17/87	01/17/90	04/17/90
X3B5	CH	RICHARD STEINBRENNER	04/24/87	01/24/90	04/24/90
X3B5	VC	SAM CHEATHAM	01/29/87	10/29/89	01/29/90
X3B5	IR	MICHAEL HOGAN	04/17/87	01/17/90	04/17/90
X3B6	CH	WILLIAM POLAND, JR.	04/17/87	01/17/90	04/17/90
X3B6	VC	C. DON WRIGHT	04/17/87	01/17/90	04/17/90
X3B6	IR	STAN REYNOLDS	05/20/87	02/20/90	05/20/90
X3B7	CH	BILL CARLSON	07/20/85	03/20/88	07/20/88
X3B7	IR	BILL CARLSON	06/20/85	03/20/88	06/20/88
X3B8	CH	JAMES BARNES	09/09/85	06/06/88	09/09/88
X3B8	VC	BILL PROCTOR	03/31/86	01/01/89	03/31/89
X3B8	IR	MICHAEL HOGAN	04/17/87	01/17/90	04/17/90
X3B8.1	CH	CLOYD TOLBERT	11/20/87	08/20/90	11/20/90
X3B9	CH	D. ODDY	06/07/85	01/01/88	06/07/88
X3B10	CH	ROBERT KITCHENER	01/06/86	09/06/88	01/06/89
X3B10	VC	ALICE DROOGAN	01/06/86	09/06/88	01/06/89
X3B10	IR	D. SEIGAL	01/06/86	09/06/88	01/06/89
X3B10.4	CH	ROBERT CALLEN	06/25/86	03/25/89	06/25/89
X3B11	CH	JOSEPH ZAJACKOWSKI	11/30/84	08/30/87	11/30/87
X3B11	VC	ROBERT BENDER	12/10/84	09/10/87	12/10/87
X3B11	IR	MICHAEL DEESE	12/15/84	09/15/87	12/15/87
X3H2	CH	DONALD DEUTSCH	04/17/87	01/17/90	04/17/90
X3H2	VC	CAROL JOYCE	10/21/86	05/21/89	10/10/89
X3H2	IR	LEONARD GALLAGHER	04/17/87	01/17/90	04/17/90
X3H3	CH	PETER BONO	08/03/87	05/03/90	08/03/90
X3H3	VC	JOHN MCCORNELL	11/15/86	07/15/89	11/15/89
X3H3	IR	JANET CHIN	05/20/87	02/20/90	05/20/90
X3H3.1	CH	DAVID BAILEY	01/06/86	09/06/88	01/06/89
X3H3.1	VC	DEBBIE CANN	05/26/87	02/26/90	05/26/90
X3H3.3	CH	KARLA VECCHIET	05/26/87	02/26/90	05/26/90
X3H3.3	VC	THOMAS POWERS	06/07/85	01/01/88	06/07/88
X3H3.4	CH	MADELINE SPARKS	12/15/84	09/15/87	12/15/87
X3H3.4	VC	GERALDINE CUTHBERT	06/07/85	01/01/88	06/07/88

# OFFICER APPOINTMENT & TERM EXPIRATION CHART

54

X3H3.5	CH	MARY MILLER	01/22/87	10/22/89	01/22/90
X3H3.6	CH	GEORGES GRINSTEIN	02/13/87	12/13/89	02/13/90
X3H4	CH	ANTHONY WINKLER	06/22/87	03/22/90	06/22/90
X3H4	VC	DAVID CARPENTER	04/17/87	01/17/90	04/17/90
X3H4	IR	DAVID THOMAS	04/17/87	01/17/90	04/17/90
X3J1	CH	JOHN KLENSIN	03/22/82	01/22/85	03/22/85
X3J1	VC	CHARLES NYLANDER	10/15/83	07/15/86	10/15/86
X3J1	AR	K. LUND	00/00/00	00/00/00	00/00/00
X3J2	CH	JAMES HARLE	04/24/87	01/24/90	04/24/90
X3J2	VC	ANDREW KLOSSNER	11/20/86	07/20/89	11/20/89
X3J3	CH	JENNIE ADAMS	04/17/87	01/17/90	04/17/90
X3J3	VC	JERROLD WAGENER	06/13/86	03/13/89	06/13/89
X3J3	IR	ANDREW JOHNSON	05/20/87	02/20/90	05/20/90
X3J4	CH	DON SCHRICCKER	09/23/85	06/23/88	09/23/88
X3J4	VC	BRUCE SINCLAIR	07/27/87	04/27/90	07/27/90
X3J4	IR	PEGGY BEARD	09/01/84	04/01/87	09/01/87
X3J7	CH	WICHAM LOH	02/13/87	12/13/89	02/13/90
X3J7	VC	C. WILSON	02/13/87	12/13/89	02/13/90
X3J7	IR	TOM BRENGAN	06/22/87	03/22/90	06/22/90
X3J9	CH	KENNETH ZEMROWSKI	11/02/83	08/12/86	11/02/86
X3J9	VC	MICHAEL HAGERTY	06/10/85	01/01/88	06/10/88
X3J9	IR	PAULA SCHWARTZ	01/06/86	09/06/88	01/06/89
X3J10	VC	G. FOSTER	00/00/00	00/00/00	00/00/00
X3J11	CH	JIM BRODIE	11/20/87	08/20/90	11/20/90
X3J11	VC	THOMAS PLUM	01/02/84	09/02/86	01/02/87
X3J11	IR	P.J. FLAUGER	07/27/87	04/27/90	07/27/90
X3J12	CH	ELI SZKLANKA	06/10/85	01/01/88	06/10/88
X3J12	VC	KENNETH LIDSTER	06/10/85	01/01/88	06/10/88
X3J14	VC	RAY DUNCAN	11/20/87	08/20/90	11/20/90
X3K1	CH	JOHN HACKNEY	07/27/87	04/27/90	07/27/90
X3K1	IR	RICHARD WERLING	10/20/87	07/20/90	10/20/90
X3K5	CH	JOHN WOOD	01/02/87	09/02/89	01/02/90
X3K5	VC	HELMUT THIESS	01/02/87	09/02/89	01/02/90
X3K5	IR	RICHARD BATEY	11/20/86	07/20/89	11/20/89
X3L2	CH	DONALD THELEN	06/13/86	03/13/89	06/13/89
X3L2	IR	HACKLIN BISHOP	09/10/84	04/10/87	09/10/87
X3L2.1	CH	DONALD THELEN	06/10/85	03/10/88	06/10/88
X3L8	CH	W. KENWORTHY, JR.	09/01/86	06/01/89	09/01/89
X3L8	VC	DUANE MARQUIS	06/28/86	03/28/89	06/28/89
X3L8	IR	KAREN KIRKERIDE	07/27/87	04/27/90	07/27/90
X3L8.4	CH	HENRY TOM	06/22/87	03/22/90	06/22/90
X3L8.6	CH	KAREN KIRKERIDE	03/22/85	12/22/87	03/22/88
X3L8.6	VC	ROGER PAYNE	05/26/87	02/26/90	05/26/90
X3L8.7	CH	DUANE MARQUIS	11/20/86	07/20/89	11/20/89
X3L8.7	VC	MARY SUMMERS	06/25/86	03/25/89	06/25/89
X3S3	CH	W. EMMONS	10/20/87	07/20/90	10/20/90
X3S3	IR	JOHN WHEELER	05/20/87	02/20/90	05/20/90
X3S3.1	CH	W. EMMONS	05/15/84	02/15/87	05/15/87
X3S3.3	CH	LYMAN CHAPIN	04/17/87	01/17/90	04/17/90
X3S3.3	VC	DAVID PISCITELLO	01/02/84	07/02/86	01/02/87
X3S3.5	CH	NEAL SEITZ	01/29/87	10/29/89	01/29/90
X3S3.7	CH	FRED BURG	03/13/87	12/13/89	03/13/90
X3S3.7	VC	RANDY SPUSTA	04/17/87	01/17/90	04/17/90
X3T1	CH	STEPHEN LEVIN	05/04/84	02/04/87	05/04/87
X3T1	VC	GEOFFREY TURNER	05/09/86	03/09/89	05/09/89
X3T1	IR	ROBERT ELANDER	00/00/00	00/00/00	00/00/00



X3T2	CH	LEONARD GALLAGHER	02/13/87	12/13/89	02/13/90
X3T2	VC	MIKE BLACKLEDGE	04/13/87	01/13/90	04/13/90
X3T2	IR	MAURICE SMITH	06/15/87	03/15/90	06/15/90
X3T5	CH	JERROLD POLEY	01/06/86	09/06/88	01/06/89
X3T5	VC	J. ASCHENBRENNER	06/22/87	03/22/90	06/22/90
X3T5	IR	J. ASCHENBRENNER	06/22/87	03/22/90	06/22/90
X3T5.1	CH	JOHN DAY	07/05/84	04/05/87	07/05/87
X3T5.1	VC	JOHN GURZICK	07/05/84	04/05/87	07/05/87
X3T5.4	CH	WILL COLLINS	11/15/86	07/15/89	11/15/89
X3T5.4	VC	MARK KLIERER	11/15/86	07/15/89	11/15/89
X3T5.5	CH	L. HOLLIS	05/20/87	02/20/90	05/20/90
X3T5.5	VC	W. (WAYNE) DAVISON	04/17/87	01/17/90	04/17/90
X3T9	CH	DELBERT SHOEMAKER	06/20/85	03/20/88	06/20/88
X3T9	VC	WILLIAM BURR	11/30/84	07/30/87	11/30/87
X3T9.2	CH	JOHN LORMEYER	02/20/87	12/20/89	02/20/90
X3T9.2	VC	I. DAL ALLAN	02/20/87	12/20/89	02/20/90
X3T9.5	CH	GENE HILLIGAN	03/21/85	01/01/88	03/21/88
X3T9.5	VC	FLOYD ROSS	06/10/85	01/01/88	06/10/88
X3T9.6	CH	LOUIS DOMSKY	02/04/85	10/04/87	02/04/88
X3V1	CH	L. COLLINS	06/07/85	01/01/88	06/07/88
X3V1	VC	ROY PIERCE	06/22/87	03/22/90	06/22/90
X3V1	IR	MACKLIN BISHOP	04/17/87	01/17/90	04/17/90
X3V1.1	CH	W. HOBGOOD	04/17/87	01/17/90	04/17/90
X3V1.3	CH	ROY PIERCE	10/02/85	07/02/88	10/02/88
X3V1.3	VC	HERMAN SILBIGER	10/02/85	07/02/88	10/02/88
X3V1.4	CH	R. CHRISTIE	04/24/87	01/24/90	04/24/90
X3V1.8	CH	WILLIAM DAVIS, JR.	10/02/85	07/02/88	10/02/88
X3V1.8	VC	LAWRENCE BECK	10/02/85	07/02/88	10/02/88
X3V1.9	CH	CATHRINE SNYDER	10/20/87	07/20/90	10/20/90
X3V1.9	VC	HELEN WALKER	10/02/85	07/02/88	10/02/88

54

# MAILING ADDRESSES OF OFFICERS, X3, X3 STANDING COMMITTEES, JTC1 TAG, X3 TECHNICAL COMMITTEES, TASK & STUDY GROUPS

54

JEANNE ADAMS, X3J3-CH  
NATL CTR FOR ATMOSPHERIC RES.  
SCD  
P.O. BOX 3000  
BOULDER, CO 80307  
303-497-1275

I DAL ALLAN, X3T9.2-VC  
ENDL  
14426 BLACK WALNUT COURT  
SARATOGA, CA 95070  
408-867-6630

J. R. ASCHENBRENNER, SC21 TAG-CH  
IBM CORPORATION X3T5-VC & IR  
DEPARTMENT E82/656  
P.O. BOX 12195  
RESEARCH TRIANGLE PARK, NC 27709  
919-254-0299

DAVID BAILEY, X3H3.1-CH  
CALCOMP/SANDERS  
DISPLAY PRODUCTS DIV.  
65 RIVER RD., CS 908  
HUDSON NH 03051-0908  
603-885-8141

JAMES L. BARNES, X3B8-CH  
C/O COMPAQ COMPUTER CORP.  
MO157  
P.O. BOX 692000  
HOUSTON, TX 77269-2000  
713-374-5674

RICHARD W. BATEY, X3K5-IR  
UNISYS  
P.O. BOX 500  
M/S CINE6  
BLUE BELL, PA 19424  
215/542-2709

PEGGY A. BEARD, X3J4-IR & SY  
NCR CORPORATION  
9900 OLD GROVE ROAD  
SAN DIEGO, CA 92131  
619-693-5730

LAWRENCE A. BECK, X3V1.8-VC  
GRUMMAN DATA SYSTEMS  
R & D, MS D12-237  
1000 WOODBURY ROAD  
WOODBURY, NY, U.S.A. 11797  
516-682-8478

JON M. BECKER, SC21 TAG-VC  
UNISYS  
P.O. BOX 1874, M/S N604  
SOUTHEASTERN PA 19398  
215/341-4643

KENNETH BEERS, X3J12-SY  
DIGITAL EQUIPMENT CORPORATION  
MK01-2/E25  
CONTINENTAL BLVD.  
MERRIMACK NH 03054  
603-884-5537

ROBERT A. BENDER, X3B11-VC  
PENTAX TECHNOLOGIES  
880 INTERLOCKEN PARKWAY  
BROOMFIELD, CO 80020  
303-460-1610

MACKLIN W. BISHOP, X3L2-IR  
IBM CORPORATION X3V1-IR  
11400 BURNET ROAD  
BLDG. 803, DEPT D34  
AUSTIN TX 78758  
512-823-2005

C. E. BISS, X3A13-CH  
PHOTOGRAPHIC SCIENCES CORP.  
P. O. BOX 338  
WEBSTER NY 14580  
716/265-1600

MIKE BLACKLEDGE, X3T2-VC  
SANDIA NATIONAL LABORATORIES  
DIVISION 7254  
P.O. BOX 5800  
ALBUQUERQUE, NM 87185  
505-846-6014

ROBERT C. BLOSS, X3A1-VC  
UNISYS  
41100 PLYMOUTH ROAD  
PLYMOUTH MI 48170  
313-451-4298

PETER R. BONO, X3H3-CH  
PETER R. BONO ASSOCIATES, INC.  
P. O. BOX 648  
GALES FERRY CT 06335-0648  
203/464-9350

TOM BRENGAN, X3J7-IR  
BOEING  
ORGN 62721/MS 55-14  
P. O. BOX 3707  
SEATTLE, WA 98124

JIM BRODIE, X3J11-CH  
J. BRODIE & ASSOCIATES  
106 S. TERRACE ROAD  
CHANDLER, AZ 85226  
602-961-0032

FRED BURG, X3S3.7-CH  
AT&T  
CRAWFORD CORNERS ROAD  
ROOM 3L-606  
HOLMDEL, NJ 07733  
201-949-0919

WILLIAM BURR, X3T9-VC & SY  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A207  
GAITHERSBURG MD 20899  
301-975-2914

DEBBIE CAHN, X3H3.1-VC  
BOEING  
COMPUTER SERVICES  
P.O. BOX 24346  
SEATTLE, WA 98124

ROBERT J. CALLEN, X3B10.4-CH  
DREXLER TECHNOLOGY CORP.  
2644 BAYSHORE PARKWAY  
MOUNTAIN VIEW CA 94043  
415-969-4428

BILL CARLSON, X3B7-CH & IR  
MAXTOR CORPORATION  
150 RIVER OAK PARKWAY  
SAN JOSE, CA 95134  
408-942-1700

DAVID E. CARLSON, X3S3.4-AC  
AT&T TECHNOLOGIES  
307 MIDDLETOWN/LINCROFT ROAD  
ROOM 3E-316  
LINCROFT, NJ 07738  
201-949-7503

DAVID CARPENTER, X3H4-VC  
PANSOPHIC SYSTEMS  
1250 E. DIEHL ROAD  
NAPERVILLE, IL 60566  
312-357-5950

GEORGE S. CARSON, X3H3.2-AC  
GSC ASSOCIATES, INC.  
P. O. BOX 2286  
13663 PRAIRIE AVENUE, SUITE B  
HAWTHORNE CA 90251-2286  
213-978-9351

LYMAN CHAPIN, X3S3.3-CH  
DATA GENERAL CORPORATION  
M/S D112  
4400 COMPUTER DRIVE  
WESTBORO, MA 01580  
617-870-6056

SAM CHEATHAM, X3B5-VC  
STORAGE TECHNOLOGY CORPORATION  
2270 SOUTH 88TH STREET MD PE  
LOUISVILLE CO 80027  
303/673-3359

JANET CHIN, X3H3-IR  
CHIN ASSOCIATES  
2980 COLLEGE AVENUE  
SUITE 2  
BERKELEY, CA 94705-2214  
415-843-9384

R. H. CHRISTIE, X3V1.4-CH  
CONTROL DATA CORPORATION  
(ARH220)  
4201 LEXINGTON AVENUE, N.W.  
ARDEN HILLS, MN 55126  
612-482-6689

WILL COLLINS, X3T5.4-CH  
CODEX CORPORATION  
M/S C-150  
20 CABOT BLVD.  
MANSFIELD MA 02048  
617-364-2000 X7367

L. M. COLLINS, X3V1-CH  
IBM CORPORATION  
200 LAS COLINAS BOULEVARD  
15-07-40  
IRVING, TX 75039  
214-536-4392

ARTHUR W. COSTON, X3J1-VR  
APPLIED INFORMATION SYSTEMS  
500 EASTOWNE DRIVE  
SUITE 207  
CHAPEL HILL, NC 27514  
919-942-7801

JOHN CUGINI, X3J2-VR  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A235  
GAITHERSBURG MD 20899  
301-975-3248

GERALDINE R. CUTHBERT, X3H3.4-VC  
SOFTWARE TECHNOLOGY, INC.  
1511 PARK AVENUE  
MELBOURNE, FL 32901  
305-723-3999

WILLIAM W. DAVIS, JR., X3V1.8-CH  
INTERNAL REVENUE SERVICE  
ROOM 1237, PM-5:FM-P  
1111 CONSTITUTION AVENUE  
WASHINGTON, DC 20224  
202-566-6533

W. (WAYNE) DAVISON, X3T5.5-VC  
RESEARCH LIBRARIES GROUP INC.  
JORDAN QUAD - JUNIPER  
STANFORD, CA 94305  
415-328-0920

JOHN D. DAY, X3T5.1-CH  
CODEX CORPORATION  
20 CABOT BOULEVARD  
MANSFIELD MA 02048  
617-364-2000 X7111

JOSEPH S. DEBLASI, JTC1 TAG-CH  
IBM CORPORATION  
2000 PURCHASE STREET  
PURCHASE NY 10577-2597  
914/697-7280

MICHAEL DEESE, X3B11-IR  
CHEROKEE DATA SYSTEMS  
C/O 4872 EARLE CIRCLE  
BOULDER, CO 80301  
303-530-0618

DONALD DEUTSCH, X3H2-CH  
GENERAL ELECTRIC COMPANY  
301 N. WASHINGTON ST.  
MCO3A  
ROCKVILLE MD 20850  
301-340-4580

LOUIS C. DOMSHY, X3T9.6-CH  
ARCHIVE CORPORATION  
1650 SUNFLOWER AVENUE  
COSTA MESA, CA 92626  
714-641-0279

ALICE DROOGAN, X3B10-VC  
MASTERCARD INTERNATIONAL  
888 7TH AVENUE  
NEW YORK, NY 10106  
212-649-5358

RAY DUNCAN, X3J14-VC  
LABORATORY MICROSYSTEMS, INC.  
3007 WASHINGTON BLVD.  
SUITE 230, P.O. BOX 10430  
MARINA DEL REY, CA 90295  
213-306-7412

VICKI ECKELS, X3J4-VR  
TEXAS INSTRUMENTS  
P.O. BOX 2909, M/S 2078  
AUSTIN, TX 78769  
512-250-6512

ROBERT ELANDER, X3T1-IR  
IBM CORPORATION  
KINGSTON, NY 12401  
914-385-6692

W. F. EMMONS, X3S3-CH  
IBM CORPORATION X3S3.1-CH  
SYSTEMS COMMUNICATIONS DIV.  
P.O. BOX 12195, DEPT. C71/656  
RES. TRIANGLE PK NC 27709-2195  
919-254-0294

JERROLD S. FOLEY, X3T5-CH  
ELECTRONIC DATA SYSTEMS CORP.  
300 EAST BIG BEAVER ROAD  
P.O. BOX 7019, CUBE 5176  
TROY, MI 48063  
313-524-8416

ROBERT H. FOLLETT, SSISG-AC  
IBM CORPORATION SC22 TAG-CH  
643/RCTR/9E  
6705 ROCKLEDGE DRIVE  
BETHESDA, MD 20817  
301-564-2108

VICTOR POLWARCZNY, X3J9-SY  
NCR CORPORATION  
E & M CAMBRIDGE  
800 COCHRAN AVENUE  
CAMBRIDGE, OH 43725-0523  
614-439-0516

ELIZABETH FONG, DBSSG-SY & IR  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A258  
GAITHERSBURG MD 20899  
301-975-3250

G. H. POSTER, X3J10-VC  
SYRACUSE UNIVERSITY  
DEPT. OF ELEC. & COMPUTER ENG.  
111 LINK HALL  
SYRACUSE, NY 13244-1240  
315/423-4375

LEONARD J. GALLAGHER, X3H2-IR  
NBS X3T2-CH  
BUILDING 225, ROOM A156  
GAITHERSBURG, MD 20899  
301-975-3251

RICHARD GIBSON, X3-CH  
AT&T  
ROOM 5A 211  
ROUTE 202 & 206N  
BEDMINSTER NJ 07921  
201-234-3795

MICHAEL M. GORMAN, X3H2-SY  
WHITEMARSH INFO. SYSTEMS  
2008 ALTHEA LANE  
BOWIE MD 20716  
301-249-1142

GEORGES GRINSTEIN, X3H3.6-CH  
UNIVERSITY OF LOWELL  
GRAPHICS RESEARCH LABORATORY  
ONE UNIVERSITY AVENUE  
LOWELL, MA 01854  
617-452-5000

JONATHAN GRUDIN, X3V1.1-SY  
MCC  
P.O. BOX 200195  
AUSTIN, TX 78720  
512-338-3615

JOHN GURZICK, X3T5.1-VC  
MCI  
8283 GREENSBORO DRIVE  
MS 1137/625  
MCLEAN, VA 22102  
703-442-5363

JOHN HACKNEY, X3K1-CH  
INTERSTATE ELECTRONICS  
P. O. BOX 3117  
ANAHEIM CA 92803  
714/635-7210

MICHAEL HAGERTY, X3J9-VC  
RADIONICS  
27911 BERWICK DRIVE  
CARMEL, CA 93923  
408-757-8877

JAMES A. HARLE, X3J2-CH  
U.S. NAVAL ACADEMY  
COMPUTING CENTER  
ANNAPOLIS MD 21402

RONALD B. HARVEY, X3V1.4-IR  
HONEYWELL INFORMATION SYSTEMS  
P.O. BOX 8000, M/S AZ13-H32  
PHOENIX, AZ 85066-8000  
602-861-4711

KURT W. HIRCHERT, X3J3-VR  
NATL. CTR SUPERCOMPUTING APPL.  
152 COMPUTING APPLICATION BLDG  
605 E. SPRINGFIELD AVENUE  
CHAMPAIGN, IL 61821

W. S. HOBGOOD, X3V1.1-CH  
IBM CORPORATION  
P.O. BOX 12195  
BLDG. 662, DEPT. J27  
RES. TRIANGLE PARK, NC 27709

MICHAEL HOGAN, X3B5-IR  
NBS X3B8-IR  
BUILDING 225, ROOM A61  
GAITHERSBURG MD 20899  
301-975-2926

L. L. HOLLIS, X3T5.5-CH  
IBM CORPORATION  
C71/656  
P. O. BOX 12195  
RES. TRIANGLE PARK NC 27709  
919-254-0292

54

PAUL D. JAHNKE, X3B7-VR  
3M COMPANY  
3M CENTER, BLDG. 236-GL-19  
ST. PAUL MN 55144  
612/736-0117

ANDREW JOHNSON, X3J3-IR  
PRIME COMPUTER, INC. X3J11-VR  
MS 10C17-3  
500 OLD CONNECTICUT PATH  
FRAMINGHAM MA 01701  
617/879-2960 X 4045

JIM JOHNSON, X3B10.1-SY  
DATA CARD CORPORATION  
11111 BREN ROAD WEST  
MINNETONKA MN 55440  
612-933-1223

CAROL D. JOYCE, X3H2-VC  
RELATIONAL TECHNOLOGY INC.  
1080 MARINA VILLAGE PARKWAY  
ALAMEDA, CA 94501  
415-769-1400

CATHERINE A. KACHURIK, X3-SY, SPC-SY  
X3 SECRETARIAT/CBEMA JTC1 TAG AC-SY  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202/737-8888

EDWARD KELLY, SMC-VC  
AMP INCORPORATED  
MS 210-01  
P. O. BOX 3608  
HARRISBURG PA 17105-3608  
717-561-6153

WILLIAM H. KENWORTHY, JR., X3L8-CH & VR  
U.S. DEPT. OF DEFENSE  
C/O 420 KIMBLEWICK DRIVE  
SILVER SPRING, MD 20904  
202/694-3361

KAREN KIRKBRIDE, X3L8-IR  
4204 SANDHURST COURT X3L8.6-CH  
ANNANDALE, VA 22003  
202/746-0797

ROBERT A. KITCHENER, X3B10-CH  
LOGICARD SYSTEMS, INC.  
P.O. BOX 637  
ARMONK, NY 10504  
914-273-8734

JOHN C. KLENSIN, X3J1-CH  
MIT  
77 MASSACHUSETTS AVENUE  
ROOM 20A-226  
CAMBRIDGE, MA 02139  
617/253-1355

MARK KLERER, X3T5.4-VC  
AT&T  
CRAWFORDS CORNER ROAD  
ROOM 3L613  
HOLMDEL, NJ 07733  
201-949-8645

ANDREW KLOSSNER, X3J2-VC  
TEKTRONIX, INC.  
P. O. BOX 1000, MS 61-183  
WILSONVILLE OR 97070

CARL KNOEDEL, X3A1-CH, VR & SY  
STANDARD REGISTER CO. X3A1.2-CH  
626 ALBANY STREET  
P. O. BOX 1167  
DAYTON OH 45401  
513/443-1072

BERNARD KOCIS, X3H2-VR  
NARDAC  
CODE 3023, BLDG. 143  
WASHINGTON NAVY YARD  
WASHINGTON, DC 20374  
202-433-4390 OR 4391

DAVE LARSON, X3H3.4-SY  
HEWLETT-PACKARD  
3404 E. HARMONY ROAD  
FT. COLLINS, CO 80525  
303-226-3800

STEPHEN LEVIN, X3T1-CH & VR  
ITT DCD  
492 RIVER ROAD  
NUTLEY, NJ 07110-3696  
201-694-0661

KENNETH LIDSTER, X3J12-VC  
D.L.S.C INCORPORATED  
11070 WHITE ROCK ROAD #210  
RANCHO CORDOVA, CA 95670-6001  
916-635-7300

WICKHAM LOH, X3J7-CH  
IBM CORPORATION  
DEPT. 75L, SUITE 2100  
11601 WILSHIRE BLVD.  
LOS ANGELES, CA 90025  
213-312-5926

JOHN LOHMEYER, X3T9.2-CH  
NCR CORPORATION  
3718 N. ROCK ROAD  
WICHITA KS 67207  
316/688-8000

DONALD C. LOUGHRY, X3-VC, SPC-AC  
HEWLETT-PACKARD  
INFORMATION NETWORKS DIVISION  
19420 HOMESTEAD RD. MS 43UX  
CUPERTINO CA 95014-0606  
408-257-7000 X2454

K. R. LUND, X3J1-AR  
IBM CORPORATION  
M77/D34  
P.O. BOX 50020  
SAN JOSE CA 95150

PETER LUSTER, X3J10-VR  
2540 N. UPLAND STREET  
ARLINGTON VA 22207  
703-525-8001

DUANE J. MARQUIS, X3L8-VC, X3L8.7-CH  
NATL TELECOM. & INFO. AGENCY  
179 ADMIRAL COCHRAN DRIVE  
ANNAPOLIS, MD 21401  
301-224-4300

JAMES D. MASON, X3V1.3-SY  
OAK RIDGE NATIONAL LABORATORY  
P.O. BOX X, 4500-S  
OAK RIDGE, TN 37831  
615/574-6973

ROBERT F. MATHIS, X3J13-AC  
9712 CERALENE DRIVE  
FAIRFAX, VA 22032-1704  
703-425-5923

DONALD J. MCCAFFREY, X3H4-SY  
7432 LONG PINE DRIVE X3H4-VR  
SPRINGFIELD VA 22151  
703/569-2390

JOHN I. MCCONNELL, X3H3-VC  
DEC X3H3.5-AV  
110 SPITBROOK ROAD  
ZK02-3/R56  
NASHUA, NH 03063  
603-884-2285

NANCY MCDONALD, DBSSG-VC  
COMPUTER TECHNOLOGY PLANNING  
10014 NORTH DALE MABRY  
SUITE 101  
TAMPA, FL 33618  
813-968-2660

JOHN MCDONNELL, X3A1-IR  
RECOGNITION EQUIPMENT, INC.  
P.O. BOX 660204  
IRVING, TX 75266-0204  
214-579-5800

JAMES C. MICHENER, X3H3-SY  
APOLLO COMPUTER, INC.  
330 BILLERICA ROAD  
CHELMSFORD, MA 01824  
617-256-6600 X5275

MARY MILLER, X3H3.5-CH  
LANDMARK GRAPHICS CORP.  
333 CYPRESS RUN  
SUITE 100  
HOUSTON TX 77094-1605  
713-531-4080

GENE MILLIGAN, X3T9-IR  
CDC X3T9.5-CH  
MAGNETIC PERIPHERALS INC.  
M/S OKM 275, P.O. BOX 12313  
OKLAHOMA CITY OK 73157  
405/324-3070

CAROL MOLLEN, X3V1.4-VC & VR  
NCR CORPORATION  
NCR COMTEN  
2700 SNELLING AVENUE, NORTH  
ROSEVILLE, MN 55113  
612/638-7777

ROY P. MULLINAX, X3K5-SY  
HOUSING & URBAN DEVELOPMENT  
2518 JACKSON PARKWAY  
VIENNA VA 22180

WES MUNSIL, X3J9-VR  
APSE, INC.  
1713 NORTH AMBER  
MESA, AZ 85203  
602-962-6815

RICHARD D. MYERS, X3L2-SY  
UNISYS  
ROUTE 202 NORTH  
FLEMINGTON, NJ 08822

54

CHARLES NYLANDER, X3J1-VC  
DIGITAL EQUIPMENT CORPORATION  
ZK02-3/N30  
110 SPIT BROOK ROAD  
NASHUA NH 03062  
603/881-2081

D. H. ODDY, X3B9-CH  
MOORE BUSINESS FORMS  
MOORE RESEARCH DIVISION  
300 LANG BOULEVARD  
GRAND ISLAND, NY 14072-1697  
716/773-0378

SHARON L. ORIEL, X3B11-SY  
DOW CHEMICAL COMPANY  
2040 WILLARD H. DOW CENTER  
MIDLAND, MI 48674  
517-636-2173

ROGER L. PAYNE, X3L8-6-VC  
U.S. GEOLOGICAL SURVEY  
523 NATIONAL CENTER  
RESTON VA 22092  
703-648-4544

GWENDY J. PHILLIPS, SPARC-SY  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202/737-8888

ROY PIERCE, X3V1-VC, X3V1.3-CH  
XEROX CORPORATION  
1301 RIDGEVIEW  
M/S 111  
LEWISVILLE TX 75067  
214-420-2804

DAVID M. PISCITELLO, X3S3.3-VC  
UNISYS  
P.O. BOX 1874, M/S N606  
SOUTHEASTERN, PA 19398  
215-341-4642

P.J. PLAUGER, X3J11-SY & IR  
WHITESMITHS, LTD.  
99 POWER ROAD  
WESTFORD, MA 01886  
617-692-7800

THOMAS PLUM, X3J11-VC  
PLUM HALL INC.  
1 SPRUCE AVENUE  
CARDIFF NJ 08232  
609/927-3770

JOSEPH PODVOJSKY, X3S3.7-SY  
MITRE CORPORATION  
MS B270  
P. O. BOX 208  
BEDFORD MA 01730  
617-271-2155

WILLIAM B. POLAND, JR., X3B6-CH  
NASA/GODDARD SPACE FLIGHT CTR.  
CODE 730.4  
GREENBELT MD 20771  
301-286-8592

THOMAS POWERS, X3H3.3-VC  
DIGITAL EQUIPMENT CORPORATION  
MLS-3/E12  
146 MAIN STREET  
MAYNARD MA 01754  
617-493-2704

BILL PROCTOR, X3B8-VC  
MEMOREX CORPORATION  
1200 MEMOREX DRIVE  
M/S 0037  
SANTA CLARA, CA 95052  
408-957-0684

CHARLES REEVES, JR., X3V1-SY  
ASSOC. OF INFORMATION  
SYSTEMS PROFESSIONALS  
10812 DINEEN DRIVE  
KNOXVILLE TN 37922  
615-574-2342

J. K. REID, X3J3-SY  
AERE HARWELL  
BUILDING 8.9  
DIDCOT, OXFORDSHIRE OX11 0RA  
ENGLAND

STAN REYNOLDS, X3B6-IR  
SANDIA NATIONAL LABORATORIES  
DIV. 7535  
P.O. BOX 5800  
ALBUQUERQUE, NM 87185  
505-844-3518

WILLIAM C. RINEHULS, SPARC-CH  
GENERAL SERVICES ADMN.  
ADTS  
8457 RUSHING CREEK COURT  
SPRINGFIELD VA 22153-2532  
202/566-1180

GARY S. ROBINSON, JTC1 TAG AC-AC  
DEC  
146 MAIN STREET, ML012B/E51  
MAYNARD MA 01754-2572  
617/493-4094

ARNOLD J. ROCCATI, X3B5-VR  
EG&G WASHINGTON X3T9-VR  
ANALYTICAL SERVICES CENTER  
1396 PICCARD DRIVE  
ROCKVILLE, MD 20850-4308  
301-840-3277

LEROY RODGERS, SPARC-VC  
DIGITAL EQUIPMENT CORPORATION  
CONTINENTAL BLVD., MK1-2/L6  
MERRIMACK, NH 03054  
603-884-8318

LYNNE ROSENTHAL, X3V1.5-SY  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B257  
GAITHERSBURG, MD 20899  
301-975-3353

FLOYD E. ROSS, X3T9.5-VC  
UNISYS  
P.O. BOX 1874, M/S S511  
SOUTHEASTERN, PA 19398  
215-341-1542

ROBERT ROUNTREE, JTC1 TAG-VC  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM B168  
GAITHERSBURG, MD 20899-0999  
301-975-2827

JOHN RUSSELL, X3L2-VR  
CONTROL DATA CORPORATION  
8100 34TH AVE., SO. BMW03M  
MINNEAPOLIS MN 55440  
612/853-5414

ROBERT L. SAMUELL, III, X3V1.4-SY  
BELLSOUTH SERVICES, INC.  
40V93 SOUTHERN BELL CENTER  
675 WEST PEACHTREE ST., N.E.  
ATLANTA, GA 30375  
404-529-7246

KENNETH M. SCHILLING, X3J12-VP  
MCBA  
425 WEST BROADWAY  
GLENDALE, CA 91204  
818-242-9600

DON SCHRICKER, X3J4-CH  
WANG LABORATORIES, INC.  
M/S 013-790, ONE INDUSTRIAL AVE.  
LOWELL MA 01851  
617-967-7628

STEVEN A. SCHRIER, X3V1-VR  
BOOZ, ALLEN & HAMILTON, INC.  
4330 EAST-WEST HIGHWAY  
BETHESDA, MD 20814  
202-767-4875

PAULA SCHWARTZ, X3J9-IR  
RESEARCH LIBRARIES GROUP INC.  
852 LA PARA AVENUE  
PALO ALTO, CA 94306  
415-329-3534

DUNHAM B. SEELEY, X3B9-SY  
THE STANDARD REGISTER COMPANY  
1150 UNIVERSITY AVE., BOX 910  
ROCHESTER, NY 14603  
716-271-3400 X 2407

D. L. SEIGAL, X3B10-IR  
AMERICAN EXPRESS  
TRAVEL RELATED SERVICES  
1647 E. MORTEN AVENUE  
PHOENIX, AZ 85020  
602-371-3637

NEAL B. SEITZ, X3S3.5-CH  
U.S. DEPT. OF COMMERCE  
NTIA/TTS.N3, 325 BROADWAY  
BOULDER CO 80303  
303/497-3106

DELBERT L. SHOEMAKER, SMC-CH  
DEC X3T9-CH  
1331 PENNSYLVANIA AVE. N.W.  
SIXTH FLOOR  
WASHINGTON, DC 20004  
202-383-5622

HERMAN R. SILBIGER, X3V1.3-VC  
AT&T TECHNOLOGIES  
ROOM 3L-603  
CRAWFORD CORNERS ROAD  
HOLMDEL, NJ 07733  
201/898-2360

BRUCE SINCLAIR, X3J4-VC  
RYAN MCFARLAND CORPORATION  
6907 CAPITOL OF TEXAS HWY. N.  
SUITE 290  
AUSTIN, TX 78731  
512-343-1010

MAURICE SMITH, X3T2-IR  
BENDIX CORP.  
D931-2A45, P.O. BOX 1159  
KANSAS CITY, MO 64141  
816-997-3590

54

CATHRINE E. SNYDER, X3V1.9-CH  
OAK RIDGE NATIONAL LABORATORY  
BLDG. 4500 N, M/S 181  
P.O. BOX X  
OAK RIDGE, TN 37831-6181  
615-574-5351

MADELEINE SPARKS, X3H3.4-CH  
UNISYS X3H3-VR  
1500 PERIMETER PKWY. STE. 400  
HUNTSVILLE, AL 35806-1686  
205-837-7610

RANDY SPUSTA, X3S3.7-VC  
BELL COMMUNICATIONS RESEARCH  
331 NEWMAN SPRINGS ROAD  
ROOM 1F-333  
RED BANK, NJ 07701

J. H. STEARNS, X3B10-SY  
DATA CARD CORPORATION  
11111 BREN ROAD WEST  
MINNETONKA, MN 55440  
612-933-1223

RICHARD STEINBRENNER, X3B5-CH  
AT&T TECHNOLOGIES  
C/O AT&T BELL LABORATORIES  
ONE WHIPPANY DRIVE  
WHIPPANY, NJ 07931  
201/386-7053

PATRICIA A. STEINER, SMC-SY  
X3 SECRETARIAT/CBEMA  
311 FIRST STREET, N.W.  
SUITE 500  
WASHINGTON, DC 20001  
202-737-8888

EDWARD L. STULL, DBSSG-CH & VR  
GTE GOVERNMENT SYSTEMS  
1700 RESEARCH BOULEVARD  
ROOM 3095  
ROCKVILLE, MD 20850  
301-294-8649

MARY SUMMERS, X3L8.7-VC  
DEFENSE INTELLIGENCE AGENCY  
ATTN: RSE-1A  
WASHINGTON, DC 20340-3466  
202-373-3007

ELI SZKLANKA, X3J12-CH  
TEC COMPUTER SYSTEMS  
30 TOWER ROAD, M/S MK02-1/H10  
NEWTON, MA 02164  
617-964-3890

DONALD J. THELEN, X3L2-CH  
AT&T TECHNOLOGIES X3L2.1-CH  
C/O AT&T INFORMATION SYSTEMS  
60 COLUMBIA TURNPIKE, A-A216  
MORRISTOWN, NJ 07960  
201/644-3728

HELMUT E. THIESS, X3K5-VC  
1834 LAMONT STREET, NW  
WASHINGTON, DC 20010  
202-265-9439

DAVID THOMAS, X3H4-IR  
MSP, INC.  
131 HARTWELL AVE.  
LEXINGTON, MA 02173-3126  
617-863-5800

CLOYS TOLBERT, X3B8-SY  
COMPAQ X3B8.1-CH  
20555 FM 149  
HOUSTON TX 77070  
713-370-0670

HENRY TOM, X3L8.4-CH  
NATIONAL BUREAU OF STANDARDS  
BUILDING 225, ROOM A252  
KCST BLDG. 225  
GAITHERSBURG MD 20899  
301-975-3271

PATRICK J. TRAGLIA, X3A1.1-CH  
IBM CORPORATION  
GLENDALE LAB.  
P. O. BOX 6  
ENDICOTT NY 13760  
607/752-3357

PAUL TSUCHIYA, X3S3.3-SY  
MITRE CORPORATION  
W420  
7525 COLSHIRE DRIVE  
MCLEAN, VA 22102  
703-883-7352

GEOFFREY W. TURNER, X3T1-VC  
BANK OF AMERICA  
#3410  
P.O. BOX 37000  
SAN FRANCISCO, CA 94137  
415-675-4204

KARLA VECCHIET, X3H3.3-CH  
METHEUS CORPORATION  
5510 N.E. ELAM YOUNG PKWY.  
P.O. BOX 1049  
HILLSBORO OR 97123  
503-640-8000

MABEL VICKERS, SC22-AV  
NATIONAL BUREAU OF STANDARDS  
TECHNOLOGY BLDG. 225, ROOM A268  
GAITHERSBURG, MD 20899-0999  
301-975-3277

JERROLD L. WAGENER, X3J3-VC  
AMOCO PRODUCTION COMPANY  
4502 EAST 41ST STREET  
P.O. BOX 3385  
TULSA OK 74102  
918-664-3415

HELEN W. WALKER, X3V1.9-VC  
IBM CORPORATION  
11400 BURNET ROAD, 69T/008  
AUSTIN TX 78758  
512/838-7861

RICHARD WERLING, X3K1-IR  
6308 BEACHWAY DRIVE  
FALLS CHURCH VA 22044  
703-820-4034

J. A. WESTENBROEK, X3B11-VR  
PHILIPS AND DUPONT OPTICAL  
CHESTNUT RUN - CR709/ESL  
WILMINGTON DE 19898  
302-999-5354

JOHN L. WHEELER, X3S3-IR  
WINTERGREEN INFORMATION SYS.  
CARRIAGE HOUSE COMMONS  
SUITE 347, 159 WEST MAIN ST.  
WEBSTER, NY 14580  
716-671-4087

57

C. W. WILSON, X3J7-VC  
MARTIN MARIETTA  
BLDG. 9113, M/S 2F  
POST OFFICE BOX Y  
OAK RIDGE, TN 37831

ANTHONY J. WINKLER, X3H4-CH  
P.O. BOX 2308  
FAIRFAX, VA 22031  
703-848-2750

JOHN R. WOOD, X3K5-CH  
IBM CORPORATION  
COMMUNICATIONS PRODUCTS DIV.  
P.O. BOX 12195, DEPT. E48-6563  
RES. TRIANGLE PARK NC 27709  
919-254-0182

C. DON WRIGHT, X3B6-VC  
LOCKHEED ELECTRONICS, INC.  
C/O 7 CAMBRIDGE ROAD  
BEDMINSTER, NJ 07921  
201-234-9484

RONALD E. YOUNG, X3B8-VR  
REY  
P.O. BOX 2297  
LOS GATOS, CA 95031  
408-947-8700

JOSEPH S. ZAJACKOWSKI, X3B11-CH  
CHEROKEE DATA SYSTEMS  
SUITE H  
1880 SOUTH FLATIRONS COURT  
BOULDER, CO 80301  
303-449-1239

KENNETH M. ZEMROWSKI, X3J9-CH  
TRW  
2751 PROSPERITY AVENUE  
FAIRFAX, VA 22031-4375  
703-876-8056

KEY:

CH = CHAIR  
AC = ACTING CHAIR  
VC = VICE CHAIR  
AV = ACTING VICE CHAIR  
IR = INTERNATIONAL REPRESENTATIVE  
AR = ACTING INTERNATIONAL REP.  
VR = VOCABULARY REPRESENTATIVE  
SY = SECRETARY

**SPARC NATIONAL AND INTERNATIONAL MONITORING**

COMMITTEE	MEMBER	REPORTING DATE - 88
X3A1	Mr. Bass	March
X3B5/SC11/SC15*	Mr. Fogle	July
X3B6/WG2	Mr. Bass	November
X3B7/SC10	Mr. Salsman	January
X3B8/SC11/SC15*	Mr. Kurihara	January
X3B9	Mr. Bass	July
X3B10/SC17	Mr. Bass	July
X3B11/SC23	Mr. Follett	July
X3H2	Mr. LaPlant	March
X3H3	Mr. Fogle	November
X3H4	Mr. McNamara	March
X3J1	Mr. Ryland	September
X3J2	Mr. Kurihara	November
X3J3	Mr. LaPlant	May
X3J4	Mr. McNamara	September
X3J7	Mr. Ryland	January
X3J9	Mr. Kurihara	March
X3J10	Mr. Fogle	January
X3J11	Mr. Kurihara	May
X3J12	Mr. LaPlant	September
X3J13	Mr. LaPlant	November
X3J14	Mr. Virtue	January
X3K1	Ms. Butler	January
X3K5/SC1	Ms. Butler	November
X3L2/SC2	Mr. Haines	July
X3L8/SC14	Mr. McNamara	November
X3S3/SC6	Mr. Follett	March
X3T1/SC20	Mr. Salsman	November
X3T2	Mr. Follett	November
X3T5	Mr. Virtue	November
X3T9/SC13	Mr. Ryland	March
X3V1/SC18	Mr. McNamara	November
SPARC/DBSSG	Ms. Butler	March
SPARC/SSISG	Mr. Follett	May
PLIP	(Inactive)	
U.S. TAG to SC7	Ms. Butler	November
U.S. TAG to SC21	Mr. Follett	March
U.S. TAG to SC22	Ms. Butler	March

\*SC11 and SC15 liaison is conducted through reporting to both TC's.

**PROJECT LIAISON**

Project	Member	Liaison With
Ada	Mr. Virtue	DOD HOLWG
MUMPS	Mr. LaPlant	MDC

**SPARC LIAISON ACTIVITIES WITH EXTERNAL ORGANIZATIONS**

Organization	Member
Information Systems Standards Board (ISSB)	Mr. Rinehals
ANSI Planning Panel on Industrial Automation	Mr. Rodgers
IEEE Computer Society Standards Activities Board	Mr. LaPlant
X9 - Financial Services	Mr. Bass
X12 - Electronic Business Data Interchange	Mr. Salsman
IEC/TC83 Working Group on Planning and Requirements	Mr. McNamara

**IAC INTERNATIONAL LIAISON ASSIGNMENTS**

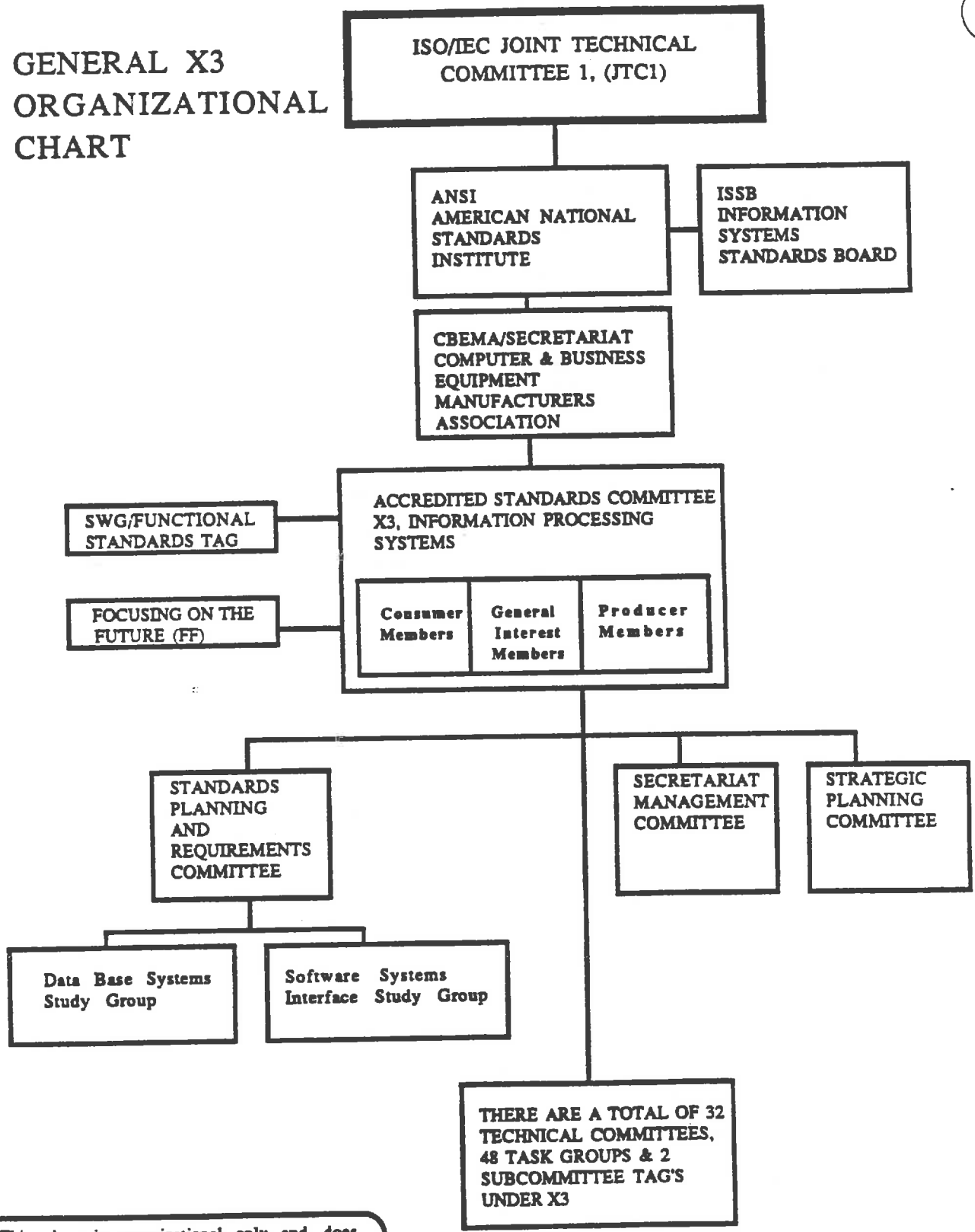
- ISO/IEC/JTC1 - JOSEPH DEBLASI
- WG2 - MARV BASS
- SC1 - GARY HAINES
- 2 - BOB ROUNTREE
- 6 - GARY ROBINSON
- 7 - SAVA SHERR
- 11 - BOB ROUNTREE
- 13 - EDWARD KELLY
- 14 - FRED VIRTUE
- 15 - MIKE HOGAN
- 17 - MARV BASS
- 18 - MARY ANNE GRAY
- 20 - GARY ROBINSON
- 21 - FRANK KIRSCHENBAUM
- 22 - BOB ROUNTREE
- 23 - GARY HAINES
- 24 - MARY ANNE GRAY

- SO/TC68 BANKING - MARV BASS
- TC145 GRAPHICS - MARY ANNE GRAY
- TC154 DATA ELEMENTS - FRED VIRTUE
- TC159 ERGONOMICS - MARY ANNE GRAY

- ECMA - GARY ROBINSON
- IEC - SAVA SHERR
- CCITT - RICHARD GIBSON
- ISO - JOSEPH DEBLASI

- ISO/TC184 IND'L AUTOMATION - ED KELLY
- SC47B MICROPROCESSORS - SAVA SHERR
- TC74 PRODUCT SAFETY - MARV BASS
- EMI - SAVA SHERR
- TC86 FIBER OPTICS - ED KELLY
- ACOS - MARV BASS
- CCITT/SGVII NETWORKS - RICHARD GIBSON
- VIII TERMINALS - MARY ANNE GRAY
- XVII DATA COMM. - RICHARD GIBSON
- XVIII DIG. NETWORKS - BOB ROUNTREE
- IEC/TC83 INFO. TECH. EQPT. - SAVA SHERR
- SGx - ERGONOMICS - MARY ANNE GRAY

# GENERAL X3 ORGANIZATIONAL CHART



This chart is organizational only and does not reflect the Standards approval process.

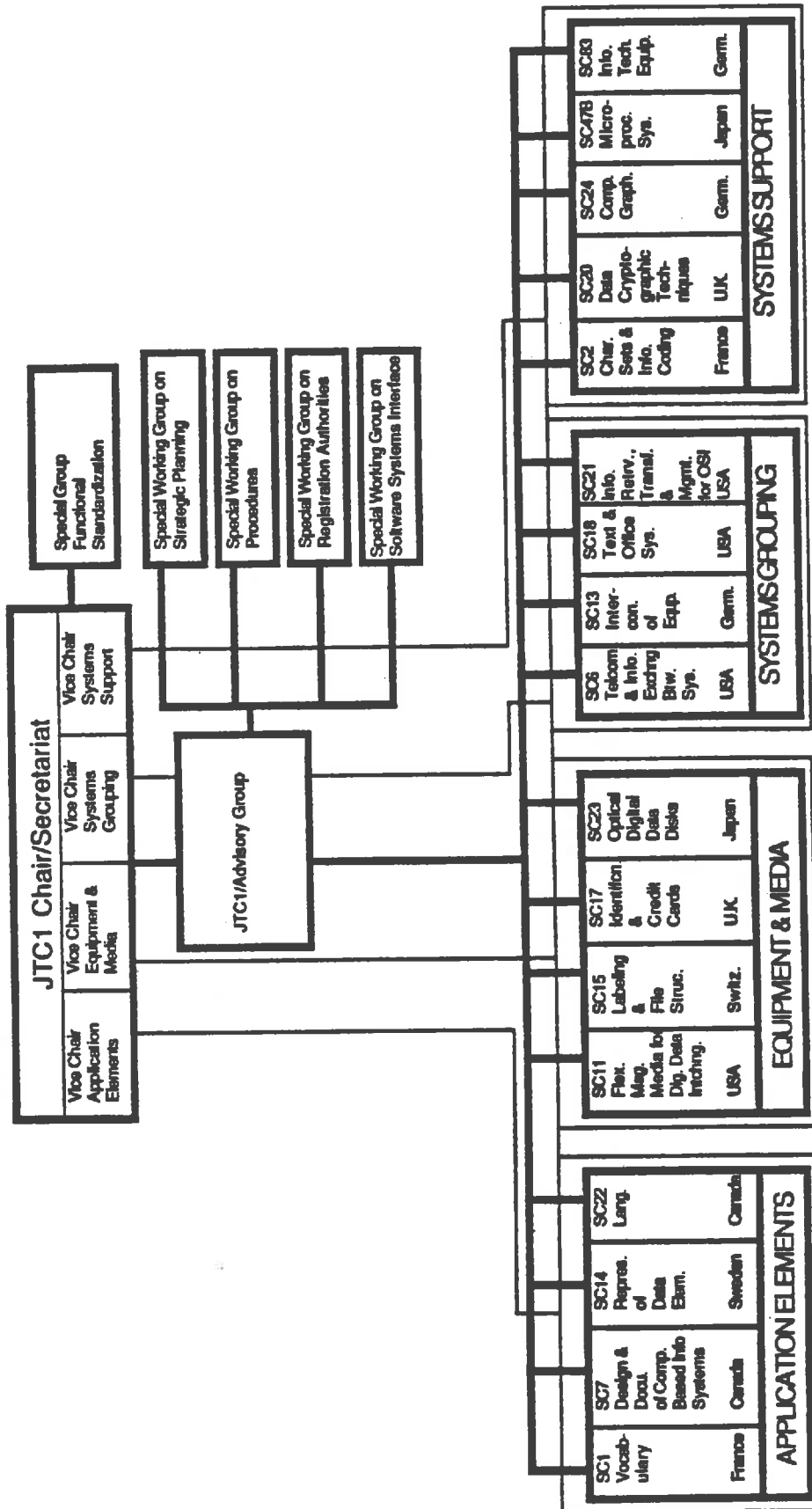


## DETAILED X3 TECHNICAL COMMITTEE ORGANIZATIONAL CHART

<u>X3 SUBCOMMITTEE TAG'S</u>	<u>TECH. ADVISORY GROUP (TAG) RESPONSIBILITY</u>	<u>X3 TECHNICAL COMMITTEE</u>	<u>TECH. ADVISORY GROUP (TAG) RESPONSIBILITY</u>
U.S. TAG TO SC21 U.S. TAG TO SC22	ISO/IEC/JTC1 SC21 ISO/IEC/JTC1 SC22		
<b>X3 TECHNICAL COMMITTEES</b>		<b>K. DOCUMENTATION</b>	
<b>A. RECOGNITION</b>		<b>L. DATA REPRESENTATION</b>	
X3A1 OCR & MICR		X3K1 Computer Documentation	
X3A1.1 Font Design		X3K5 Vocab. for Info. Proc. Sys.	ISO/IEC/JTC1 SC1
X3A1.2 OCR Supplies & Forms			
X3A1.3 Image Def. & Measurement			
<b>I. MEDIA</b>		<b>M. DATA COMMUNICATION</b>	
X3B5 Digital Magnetic Tape	ISO/IEC/JTC1 SC11/SC15	X3S3 Data Communication	ISO/IEC/JTC1 SC6
X3B6 Instrumentation Tape	ISO/IEC/JTC1 WG2	X3S3.1 Data Comm. Planning	ISO/IEC/JTC1 SC6
X3B7 Magnetic Disks		X3S3.2 Data Comm. Vocabulary	ISO/IEC/JTC1 SC6
X3B8 Flexible Disk Cartridges	ISO/IEC/JTC1 SC11/SC15	X3S3.3 Network Layer	ISO/IEC/JTC1 SC6 WG2
X3B8.1 Track Formats for FDC'S	ISO/IEC/JTC1 SC11/SC15	X3S3.4 Control Procedures	ISO/IEC/JTC1 SC6
X3B9 Paper Forms/Layouts		X3S3.5 Comm. Sys. Performance	ISO/IEC/JTC1 SC6
X3B10 Credit/ID Cards	ISO/IEC/JTC1 SC17	X3S3.7 Pub. Data Network Access	ISO/IEC/JTC1 SC6
X3B10.1 Integrated Circuit Cards	ISO/IEC/JTC1 SC17 WG4		
X3B10.2 Revision of X4.18	ISO/IEC/JTC1 SC17		
X3B10.3 Min. Phys. Req'ts/Savingsbooks	ISO/IEC/JTC1 SC17		
X3B10.4 Optically Encoded Card Media	ISO/IEC/JTC1 SC17		
X3B11 Optical Digital Data Disks	ISO/IEC/JTC1 SC23/SC15		
<b>H &amp; J. LANGUAGES</b>		<b>T &amp; V. SYSTEMS TECHNOLOGY</b>	
X3H2 Database	ISO/IEC/JTC1 SC21 WG3	X3T1 Data Encryption	ISO/IEC/JTC1 SC20
X3H3 Computer Graphics	ISO/IEC/JTC1 SC24	X3T2 Data Interchange	ISO/IEC/JTC1 SC21 WG6, ISO/IEC/JTC1 SC22 WG11, ISO/IEC/JTC1 SC21 WG6
X3H3.1 PHIGS	ISO/IEC/JTC1 SC24	X3T5 OSI	ISO/IEC/JTC1 SC21 WG1
X3H3.2 FDVT	ISO/IEC/JTC1 SC24	X3T5.1 OSI Architecture	ISO/IEC/JTC1 SC21 WG4
X3H3.3 Virtual Device Interface	ISO/IEC/JTC1 SC24	X3T5.4 OSI Mgmt. Protocols	ISO/IEC/JTC1 SC21 WG5
X3H3.4 Language Binding	ISO/IEC/JTC1 SC24	X3T5.5 Applic. & Proc. Layers	ISO/IEC/JTC1 SC13
X3H3.5 GKS	ISO/IEC/JTC1 SC24	X3T9 I/O Interface	ISO/IEC/JTC1 SC13
X3H3.6 Display Mgmt. Graphcl. Dev.	ISO/IEC/JTC1 SC24	X3T9.2 Lower Level Interface	ISO/IEC/JTC1 SC13
X3H4 Info. Resources & Dict.	ISO/IEC/JTC1 SC21	X3T9.3 Device Level Interface	ISO/IEC/JTC1 SC13
X3H4.1 IRDS Reference Model	ISO/IEC/JTC1 SC21	X3T9.5 Local Dist. Data Intfc.	ISO/IEC/JTC1 SC13
X3H4.2 IRDS Exten. Software Intfc	ISO/IEC/JTC1 SC21	X3T9.6 Cartridge Tape Drives	ISO/IEC/JTC1 SC18
X3J1 PL/I	ISO/IEC/JTC1 SC22	X3V1 Text: Off & Pub. Sys.	ISO/IEC/JTC1 SC18 WG1
X3J1.3 General Purpose Subset	ISO/IEC/JTC1 SC22	X3V1.1 User Req. M.S.T.	ISO/IEC/JTC1 SC18 WG3
X3J2 BASIC	ISO/IEC/JTC1 SC22 WG8	X3V1.3 Document Architecture	ISO/IEC/JTC1 SC18 WG4
X3J3 FORTRAN	ISO/IEC/JTC1 SC22 WG5	X3V1.4 Text Interchange	ISO/IEC/JTC1 SC18 WG5
X3J4 COBOL	ISO/IEC/JTC1 SC22 WG4	X3V1.5 Content Architecture	ISO/IEC/JTC1 SC18 WG8
X3J7 APT	ISO/IEC/JTC1 SC3	X3V1.8 Text. Descr. & Proc. Langs.	ISO/IEC/JTC1 SC18 WG9
X3J7.1 Processor Languages	ISO/IEC/JTC1 SC3	X3V1.9 User Sys. Intfc & Symbols	ISO/IEC/JTC1 SC18
X3J7.2 Postprocessor Languages	ISO/IEC/JTC1 SC3	X3V1.10 Font Resources	
X3J7.3 Latho Languages	ISO/IEC/JTC1 SC3		
X3J7.4 Robotics Language	ISO/IEC/JTC1 SC3		
X3J9 PASCAL	ISO/IEC/JTC1 SC22 WG2		
X3J10 APL	ISO/IEC/JTC1 SC22 WG3		
X3J11 C Language	ISO/IEC/JTC1 SC22 WG14		
X3J12 DIBOL			
X3J13 Common LISP	ISO/IEC/JTC1 SC22 WG16		
X3J14 Programming Lang. FORTH			
		<b>SPARC STUDY GROUPS</b>	
		DS5SG Data Base Systems	ISO/IEC/JTC1 SC21
		SK5SG Software Systems Interface	ISO/IEC/JTC1 SWG SS1

This Chart is Organizational only & does not reflect the Standards Approval Process

JTC1 STRUCTURE:



**KEY:**  
 ——— JTC1 STRUCTURE  
 ——— Coordination responsibilities of Vice Chairmen

54

# Names and Addresses of Chairmen of U.S. Held Secretariats for ISO/IEC JTC1 Subcommittees

54

## JTC1

Mr. L. John Rankine  
Dir., Stds. & Data Security  
IBM Corporation  
2000 Purchase Street  
Purchase, NY 10577-2597  
914-697-7230

## JTC1/SC6

Mr. Harold Folts  
OMNICOM, Inc.  
115 Park Street, SE  
Vienna, VA 22180  
703-281-1135

## JTC1/SC11

\*Mr. Richard Steinbrenne.  
AT&T  
One Whippany Drive  
Whippany, NJ 07931  
201-386-7053

## JTC1/SC18

Mary Anne Gray  
IBM Corporation  
2000 Purchase Street  
Purchase, NY 10577-2597  
914-697-7224

## JTC1/SC21

Mr. Richard desJardins  
CTA  
7927 Jones Branch Drive  
McLean, VA 22102  
703-848-2700

\*Acting Chair

## Names and Addresses of Working Group Convenors

### SC1/WG4. Fundamental Terms

Richard W. Batey  
UNISYS  
P.O. Box 500 C1-NE6  
Blue Bell, PA 19424  
215-542-2709

### SC17/WG5. Registration Management Group

Beryl Ann Barber  
Chase Manhattan Bank, N.A.  
3 Shortlands Hammersmith  
London W68RZ UK  
01-747-4453

### SC22/WG3. FORTRAN

Jeanne Martin  
Lawrence Livermore Laboratories  
P.O. Box 808L-300  
Livermore, CA 94550  
415-422-3753

### SC2/WG6. Control Functions

Macklin W. Bishop  
IBM Corporation  
11400 Burnet Road  
Bldg. 803, Dept. D34  
Austin, TX 78758  
512-823-2005

### SC18/WG8. Text Descript. & Processing Languages

James D. Mason  
Oak Ridge National Laboratory  
P.O. Box X, 4500-S  
Oak Ridge, TN 37831  
615-574-6973

### SC22/WG4. COBOL

Mabel Vickers  
National Bureau of Standards  
Technology Bldg. 225, Room A-268  
Gaithersburg, MD 20899  
301-975-3277

### SC6/WG1. Data Link Layer

David E. Carlson  
AT&T  
Room 3N 610  
Crawfords Corner Road  
Holmdel, NJ 07733  
201-949-7503

### SC20/WG3. Use of Encyph. Techniques in Comm. Archit.

Vacant

### SC22/WG9. ADA

Vacant

### SC7/WG1. Symbols, Charts & Diagrams

Richard Werling  
6308 Beachway Drive  
Falls Church, VA 22044

### SC21/WG6. OSI Session, Pres. & Common Application Services

Paul D. Bartoli  
AT&T  
Room 1M-311  
Crawfords Corner Road  
Holmdel, NJ 07733  
201-949-5965

### SC22/WG11. Binding Techs. for Programming Languages

Donald F. Nelson  
Tandem Computer, Inc.  
10555 Ridgeview Court  
Cupertino, CA 95014  
408-996-6340

### SC17/WG3. Passport Cards

Vacant

### SC22/WG15. POSTX

Jim Isaak  
Digital Equipment Corporation  
MK02-2/B05  
Continental Boulevard  
Merrimack, NH 03054-0430  
603-884-1913

107(\*) JCA-10

X3/SD-9

MAY 1987

55

ACCREDITED STANDARDS COMMITTEE\*  
X3-INFORMATION PROCESSING SYSTEMS

---

POLICY AND GUIDELINES

---

\*Operating Under the Procedures of the American National Standards Institute

SECRETARIAT:

Computer and Business Equipment Manufacturers Association

Insert  
Logo

# X3 Standing Documents

This document is one of a series, developed by X3 and the X3 Secretariat, which provides a "data base" of information on Accredited Standards Committee X3 - Information Processing Systems. Each document is updated periodically on an individual basis.

The series is intended to serve several purposes:

- o To describe X3 and its program to inquirers
- o To inform committee members of the organization and operation of X3
- o To provide a system of orderly administration incorporating the procedures required by ANSI together with supplements approved by the X3 Secretariat, for the guidance of X3 officers, members, subgroups and the Secretariat staff.

The series of Standing Documents consists of the following:

- X3/SD-0      Informational Brochure - September 1985
- X3/SD-1      Master Plan - May 1987
- X3/SD-2      Organization & Procedures - October 1985
- X3/SD-3      Project Proposal Guide - May 1987
- X3/SD-4      Projects Manual - Updated Quarterly
- X3/SD-5      Standards Criteria - September 1984
- X3/SD-6      Membership and Officers - Updated Quarterly
- X3/SD-7      Meeting Schedule and Calendar - Updated Quarterly
- X3/SD-9      Policy and Guidelines - May 1987
- X3/SD-10     X3 Subgroup Annual Report Form -October 1985

---

Corrections and suggestions for improvement will be welcomed, and should be addressed to:

Director,  
X3 Secretariat/CBEMA  
311 First Street, NW  
Suite 500  
Washington, DC 20001-2178

TABLE OF CONTENTS

55

	<u>PAGE</u>
INTRODUCTION	i
SECTION 1 - EXTERNAL POLICY AFFECTING X3	1
1.1 CBEMA STANDARDS PROGRAM POLICY STATEMENT	1
1.2 ANSI Policy for Voting on ISO Draft International Standards	3
Recommended Language to be Used in Explaining ANSI Votes on ISO Draft International Standards	4
1.3 ANSI Policy on Identification of Metric Standards	5
1.4 ANSI Policy on Dimensions and Quantities in American National Standards	5
1.5 CBEMA Policy on 1988 Secretariat Service Fees for Participation in X3	6
1.6 ANSI's Patent Policy and Approved Statement of Patent Holder	7
1.7 CBEMA Publication and Copyright Policy	8
Statement of Patent Holder Concerning the Use of Patented Device or Design in Conjunction with an American National Standard	9
SECTION 2 - X3 INTERNAL POLICY	11
2.1 Policy for Programming Language Standards Concerning Order of Expression Evaluation	11
2.2 Use of ANSI/IEEE 268-1982, Metric Practice, for Conversion Rules	11
2.3 Criteria for Selection of TC97 Advisory Group Members	11
SECTION 3 - GUIDELINES	12
3.1 Intra-language Compatibility Guidelines	12-16
SECTION 4 - BIBLIOGRAPHY	17

The X3/SD-9, Policy and Guidelines, is a listing of external and internal policies which affect X3 and guidelines to assist X3 technical committees in their programs of work.

In addition, the SD-9 provides a mechanism to ensure that the X3 community, particularly new participants, will be aware of documents and any revisions that affect X3. Please address the Chairman of the Secretariat Management Committee should you have any suggestions for additions.

This standing document is divided into four sections:

- Section 1      External Policy Affecting X3 - This section includes policy made by ANSI, ISO and CBEMA which directly affect X3.
- Section 2      X3 Internal Policy - This section includes policy voted upon by X3 to control its own work.
- Section 3      Intra-Language Compatibility Guidelines - This section lists the guidelines established to reflect the policy and philosophy of the U. S. with relationship to the TC97/SC22 activities dealing with intra-language compatibility issues.
- Section 4      Bibliography - Is a listing of documents from ANSI, ISO, CBEMA or other sources which might be useful to TC's within their program of work. It will be maintained by the Secretariat Management Committee.

SECTION 1 - EXTERNAL POLICY AFFECTING X3

1.1. CBEMA STANDARDS PROGRAM POLICY STATEMENT

The Standards Program identifies, analyzes and develops CBEMA positions on vital issues in the area of standards, certification, validation and accreditation which bear on the free and open market competition for our industry's products. It acts only when there is a common interest of the membership. Areas include Ergonomics, Electromagnetic Interference and Susceptibility, Environment, Health, Safety, and Hardware, Software and Systems Functional and Performance Characteristics. As an organization, CBEMA does not develop or promulgate standards, but members actively participate in the voluntary standards process to develop National and International standards. CBEMA'S goal is to achieve acceptance of the voluntary standards process as the primary standardization mode domestically and internationally.

To carry out this program successfully it is essential to adhere to the following policy statements:

1. CBEMA supports a standards policy which promotes technological innovation and free marketplace competition, and which minimizes government legislation, regulations or oversight of standards-related activities in the information systems industry.
2. CBEMA supports development of voluntary consensus standards that are beneficial to industry and the general public. These standards may increase productivity and efficiency in industry, expand opportunities for international trade, conserve resources and improve health and safety. When properly conducted, the voluntary consensus process will yield useful standards which will not:
  - o conflict with other standards or make them redundant.
  - o suppress free and fair competition.
  - o inhibit technological innovation and progress.
  - o impede safer and less expensive products.
  - o otherwise adversely affect trade, commerce, health or safety.

These standards should be based on functional requirements rather than design specifications.

3. CBEMA supports the voluntary standards process in the United States. CBEMA also supports the American National Standards Institute and its procedures for management and coordination of American National Standards.
4. CBEMA supports development of international standards through the voluntary process as exemplified by IEC and ISO.
5. CBEMA supports harmonization of national and international standards.



- 6. CBEMA supports adoption of voluntary consensus standards by governments whenever feasible and consistent with law and regulation, in preference to government developed standards.
- 7. CBEMA supports government participation in the voluntary consensus standards development process.
- 8. CBEMA supports the GATT Code on Standards.
- 9. CBEMA supports the principle of self certification (verification), whereby the manufacturer conducts tests on his own products and certifies/verifies that they are in compliance with the relevant standards. In lieu of the preferred method, CBEMA supports reciprocal testing and certification to a single set of standards and procedures so that a product can be certified by one test house and accepted universally.
- 10. CBEMA supports development and implementation of standards, both national and international, relating to any area which can improve health and safety.

CBEMA believes, however, that such standards should not impose requirements that unnecessarily increase the cost to the users without providing a correspondingly proportionate improvement in the health and safety aspects. Overly stringent requirements have the effect of restricting the use of a product and thereby defeating the purpose of its development.

CBEMA has paid careful attention to the health and safety aspects of our industry's products for many years. We have also been very alert to the health and safety of our own industry employees, including production (factory) employees, office employees, and all other categories. These considerations have been and continue to be demonstrated by our active participation in standards activities and with regulatory agencies throughout the world.

CBEMA has established committees on environment, health, safety and human factors in the following areas:

- |                           |                         |
|---------------------------|-------------------------|
| Acoustic Radiation        | Human Factors           |
| Electromagnetic Radiation | Hazardous Materials     |
| Laser Radiation           | Toxic Substances        |
| Electrical Safety         | Safety of the Workplace |
| Power Interface           |                         |

1.2 ANSI Policy for Voting on ISO Draft International Standards

55

With respect to votes on ISO Draft International Standards (DIS):

- A. ANSI should vote in the affirmative:
1. if there is an existing national standard (a standard generally accepted within the United States) not in conflict with the DIS; that is, for example, if a product or procedure meeting the provisions of either of the two has a reasonable possibility of satisfying the other; or,
  2. if any conflict between the DIS and an existing national standard is small, and the Technical Advisory Group believes the national standard will be modified to conform at an early date; or,
  3. if no U.S. document exists, and the TAG believes that the DIS or something in harmony with it would be accepted as a national standard.
- B. ANSI should vote "no" if the DIS is in clear conflict with an existing national standard; that is if a product or procedure meeting the provisions of one of the two has no reasonable possibility of satisfying the other.
- C. If a national standard exists which differs from the DIS only because either includes additional or more stringent requirements, the vote to be cast by ANSI may be either "yes" or "no" depending upon the history of the U.S. participation in the preparation of the document. If U.S. delegations and U.S. comments on earlier drafts have argued unsuccessfully for inclusion in the ISO Standard of the provisions of the national standard the ANSI vote ought to be "no". If the U.S. participation has not been an active advocacy of the national standard, ANSI may vote "yes" but comment pointing out the additional requirements that must be met to satisfy our national standard.
- D. If at all possible ANSI should vote on all Draft International Standards prepared by Technical Committees for which ANSI is a "P" member.
- E. ANSI should include in its voting document all comments pertinent to the harmony between the ISO document and any existing U.S. standard.
- F. Exceptions to the above stated voting policy shall be permitted only upon recommendation of the TAG Chairman involved and with the concurrence of the responsible Standards Management Board Chairman.
- G. In explaining votes, the recommended language contained in Appendix A to this document should be used.

**NOTE:** This ANSI voting policy has been approved by the International Standards Council and the Executive Standards Council; it forms a part of the Operating Procedures for the International Standards Activities of ANSI. All Technical Advisory Group are required to follow this policy when recommending ANSI votes on ISO Draft International Standards.

**Recommended Language to be Used in  
Explaining ANSI Votes on ISO  
Draft International Standards**

---

The following are examples of language to be used in explaining an ANSI vote. If they are not suitable to cover a given situation, the TAG Chairman may use other language.

**Affirmative Vote**

1. None
2. If approved by ISO, the document will probably serve as the basis of a national standard in the United States.
3. If approved by ISO, the document will probably serve as the basis of a national standard in the United States, which may contain some changes or additions.
4. ANSI believes the document represents the best international agreement that can be reached at this time, but even if it is approved by the ISO it is not likely to have much application in the United States because national circumstances prevent its adoption now or in the immediate future. (If at all possible, specific examples of the special "national circumstances" should be given).

**Negative Vote** - (Under ISO statutes, reasons must be given for a negative vote.)

1. If approved by ISO, the document cannot be accepted as a national standard in the United States because so many exceptions or additions will be required to conform with our practices and regulations.
2. ANSI believes that the following substantive technical changes are necessary: (TAG Chairman to supply them).
3. Same as 4. above.

**1.3 ANSI Policy on Identification of Metric Standards**

A method for identifying metric standards has been approved by the Executive Standards Council of the American National Standards Institute, to take effect immediately. This method consists of the use of uppercase "M" after the designation of a standard (for example, ANSI/EIA RS-989M) or the term "metric" in the title of a standard (for example, "Metric Thread Fuel Injection Tubing Connections").

Responsibility for determining whether a standard should carry the metric designation rests with the promulgating committee and is based on two criteria:

(1) The standard must be expressed in metric units. It may include customary units also, but metric units must be the primary and controlling units. Many standards are dimensionless, that is, they do not use units of measure. Such standards are applicable in both customary and metric environments and use of the metric designation is permitted.

(2) The standard must be intended for use in a metric environment and must be the metric version of the item in question. Both "hard converted" and "soft converted" standards qualify for the metric identifier if they are intended for use as metric standards. Examples of hard converted standards are ANSI B32.3 and ANSI B32.4, which cover metric sizes for flat and bar metal products. Examples of soft converted standards are American Petroleum Institute Standards for tubular goods (casing, tubing, and drill pipes), which will continue in use but which will be converted to their millimeter equivalents without changing the size of the products.

The new metric identification will be of particular value for ordering standards, listing standards in catalogs, and referencing standards in contracts. While it is desirable that this identification be applied to all metric standards, it is particularly important that it be applied to all American National Standards.

**1.4 ANSI Policy on Dimensions and Quantities in American National Standards**

Dimensions and quantities should be given in customary units or in SI units, or in both, whichever in the opinion of the standards developing group appears to be in the best interests of the users of the standard.

Effective July 7, 1975, all proposed American National Standards shall include SI units as well as other appropriate units, where practicable. Where both are given, the standards developing group should determine the order in which they are presented.

Subsequent to the effective date indicated above, submitters of proposed American National Standards that do not include SI units shall indicate the reason(s) why it is not practicable to do so.

1.5 CBEMA Policy on 1988 Secretariat Service Fees for Participation in X3

Due to rising costs and ever-increasing demands for improved support of standardization efforts, it is necessary to increase the level of funds provided by X3 service fees. X3 members' service fees have remained stable for several years.

The Secretariat Management Committee has supported the concept of a sliding scale service fee for the past several years and has provided CBEMA with examples to be considered. X3 Technical Committee service fees are \$200 per principal and one alternate, \$150 per second and/or additional alternates, and observers.

1988 and 1989 X3 SECRETARIAT SERVICE FEES

All member organizations of X3 would fall into one of the following service fee categories:

Producers

Government

Users, Classification A -- Companies listed in Fortune's 500 or Fortune's Service 500 list.

Organizations -- User Groups and Professional Societies

Users, Classification B -- Companies other than those listed in Fortune's 500 or Fortune's Service 500 list.

These classification assignments for 1988 will be based on the latest version of the Fortune 500 list, May, 1987.

Service fees for 1988 and 1989 relative to the above categories are:

Producers	: \$5,750
Government	: \$5,750
User, Classification A	: \$4,600
Organizations	: \$3,450
User, Classification B	: \$1,150
Observers	: \$1,725

These figures, in conjunction with the service fees collected from the Technical Committee participants, other income such as document sales, and the CBEMA contribution, will allow responsible financial coverage of the Secretariat activities.

The service fee invoices will be issued October 15, due and payable by January 15, 1988. As in the past, X3 members may apply for waivers of any percentage of the invoiced amount by addressing a letter to the Chair of the Secretariat Management Committee, stating the reason for such a request and supplying any supporting documentation which could assist in the case-by-case review and recommendation to CBEMA. Waiver requests are due at the Secretariat offices not later than November 10.

1.6 ANSI's Patent Policy and Approved Statement of Patent Holder

1. Inclusion of Patents in American National Standards

There is no objection in principle to drafting a proposed American National Standard in terms that include the use of a patented item, if it is considered that technical reasons justify this approach.

If the Institute receives a notice that a proposed American National Standard may require the use of a patented invention, the procedures in Sections 2 through 5 shall be followed.

2. Statement from Patent Holder

Prior to approval of such a proposed American National Standard, the Institute shall receive from the patent holder (in a form approved by the Institute) either: assurance in the form of a general disclaimer to the effect that the patentee does not hold and does not anticipate holding any invention whose use would be required for compliance with the proposed American National Standard or assurance that:

(1) A license will be made available without compensation to applicants desiring to utilize that license for the purpose of implementing the standard, or

(2) A license will be made available to applicants under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

The terms and conditions of any license shall be submitted to ANSI for review by its counsel, together with a statement of the number of independent licensees, if any, which have accepted or indicated their acceptance of terms and conditions of the license.

3. Record of Statement

A record of the patent holder's statement (and a statement of the basis for considering such terms and conditions free of any unfair discrimination) shall be placed and retained in the files of the Institute.

4. Notice

When the Institute receives from a patent holder the assurance set forth in 2(1) or 2(2), the standard shall include a note as follows:

NOTE: The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights.

By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. The patent holder has, however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

## 5. Responsibility for Identifying Patents

The Institute shall not be responsible for identifying all patents for which a license may be required by an American National Standard or for conducting inquiries into the legal validity or scope of those patents which are brought to its attention.

### 1.7 CBEMA Publication and Copyright Policy

CBEMA shall copyright all publications other than fillings, testimony, minutes, agendas, correspondence, memoranda and meeting notices. This will include "white papers", technical reports, periodical journals, newsletters, etc. In those instances where there is concern over the possibility of copyright infringement, e.g. for commercialized publications, the copyright shall be registered in CBEMA's name with the Copyright Office.

The granting of rights to reproduce, reprint or otherwise use copyrighted material will be handled on the basis of purpose for original publication. In the instance of periodical journals, reprint rights will be granted upon request. In the instance of commercialized publications, such rights may be licensed under a standard form agreement which specifies purpose, number of copies, royalty structure and limits on reproduction.

In any event one copy of the publication and rights to reproduce, reprint or otherwise use the publication, camera ready copy or magnetic media will be available to each CBEMA member, provided that the purpose for reproducing, reprinting or other use is not for commercial use.

All matters concerning CBEMA publication policy will be referred to the CBEMA President for resolution.



**STATEMENT OF PATENT HOLDER CONCERNING THE USE OF PATENTED DEVICE OR DESIGN IN CONJUNCTION WITH AN AMERICAN NATIONAL STANDARD**

**Note:** This form is to be used to record the statement of a patent holder whose patented device or design (pending or approved) may have to be used by a person or organization complying with an American National Standard. This statement is filed and retained pursuant to Section 7.4 of the American National Standards Institute Procedures for Management and Coordination of American National Standards.

1. **Name of Patent Holder:** \_\_\_\_\_

**Address:** \_\_\_\_\_

**Telephone:** \_\_\_\_\_

**Contact:** \_\_\_\_\_  
(Name) (Title)

2. **Number and Description of Patent(s)** (use extra sheet if necessary)

**Patent Number(s):** \_\_\_\_\_  
(Attach copy of patent application)

**Description:** \_\_\_\_\_

**List Claims Pertinent to Compliance with Standard:** \_\_\_\_\_

**Date of Expiration:** \_\_\_\_\_

Pending  Approved  (check one)

If Pending, indicate date of application: \_\_\_\_\_

If Patent Holder is a transferee of original claimant, attach the transfer agreement.

3. **Name of Proposed ANSI Standard:** \_\_\_\_\_

**List sections of standard to which patent claims relate:** \_\_\_\_\_



4. Terms and Conditions of License

Are there any licensees \_\_\_\_\_ If so, how many \_\_\_\_\_

55

Please attach a copy of license agreement, if any.

5. Statement of Patent Holder

Please check one of the following statements:

- (a) A license shall be made available without compensation to applicants desiring to utilize the license for the purpose of implementing the standard or, (check one)
- (b) A license shall be made available to applicants under reasonable terms and conditions that are demonstrably free of any unfair discrimination.
- (c) An assurance of non-assertion will be filed with the Patent Office by the effective date of ANSI approval of the listed standard.

Signed: \_\_\_\_\_  
(Name) (Title)

Company: \_\_\_\_\_

Date: \_\_\_\_\_

SECTION 2 - X3 INTERNAL POLICY

2.1 Policy for Programming Language Standards Concerning Order of Expression Evaluation

In all subsequent language standards, or revisions thereof, a statement will be included explaining the order of evaluation of expressions (whatever that order may be) and, if permitted by the language, a statement specifying allowable deviations from the order.

2.2 Use of ANSI/IEEE 268-1982, Metric Practice, for Conversion Rules

All Committees drafting standards using SI (metric) units must use the conversion rules as defined by ANSI/IEEE 268-1982.

2.3 Criteria for Selection of TC97 Advisory Group Members

In selecting members for the TC97 Advisory Group, the IAC used the following criteria for developing the recommended list for X3's approval:

1. An attempt is made to keep the U. S. representation to a reasonable number.
2. Membership priority is given to members of IAC.
3. An attempt is made to establish continuity of the E.G. membership.
4. IAC members are selected by seniority on the committee and desire to serve.
5. An attempt is made to broaden the experience level of other X3/IAC participants by rotating them onto the A.G.
6. Chairs of the U. S. - Held Secretariats may attend the A. G. independent of the total delegation.
7. The A. G. membership may be supplemented by issue-specific experts attending individual meetings.

**3.1 INTRA-LANGUAGE COMPATIBILITY GUIDELINES**

This section includes guidelines for X3 for use in administering its work. After the development and approval of the initial standard for a programming of systems language, the X3 Technical Committee responsible for carrying out that project must determine the future method to be used to support standardization of that programming or systems language. This method may be:

- a. To revise the current standard, thereby creating a new standard that will replace the currently existing standard, or to supplement the current standard such that change to the technical content is made.
- b. To develop a new "companion" or "supplemental" standard that will not replace or change the technical content of the currently existing standard, but rather is another in a family of standards for that particular programming or systems language.
- c. To reaffirm the current standard.
- d. To withdraw the current standard with no replacement.

Although a Technical Committee developing the initial standard for a programming or systems language should be aware of these Guidelines, these Guidelines apply only to the method covered by paragraph a above.

As X3 Technical Committee revised programming and systems language standards, conflicts may arise between the following two goals:

- a. Upward compatibility to maintain programmer productivity and protect investment in existing software and programmers;
- b. Increased programmer productivity through improved programming technology; i.e., the new technology may not preserve compatibility..

The purpose of these Guidelines is to assist Technical Committees in balancing these goals as they revised standards for programming and systems languages. It is not to dissuade committees from adopting any incompatible change to a standard. These Guidelines provide an objective means to determine the effect of proposed incompatibilities with prior standards and variations from existing practice. Additional visibility of the decision process ensures that uniform and widely accepted criteria are applied by X3 Technical Committees as they develop and revise standards.

These Guidelines specify that the decisions for each incompatibility be documented using the following criteria:

- a. Rationale, with benefits and costs, for proposed change.
- b. Type of difference between versions.
- c. Conversion complexity.

- d. Breadth of impact of incompatibility or variation from existing practice.
- e. Conversion guidance, where appropriate.

### 3.1.1 SCOPE AND APPLICABILITY

These Guidelines apply whenever an X3 programming or systems language standard, language binding, appendix, or optional section is being revised.

### 3.1.2 ANALYSIS REQUIREMENTS

For each addition, deletion or modification that represents a potential incompatibility from the last existing standard, X3 Technical Committees applies the following criteria:

- a. Rationale, with benefits and costs, for proposed change.
- b. Type of difference between versions.
- c. Conversion complexity.
- d. Breadth of impact of incompatibility or variation from existing practice.

### 3.1.3 RATIONALE FOR PROPOSED CHANGE

The rationale for the changes includes:

- a. Specific benefits, and how the benefits result from the change.

Benefits may fall into such categories as improved programming practice, better portability, better machine performance, elimination of ambiguity, or improved consistency and clarity of the language specification.

- b. Cost (other than those directly associated with compatibility; these are discussed in the next three subsections).

Costs may fall into such categories as usability, performance, or ease of learning.

### 3.1.4 TYPES OF DIFFERENCE BETWEEN VERSIONS

The Technical committee determines which of the following classifications best describes the way in which the original language feature may be adversely affected by the proposed change. The following are listed in order of decreasing severity.

3.1.4.1 Changes To Semantics of Well Defined Feature - The semantic specifications of a feature for which the original document guarantees a certain reasonably precise result is changes. The feature remains syntactically valid, but a program may now produce different results.

55

3.1.4.2 Deletion of Well Defined Feature - An originally well-defined feature is rendered invalid by the new specification.

3.1.4.3 Deletion of Ill Defined Feature - An originally ill-defined feature is rendered invalid by the new specification.

3.1.4.4 Clarification of Ill Defined Feature - This category covers the definition of a language construct whose interpretation was questionable under the base document. Strictly speaking, it is not an incompatibility, since no guarantee has been withdrawn, but it is included here for completeness.

3.1.4.5 Deletion of a feature which was introduced in the current standard but has had few, if any, correct implementations, has had little or not correct use, and was an unfortunate mistake from the beginning.

3.1.4.6 Deletion of Obsolescent Feature - A feature which has been designated as obsolescent in the current specification is deleted in the new specification.

Sections 3.2 and 3.3 do not apply to actions falling into the categories covered by paragraphs 3.1.4.5 and 3.1.4.6 above.

## 3.2 CONVERSION COMPLEXITY

The Technical Committee estimates the level of difficulty entailed in converting affected programs so as to conform to the new standard. At least four levels of difficulty may be distinguished. From the standardization point of view, the following are listed in order of decreasing conversion effort.

- 3.2.1 No Possible Translation - There is no feasible way to perform the original function using the new standard.
- 3.2.2 Semantic Transformation - The original function can still be performed using the language, but human translation, based on knowledge of the purpose of the program, is required.
- 3.2.3 Significant Syntactic Transformation - A mechanical translation is feasible, but it involves some analysis of the program structure as a whole, and/or a significant amount of code may be generated in place of the original code.
- 3.2.4 Simple Syntactic Transformation - Old statements can be mechanically transformed to the new syntax with little or no knowledge of the rest of the program or its purpose. Changing user-defined identifiers because of the introduction of new reserved words falls into this category.

3.3 PROPORTION OF PROGRAMS AFFECTED

The Technical Committee includes in the draft standard (as an appendix):

- a. An explicit list of known incompatibilities with annotated examples, as appropriate.
- b. The analysis for each incompatibility based on the decision criteria in Section 3.1.2.
- c. The documentation required in paragraphs 3.5 through 3.8.

3.4 ACTION REQUIREMENTS TABLE

A Technical Committee provides the documentation and takes the actions specified by the sections referenced in the following table.

Type of Difference	Type of Difference	Action
Change well-defined (3.1.4.1)	Change well-defined (3.1.4.1)	3.6 and 3.7
Delete well-defined (3.1.4.2)	Delete well-defined (3.1.4.2)	3.5 and 3.7
Delete ill-defined (3.1.4.3)	Delete ill-defined (3.1.4.3)	3.5 and 3.8
Clarify ill-defined (3.1.4.4)	Clarify ill-defined (3.1.4.4)	3.8
Delete obsolescence (3.1.4.6)	Delete obsolescence (3.1.4.6)	3.8

3.5 OBSOLESCENT CATEGORY

The deleted feature is placed in the obsolescent category, whereby the feature is included in the standard but with notification that it will be withdrawn in the next revision.

3.6 TRANSITION SEMANTICS

The Technical Committee provides for the transition between the old and the new definitions of the same language feature in the following ways:

- a. The standard may recommend implementations to make both the old and new interpretations available to the user; e.g., through the use of a switch (which itself may or may not be part of the language).
- b. If the Technical Committee judges that the costs of such measures outweigh the benefits, it may simply adopt the new interpretation.

3.7 PROVIDE CONVERSION GUIDANCE

The Technical Committee provides written guidance for the conversion of programs to the new standard. For example:

- a. An algorithm that is simple but detailed enough to be understood by a reasonably informed user of the language.
- b. A narrative of the conversion process.
- c. In case where a well-defined conversion process is not feasible, the Technical committee must provide as much conversion information as possible.

3.8 NO FURTHER ACTION

The Technical Committee is not obliged to take any action beyond the inclusion of the change in the incompatibility list described in section 3.4.

3.7 PROVIDE CONVERSION GUIDANCE

The Technical Committee provides written guidance for the conversion of pr to the new standard. For example:

- a. An algorithm that is simple but detailed enough to be understood a reasonably informed user of the language.
- b. A narrative of the conversion process.
- c. In case where a well-defined conversion process is not feasible, Technical committee must provide as much conversion information possible.

3.8 NO FURTHER ACTION

The Technical Committee is not obliged to take any action beyond the inclusion of the change in the incompatibility list described in section 3.4.



SECTION 4 - BIBLIOGRAPHY

- ANSI Procedures for Development and Coordination of  
American National Standards
- ANSI Guide to Submitting Standards to ANSI for Approval
- ANSI Constitution and By-laws
- ANSI Operating Procedures of the Board of Standards Review
- ANSI Appeals Board Operating Procedures
- ISO Directives for the Technical Work of ISO

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

Doc. No.: **X3/88-039-X.S.H**  
 Date: January 11, 1988  
 Project:  
 Ref. Doc.:  
 Reply to:

ATTENTION TC OFFICERS AND SPARC LIAISONS

TO: X3, SPARC, SMC  
 Officers, XI/TC's and SPARC/SC's

SUBJECT: Transmittal of Report of Overage Standards due for 5-Year Review

Enclosed is a report of all standards that are due for a 5-year review. As you are aware, ANSI procedures require that every standard be reviewed every five years to determine whether that standard should be REVISED, REAFFIRMED, or WITHDRAWN. It is the responsibility of the Technical Committees to make this determination for the standards under their jurisdiction.

The documents listed in the enclosed report are due for 5-year review.

TC OFFICERS: Please review the attached report and start the review of any standards under your committee's jurisdiction.

SPARC Liaisons: Please remind your Technical Committees to start reviewing these standards.

*Henry J. Phillips*  
 Henry J. Phillips  
 Recording Secretary, X3

Encl: Report on Overage Standards

\*Operating under the procedures of The American National Standards Institute.  
 X3 Secretariat: Computer and Business Equipment Manufacturers Association  
 311 First Street, N.W., Suite 800, Washington, DC 20001-2178

Tel: 202/737-6868  
 Fax: 202/638-4922

January 7, 1988 4:11 PM  
 X3 28 PAGE 1

X3 Secretariat/CBEMA

Over Age Standards Report

Standard Designation 1 Year Approved	Year	Proj. No	Title	X3 Technical Committee
X3.18	1982	0076 N	One-Inch Perforated Paper Tape for Information Interchange	SPARC
X3.19	1982	0077 N	Eleven-Sixteenths Inch Perforated Paper Tape for Information Interchange	SPARC
X3.20	1982	0078 N	Take-up Reels for One Inch Perforated Tape for Information Interchange	SPARC
X3.5	1970	0081 M	Film Chert Substrate and Tissue Used in Information Processing	SPARC
X3.17	1981	0057 N	Character Set for Optical Character Recognition (OCR-A)	X3A1
X3.2	1974	0017 R	Print Specifications for Magnetic Ink Character Recognition Incorporates X3 Proj. 314	X3A1
X3.45	1982	0042 N	Character Set for Handprinting	X3A1
X3.49	1982	0061 N	Character Set for Optical Character Recognition (OCR-B)	X3A1
X3.93M	1981	0069 N	Optical Character Recognition (OCR) Character Positioning	X3A1
X3/TR-5	1982	0274 N	Design of OCR Forms	X3A1
X3/TR-XI		0312 BT	Basic Information on OCR	X3A1
X4.16A	1977	0432 N	Addendum to X4.16-1976: Encodings for Track 33 Liaison with X9	X3B10
X4.21	1981	0434 N	Interindustry Message Specifications for Credit Cards	X3B10
X3.55	1982	0221 R	Unrecorded Magnetic Tape Cartridge for Information Interchange (0.250 Inch, 1400 RPM; Phase Encoded)	X3B5

95

56

167(X) JCH-11

Standard Designation & Year Approved	Proj. No	Title	X3 Technical Committee
X3.63	1981	0225 M Interchangeable Magnetic Twelve-Disk Pack (100 Megabytes)	X3B7
X3.04	1981	0251 M Interchangeable Magnetic Twelve-Disk Pack (200 Megabytes)	X3B7
ISO 7665		0272 D Flexible Disk Labels and File Structures	X3B8
X3.82	1980	0306 R One-Sided Single-Semitsu Unformatted 5.25 Inch Flexible Disk Cartridge	X3B8
X3.11	1969	0101 R Specification for General Purpose Paper Cards for Information Interchange	X3B9
X3.21	1980	0102 M Rectangular Holes in Twelve-Hole Punched Cards	X3B9
X3/TR-2	1982	0440 M Conversion of Paper Substance Weights from Gram Weights to 4/82	X3B9
X3.60	1978	0215 RF Preprocessing Language Minimal BASIC	X3J2
X3/TR-3	1982	0316 MI APT Language - Expository Remarks Concerning X3.37-1980	X3J7
X3/TR-4	1982	0315 MI APT Language - Postprocessor Interface Modules	X3J7
X3/TR-6	1982	0016 R Guide for Technical Documentation of Computer Projects	X3K1
X3/TR-1	1982	0026 D American National Dictionary for Information Processing Systems (ANRIPS)	X3K5

56

Standard Designation & Year Approved	Proj. No	Title	X3 Technical Committee
X3.32	1973	0006 M Graphic Representation of the Control Characters of the American National Code for Information Interchange	X3L2
X3.41	1974	0105 R Code Extension Techniques for Use with the 7-Bit Coded Character Set for the ANS Code for Information Interchange	X3L2
X3.6	1973	0107 M Perforated Tape Code for Information Interchange	X3L2
X3.95	1982	0304 M Hexadecimal Input/Output to Microprocessors Using 5-Bit and 7 Bit Teleprinters	X3L2
X3.44	1979	0004 R Control Codes for 8-Bit Sets	X3L2.2
Z39.27	1976	0096 L Identification of Countries (ANSI Z39/SC27)	X3L8
X3.24	1968	0113 L Signal Quality at Interface Between DTE and Synchronous DCE for Serial Data Transmission (Liaison with EIA TR-30)	X3S3
X3.66	1979	0049 R Advanced Data Communication Control Procedures (ARCCP)	X3S3.4
X3.42	1975	0104 RF Representation of Numeric Values in Character Strings for Information Interchange	X3T2

[ 35 Records ]

107 (\*) JCA-12 (57)

X3/88-088-X,S

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

**NEWS RELEASE**

January 15, 1988

For more information contact:  
Jim Brodie, X3J11 Chair  
at 602-961-0032  
or  
Tom Plum, X3J11 Vice Chair  
at 609-927-3770

**X3 ANNOUNCES SECOND PUBLIC REVIEW AND COMMENT PERIOD ON  
DRAFT AMERICAN NATIONAL STANDARD X3.159-198X,  
PROGRAMMING LANGUAGE C**

Washington, D. C. -- X3, the Accredited Standards Committee on Information Processing Systems, announces a **SECOND PUBLIC REVIEW** and two-month comment period on draft proposed American National Standard, X3.159-198X. The public review period extends from February 12, 1988 to April 12, 1988.

X3J11, the technical committee charged with developing a standard for the C programming language, has released a new draft proposed standard for formal public comment. This new draft contains many changes in response to the comments received during the first formal public review period.

The committee received and reviewed about 500 individual requests for changes, contained in 32 letters. The comments received were helpful in Standard and Rationale, pointing out the need to clarify or reword certain sections, and leading to Committee discussions resulting in either technical improvements to the draft Standard and Rationale or reaffirmation of previous decisions.

Major wording changes have been made in an attempt to clarify and make more precise the meaning of several sections. The areas of types and type compatibility received particular attention.

The committee also received several requests for additional language support and changes from the International community. Since one of the main goals of the committee has been to develop a standard which would be acceptable as an International Standards Organization (ISO) standard as well as an American National Standards Institute (ANSI) standard, these requests were also carefully

reviewed and considered. A significant change which resulted from this work has been the addition of low level support for multi-byte characters such as those needed to represent Japanese and Chinese characters sets.

This draft standard is available for public review and comment for a two-month period ending April 12, 1988. Copies may be obtained from GLOBAL ENGINEERING DOCUMENTS, INC. by calling 800-854-7179.

Single Copy Price: \$65.00  
International Orders: \$84.50

‡ ‡ ‡ ‡ ‡

Monday, February 8, 1988

- 8:15 Opening Business (J. Adams)  
Public Review Comment Document (S12, I. Philips)<sup>\*</sup>, 107-IRP-2, 107-IRP-3)  
Public Review Processing Procedures (106-107, 106-108, 107-CDB-1)  
List of Approved Changes (S16, A. Johnson)<sup>\*</sup>, separate distribution)  
~~Responses to WGS Liverpool Resolutions (S14, J. Wagoner) (106-61)~~  
~~Clooney (106-50)~~  
Fortran 77 Interpretations (106-104, 107-EAJ-1)<sup>\*</sup>, 107-EAJ-4)
- 1:15 Subgroup Meetings  
4:30 Subgroup Heads Coordination Meeting

Tuesday, February 9, 1988

- 8:15 Subgroup Reports (may include public review comment processing action)  
Misc. Editorial Action (106-62, 106-63, 106-11, 107-LWC-1)<sup>\*</sup>, 107-LWC-3, 106-50)  
Syntax Rule Number Changes (107-LWC-2)  
Fortran 77 Audit of S8 (106-55, 106-87, 106-92, 106-113, ~~107-BTS-1~~,  
107-LJM-1, 107-JLS-1, 107-KWH-1, 107-TMRE-1,  
107-TMRE-2, 107-TMRE-3, ~~107-TMRE-4~~)  
~~Control Constructs Edits (106-85, 106-86, 107-JHM-2)~~  
~~Section 12 Notes (107-KWH-2)~~  
Source Form (106-29, 107-RAH-1, 107-EAJ-3)<sup>\*</sup>, 107-RAH-2)
- 1:15 Subgroup Meetings

Wednesday, February 10, 1988

- 8:15 Subgroup Reports  
Editorial Committee (L. Campbell)<sup>\*</sup>, 107-PLS-3, 107-KWH-5, 107-KWH-7)  
WGS-L11, Deprecated Features (107-EAJ-2)  
WGS-L17, Processor Limits (106-31)  
WGS-L21, Square Brackets (107-GP-2)<sup>\*</sup>, 107-KWH-8, 107-RCS-1)  
IDENTIFY (107-GP-1)<sup>\*</sup>, 107-GP-2)  
DATA Statement (107-IRP-1)
- 1:15 Subgroup Meetings

Thursday, February 11, 1988

- 8:15 Subgroup Reports  
Editorial Committee (L. Campbell)  
~~IOLNGTH Scope (106-115, 107-JHM-1)~~  
IOLNGTH, RECL (106-62, ~~107-PS-2~~)<sup>\*</sup>, 107-CDB-3)  
SCRATCH Files (107-PS-1)<sup>\*</sup>, 107-PLS-2)  
Internal Files (106-65, 107-CDB-2)
- 1:15 Subgroup Meetings  
4:30 Subgroup Heads Coordination Meeting

Friday, February 12, 1988

- 8:15 Subgroup Reports  
Function Results (~~107-AW-1~~)<sup>\*</sup> (107-62)  
MOLD Alternatives (106-36, ~~107-AW-2~~)  
Intrinsic Function Names (106-45, 107-BLT-1)  
REPEAT Function (107-KWH-3)  
Random Numbers (106-38)  
Deallocating Function Results (107-KWH-4)
- 1:15 Editorial Committee (L. Campbell)  
Closing Business (J. Adams)

(Document numbers of the form:  
106-*nnn* represent papers submitted to meeting #106 (November, 1987);  
107-*xyz-n* represent papers expected to be submitted to meeting #107.)

January 25, 1988

TO: X303 next meeting

FROM: Chen Ming-Yuan  
 Beijing Wire Communications Plant  
 P. O. Box 97  
 Beijing  
 People's Republic of China

SUBJECT: A Proposal on Hanzi Processing in FORTRAN 8X

From 18 to 20 in January, 1988 FORTRAN 8X Working Group of China and Chinese Internal Code and Data Type Working Group held a meeting in Beijing on which we discussed how to extend the functions of processing single-octet and multi-octet data in FORTRAN 8X.

We have some suggestions; they are described as follows.

1) Several new definitions and concepts:

a) Essential character set

Essential character set is the character set which is used by the processor by which the FORTRAN compiler is supported.

b) Character data type

Various kinds of character data types correspond to different kinds of character sets respectively. Therefore the kind of character set will be a character data type in FORTRAN. Every character data type has two basic attributes. One is the length of character-storage-unit and another is the control-sequence. Control-sequence is used to distinguish various kinds of characters in source programs or I/O data streams, except for characters in essential character set. Every character occupies a character-storage-unit, it does not matter to which kind of character data type. But the length of different kind of character-storage-units may be same or different.

c) The width of character entity

The width of a character entity is the width it occupies when the content of character entity occurs on the external medium.

2) The declaration of character data type

We suggest two forms for the declaration of character data type.

One of the forms is:

```
CHARACTER [(KIND=n)] [*len[,]] nam[,nam]...
```

where n is an integer constant expression. Its value is greater than or equal to one. It is the number of character data type. When n=1, it represents the character data type of

essential character set. From 2 on, it represents some kind of character data type respectively. The relationship between the number of a character data type and the character set corresponding the character data type is established by auxiliary module of FORTRAN compiler.

Another form for the declaration of character data type is:

```
[lang-name] CHARACTER [*nam[,]] nam[,nam]...
```

where lang-name is the name of character data type. It may be a set of limited key words which represent different character data type. For example, HANZI indicates Chinese character, KANJI indicates Japanese character and so on. The langname is also established by an auxiliary module in FORTRAN compiler.

3) The operations of character data type

Operations, e.g. concatenation and relational operations, on data of multi character data type are allowed when both operands have the same character data type or when the essential character set is a subset of another character set. The data of essential character set can be translated into the data of another character data type. Then they can be operated together in expressions or assigned in character assignment statements.

4) Some extended internal functions

Character functions remain unchanged for essential character set. A group of new character functions are used to deal with the data of other character types. They may be as follow:

function name	type of variable	type of function
NICHAR(a)	any character type	integer
NCHAR(i[,n])	integer	any character type
* the meaning of n is the same with the n in character declaration statement.		
NLEN(a)	any character type	integer
NWIDTH(a)	any character type	integer
* the meaning for the value of the function NWIDTH(a) is the same with the concept of the width of character entity.		

陳明遠 chen Ming-Yuan

Chen Ming-Yuan  
CAS/FORTRAN 8X WG  
Chinese Internal Code and Data Type



(60)

Dear Loren,

I have no objection at all to your publishing our 106 pointer paper in Fortran Forum, and I expect any changes you have made will be improvements. As to some of your other comments I think what you want to do is quite legitimate and it was our intention to allow just such code.

I thought we had been fairly explicit in the model. Our interpretation is that when the ALLOCATABLE, ALIAS attributes, or both, are specified for a given name a descriptor is created. The name refers to the storage to hold the descriptor which at this stage is undefined in the Fortran sense, it is not yet associated with space to hold an object of the type and rank declared. When the name appears in an ALLOCATE statement or as the pointer in an IDENTIFY statement the values in the descriptor become defined, the descriptor depending on implementation will contain an address in memory for the actual space capable of containing an object value of the appropriate type and the values of any additional parameters (like dope vector values) necessary to locate the pieces of a compound object.

As I see it the critical question of your example is can you have a statement like,

```
IDENTIFY(current = current%ptocell)
```

when CELL is a recursive data structure and current is a pointer to such a structure. That is, CELL is a type defined essentially like,

```
TYPE CELL
  INTEGER::val
  type(CELL),ALLOCATABLE,ALIAS::ptocell
ENDTYPE CELL
```

and current declared as

```
type(CELL),ALLOCATABLE,ALIAS::current
```

The model I thought was pretty unambiguous. Ordinary assignment copies values with dereferencing but IDENTIFY copies descriptors. A statement such as

```
current = current%ptocell
```

would cause the pointer current%ptocell to be dereferenced to produce the value of the integer and the descriptor which is the pto cell component in the target CELL. This value component and descriptor component value are copied into storage to hold a CELL object at present associated with the pointer to a CELL named current. Basically current is made to refer to the next CELL in a list. The statement

```
IDENTIFY(current = current%ptocell)
```

causes the value of the descriptor in current%ptocell to be copied to the descriptor which is referred to by the name current. In the first case the space associated with current does not change but the values stored in that space does. In the second the pointer current is made to associate with different space. In one case the list is moved up through the current

60

window. In the other the current window moves down through the list.

Your question about the creation of pointers on the heap I think is also covered. If the pointer current is allocated, space is created on the heap to hold a CELL. That is, space is allocated to hold an integer and the descriptor for a CELL. This heap resident pointer is not however "anonymous" since it is referenced by the name current%ptocell. If the pointer current%ptocell is itself allocated a further heap pointer would be created which would be referenced by the name current%ptocell%ptocell, and so on.

I think this covers the main thrust of your comments. I totally agree with you by the way that spelling POINTER as ALLOCATABLE, ALIAS is inelegant in the extreme and highly misleading since the two attributes taken together are subtly different from either separately. An allocatable only object is always dereferenced in all contexts except as an actual argument matching a dummy argument that is allocatable, at least by our current rules. The current allocatable rules are I am afraid confused. Most of them are simply those required of a dynamic sized array. The objects could be implemented with little more than a slightly extended stack. However the rules relating to allocatable function results and arguments are really those of a pointer and require heap management. Since writing the paper for meeting 106 I have become convinced that we should separate allocatable arrays from pointers. We should not permit allocatable arrays as dummy arguments or as function results, where it is clearly a pointer that is being passed, but restrict them to local contexts where a dynamic sized array is needed. (I think we politically need to do this even if I personally would prefer to remove them altogether and provide the functionality by proper pointers.) Provided an ALIAS is allowed to be allocated as well as identified, the basic properties of pointers could then be defined as in the model I outline in the paper without the awful syntax and confusing semantics of the ALLOCATABLE, ALIAS combination. I don't really care what the attribute is called as long as it is not in addition to ALIAS but is instead of and is exclusive with ALLOCATABLE.

I don't understand your point about two definable objects that are associated by being addressed by the same pointer. I do not see how this could occur. At any one time a pointer is associated with, has as its current target, one and only one object. Of course this may be a compound object but I don't see any problem there. A pointer P may at one time be targeted on object A and at another on object B. In Fortran terms this does not constitute an association between A and B, even if it does in the normal English use of the word.

I have considerable sympathy for the Algol68 distinction which clearly separates the object which is a value, one that is the name of a value, one that is the name of a name of a value and so on. However I don't think I could sell this to X3J3, nor do I think it essential just helpful in providing clarity of description. In the technical sense that Fortran uses the word associate I think it is the correct term. A constant has a fixed association with space and value. A variable has a fixed association with space but a possibly changing association with a value. A pointer has a possibly changing association with space as well as the space with value.

I hope this is of some clarification. Trying to make sure the model did what you want I found useful. Thanks for your interest and help. I think it will be a tough fight to get even these pointers into F8x but I intend to try given even modest support in the comments.

60

Yours sincerely

Dr.J.L.Schonfelder

## CERN USERS' VOTES ON FORTRAN 8X

1. Should there be a bit data type? 45/1/5
  - a. If yes, is the proposal in Appendix F. adequate? 36/2/13
  - b. Should there be a set of BIT intrinsics if no BIT data type is introduced? 43/0/8
2. Should there be a pointer facility? 18/22/11
  - a. If yes, should it be explicit? 20/1/30
  - b. If yes, should it be strongly typed? 18/5/28
  - c. And should pointer arithmetic be allowed? 19/5/29
3. Should blanks be significant in the new source form? 29/11/11
4. Should there be I/O facilities for binary/octal/hex numbers? 47/3/1
5. Should there be provision for multi-byte characters? 15/22/14
6. Should there be a facility for varying length strings? 35/9/7
7. Should there be a facility for vector-valued subscripts? 30/10/11
8. Is the proposed language 'too big'? 10/21/20
9. Is the obsolescent and deprecated mechanism a good one (on the whole)? 37/1/13
10. Are you prepared to give up storage association in 20-30 years' time? 15/19/17
11. Should multiple and trailing underscores be forbidden? 21/18/12
12. Should multiple statements on a single source line be forbidden? 15/27/9
13. Should the RANGE facility be removed? 36/3/12
14. Should recursion be removed? 6/37/8
15. Should the NAMELIST facility be removed? 22/9/20
16. Should keyword and optional arguments be removed? 5/29/17
17. Should the proposal for generalised precision be replaced by another having the effect of retaining double-precision and adding double-precision complex? 10/19/22
18. Would you prefer to see the current proposal implemented with little further delay, rather than wait 2-3 years for new features to be introduced? 21/21/9

62

62a

TO: X3J3  
FROM: Alan Wilson  
SUBJECT: Revisions to S8.104(JUNE 1987)  
DATE: 23 Dec. '87

INTRODUCTION

There appears to be a number of problems with the rules governing the specification of array-valued functions, especially when they are allocatable.

[1] There is a problem with stating that the function characteristics are specified by the interface block, since the latter only specifies the rank but not the extent(s) of an allocated function result. See 12-2 (lines 6-9), 12-3 (lines 37-38).

[2] The model of allocation/deallocation as it applies to an allocatable function result is apparently meant to be that the result is always allocated inside the function body and implicitly deallocated after the result is used in the reference that invoked the function: this model should be made more explicit. See 6-4 (lines 7-8).

[3] The clear statement of just how the function gets its value should be brought forward from F'77. (page 15-7 of ANSI X3.9-1978 ).

[4] It should be made clear that the result variable automatically acquires the characteristics of the function\_name, including its allocatability. (12-9 (of S.8) line 36...).

[5] It is clearly stated (page 12-9, lines 36 to 41) that when there is a result variable it follows that "all occurrences" of the original function name are recursive function references. Now, one kind of "occurrence" is to assign to (or otherwise define) the original function name, which leaves open the possibility that such an occurrence constitutes a recursive function reference. But so far as I am aware, that's nonsense. So shouldn't S.8 be clear on this point, which I take to be that ONLY the result variable, when there is one, can be defined in the function body.

P. 434

[6] There is at least one place where the assumption appears to be made that something other than an array is allocatable. ( 12-2 line 33 ) That assumption is incorrect.

.....  
EDITORIAL CHANGES

.....  
I recommend the following changes to S.8, which I believe are merely editorial.

<u>LOCATION</u>	<u>RECOMMENDATION</u>
[1] page 12-2, line 7	CHANGE: 'shape' TO: 'rank'
[2] page 12-2, line 7	AFTER: ',and' INSERT: 'if it is an array,'
[3] page 12-2, line <sup>7</sup>	ADD THE <del>FINAL</del> SENTENCE: "If a function result is an array <del>but</del> not allocatable, its shape is a <sup>that is</sup> characteristic."
[4] page 12-2, line 33	DELETE: the whole line
[5] page 12-2, lines 34,35,37	CHANGE: (d)(e)(f) to (c)(d)(e)
[6] page 12-7, lines 39 to 41	REPLACE the sentence that begins on line 39 with the sentence: 'The characteristics of the function result (12.2.2) are <del>specified by</del> <sup>determined</sup> the interface of the function.'
<del>[7]</del> page 12-9, line 32.5	INSERT (a new line): "If the function result is allocatable, it must be allocated within the function body".
<del>[8]</del> page 12-9, line 35	INSERT after the period: "Unless RESULT is specified, function-name must become defined during every execution of the function, and once defined may be referenced or become redefined. "

~~62~~  
62a

~~[9]~~ page 12-7, line 38 | INSERT after the period the  
additional sentence: "Result-name  
has the same characteristics as  
function-name and must become  
defined during every execution of  
the function; once defined it may  
be referenced or become redefined.  
Function-name must not be defined  
when RESULT is specified."

~~[10]~~ page 12-7, line  
41.5 | INSERT (new line): "The value of  
the result variable when a RETURN  
or END statement is executed in  
the subprogram is the value of the  
function."

.....  
END OF RECOMMENDATIONS  
.....

p. 436

(63)

DATE: February 2, 1988  
TO (M/S): X3J3 Subgroup 22  
FROM (M/S): Andrew D. Tait  
EXT NO:  
SUBJECT: Response to P. Sinclair's Comments 20 and 21  
REFERENCE: 106(\*)JLW-2

The attached revision to S8.104 paragraph 9.6 is proposed as the response to comments 20 and 21 from Paul Sinclair (see 106(\*)JLW-2 for details). In summary the proposed changes are:

Page 9-18:

- o change line 40
- o insert new line after line 40
- o add new rules (R924 and R925) after line 41

Page 9-19:

- o delete line 4
- o delete line 5
- o change line 6 to begin R926
- o add constraints after line 25
- o delete line 26
- o delete line 27

Page 9-21:

- o line 39, change 9.6.1.20 to 9.6.2
- o line 45, change 9.6.1.21 to 9.6.3
- o delete line 48
- o delete line 49

Page 9-22:

- o delete line 1
- o change line 2

I believe that these are simply editorial changes and it is not the intention of the revision to make substantive changes to S8.104.

Andrew D. Tait

at0202c:kc

Att.

P. 437



63

16 ereo to have been written. Thus, only those records may be read during subsequent direct  
17 access connections to the file.

18 After execution of an ENDFILE statement, a BACKSPACE or REWIND statement must be  
19 used to reposition the file prior to execution of any data transfer input/output statement.

20 Execution of an ENDFILE statement for a file that is connected but does not exist creates the  
21 file.

22 An example of an ENDFILE statement is:

23 ENDFILE K

24 9.5.3 REWIND Statement. Execution of a REWIND statement causes the specified file to  
25 be positioned at its initial point. Note that if the file is already positioned at its initial point,  
26 execution of this statement has no effect on the position of the file.

27 Execution of a REWIND statement for a file that is connected but does not exist is permitted  
28 but has no effect.

29 An example of a REWIND statement is:

30 REWIND 10

31 9.6 File Inquiry. The INQUIRE statement may be used to inquire about properties of a  
32 particular named file or of the connection to a particular unit. There are three forms of the  
33 INQUIRE statement: inquire by file, which uses the FILE= specifier, inquire by unit, which  
34 uses the UNIT= specifier, and inquire by output list, which uses only the IOLENGTH=  
35 specifier. All specifier value assignments are performed according to the rules for assignment  
36 statements.

37 An INQUIRE statement may be executed before, while, or after a file is connected to a unit.  
38 All values assigned by an INQUIRE statement are those that are current at the time the state-  
39 ment is executed.

40 R923 *inquire-stmt* is INQUIRE (*inquire-by-file-spec-list*)  
or INQUIRE (*inquire-by-unit-spec-list*)  
41 or INQUIRE (IOLENGTH = *scalar-int-variable*) *output-item-list*

R924 *inquire-by-file-spec-list* is [*inquire-spec-list*]

■ FILE = *file-name-expr* ■

■ [, *inquire-spec-list*]

R925 *inquire-by-unit-spec-list* is [*inquire-spec-list*]

■ [UNIT = ] *external-file-unit* ■

■ [, *inquire-spec-list*]

42 Examples of INQUIRE statements are:

43 INQUIRE (IOLENGTH = IOL) A (1:N)

44 INQUIRE (UNIT = JOAN, OPENED = LOG\_01, NAMED = LOG\_02, &

2 9.6.1. Inquiry Specifiers. Unless constrained, the following inquiry specifiers may be used  
3 in either of the inquire by file or inquire by unit forms of the INQUIRE statement:

4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27

R926 inquire-spec.

- is IOSTAT = scalar-int-variable
- or ERR = label
- or EXIST = scalar-logical-variable
- or OPENED = scalar-logical-variable
- or NUMBER = scalar-int-variable
- or NAMED = scalar-logical-variable
- or NAME = scalar-char-variable
- or ACCESS = scalar-char-variable
- or SEQUENTIAL = scalar-char-variable
- or DIRECT = scalar-char-variable
- or FORM = scalar-char-variable
- or FORMATTED = scalar-char-variable
- or UNFORMATTED = scalar-char-variable
- or RECL = scalar-int-variable
- or NEXTREC = scalar-int-variable
- or BLANK = scalar-char-variable
- or POSITION = scalar-char-variable
- or ACTION = scalar-char-variable
- or DELIM = scalar-char-variable
- or PAD = scalar-char-variable

63

Constraint: An inquire-by-file-spec-list must contain at least one inquiry specifier.

Constraint: An inquire-by-unit-spec-list must contain at least one inquiry specifier.

Constraint: An inquiry specifier may appear only once in an inquire-spec-list

28 Constraint: In the inquire by unit form of the INQUIRE statement, if the optional characters  
29 UNIT= are omitted from the unit specifier, the unit specifier must be the first  
30 item in the inquire-

31 When a returned value of a specifier other than the NAME= specifier is of type character and  
32 the processor is capable of representing letters in both upper and lower case, the value  
33 returned is in upper case.

34 If an error condition occurs during execution of an INQUIRE statement, all of the inquiry  
35 specifier variables become undefined, except for the variable in the IOSTAT= specifier (if  
36 any).

37 9.6.1.1. FILE= Specifier in the INQUIRE Statement. The value of file-name-expr in the  
38 FILE= specifier specifies the name of the file being inquired about. The named file need not  
39 exist or be connected to a unit. The value of file-name-expr must be of a form acceptable to  
40 the processor as a file name. If a processor is capable of representing letters in both upper  
41 and lower case, the interpretation of case is processor dependent.

42 9.6.1.2. EXIST= Specifier in the INQUIRE Statement. Execution of an INQUIRE by file  
43 statement causes the scalar-logical-variable in the EXIST= specifier to be assigned the value  
44 true if there exists a file with the specified name; otherwise, false is assigned. Execution of  
45 an INQUIRE by unit statement causes true to be assigned if the specified unit exists; other-  
46 wise, false is assigned.

P. 439

- 1 9.6.1.3. OPENED = Specifier in the INQUIRE Statement. Execution of an INQUIRE by file  
2 statement causes the *scalar-logical-variable* in the OPENED = specifier to be assigned the  
3 value true if the file specified is connected to a unit; otherwise, false is assigned. Execution  
4 of an INQUIRE by unit statement causes *scalar-logical-variable* to be assigned the value true if  
5 the specified unit is connected to a file; otherwise, false is assigned.
- 6 9.6.1.4. NUMBER = Specifier in the INQUIRE Statement. The *scalar-int-variable* in the  
7 NUMBER = specifier is assigned the value of the external unit identifier of the unit that is cur-  
8 rently connected to the file. If there is no unit connected to the file, the value -1  
9 assigned.
- 10 9.6.1.5. NAMED = Specifier in the INQUIRE Statement. The *scalar-logical-variable* in the  
11 NAMED = specifier is assigned the value true if the file has a name; otherwise, it is assigned  
12 the value false.
- 13 9.6.1.6. NAME = Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
14 NAME = specifier is assigned the value of the name of the file if the file has a name; other-  
15 wise, it becomes undefined. Note that if this specifier appears in an INQUIRE by file state-  
16 ment, its value is not necessarily the same as the name given in the FILE = specifier. For  
17 example, the processor may return a file name qualified by a user identification. However,  
18 the value returned must be suitable for use as the value of *file-name-expr* in the FILE =  
19 specifier in an OPEN statement. If a processor is capable of representing letters in both  
20 upper and lower case, the interpretation of case is processor dependent.
- 21 9.6.1.7. ACCESS = Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
22 ACCESS = specifier is assigned the value SEQUENTIAL if the file is connected for sequential  
23 access, and DIRECT if the file is connected for direct access. If there is no connection, it is  
24 assigned the value UNDEFINED.
- 25 9.6.1.8. SEQUENTIAL = Specifier in the INQUIRE Statement. The *scalar-char-variable* in  
26 the SEQUENTIAL = specifier is assigned the value YES if SEQUENTIAL is included in the set  
27 of allowed access methods for the file, NO if SEQUENTIAL is not included in the set of  
28 allowed access methods for the file, and UNKNOWN if the processor is unable to determine  
29 whether or not SEQUENTIAL is included in the set of allowed access methods for the file.
- 30 9.6.1.9. DIRECT = Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
31 DIRECT = specifier is assigned the value YES if DIRECT is included in the set of allowed  
32 access methods for the file, NO if DIRECT is not included in the set of allowed access meth-  
33 ods for the file, and UNKNOWN if the processor is unable to determine whether or not  
34 DIRECT is included in the set of allowed access methods for the file.
- 35 9.6.1.10. FORM = Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
36 FORM = specifier is assigned the value FORMATTED if the file is connected for formatted  
37 input/output, and is assigned the value UNFORMATTED if the file is connected for unformat-  
38 ted input/output. If there is no connection, it is assigned the value UNDEFINED.
- 39 9.6.1.11. FORMATTED = Specifier in the INQUIRE Statement. The *scalar-char-variable* in  
40 the FORMATTED = specifier is assigned the value YES if FORMATTED is included in the set  
41 of allowed forms for the file, NO if FORMATTED is not included in the set of allowed forms for  
42 the file, and UNKNOWN if the processor is unable to determine whether or not FORMATTED  
43 is included in the set of allowed forms for the file.
- 44 9.6.1.12. UNFORMATTED = Specifier in the INQUIRE Statement. The *scalar-char-variable*  
45 in the UNFORMATTED = specifier is assigned the value YES if UNFORMATTED is included  
46 in the set of allowed forms for the file, NO if UNFORMATTED is not included in the set of  
47 allowed forms for the file, and UNKNOWN if the processor is unable to determine whether or  
48 not UNFORMATTED is included in the set of allowed forms for the file.

1 9.6.1.13 RECL= Specifier in the INQUIRE Statement. The *scalar-int-variable* in the  
2 RECL= specifier is assigned the value of the maximal record length of the file. If the file is  
3 connected for formatted input/output, the length is the number of characters. If the file is  
4 connected for unformatted input/output, the length is measured in processor-defined units. If  
5 the file does not exist, *scalar-int-variable* becomes undefined. (63)

6 9.6.1.14 NEXTREC= Specifier in the INQUIRE Statement. The *scalar-int-variable* in the  
7 NEXTREC= specifier is assigned the value  $n + 1$ , where  $n$  is the record number of the last  
8 record read or written on the file connected for direct access. If the file is connected but no  
9 records have been read or written since the connection, *scalar-int-variable* is assigned the  
10 value 1. If the file is not connected for direct access or if the position of the file is indetermi-  
11 nate because of a previous error condition, *scalar-int-variable* becomes undefined.

12 9.6.1.15 BLANK= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
13 BLANK= specifier is assigned the value NULL if null blank control is in effect for the file con-  
14 nected for formatted input/output, and is assigned the value ZERO if zero blank control is in  
15 effect for the file connected for formatted input/output. If there is no connection, or if the  
16 connection is not for formatted input/output, *scalar-char-variable* is assigned the value UNDE-  
17 FINED.

18 9.6.1.16 POSITION= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
19 POSITION= specifier is assigned the value REWIND if the file is connected by an OPEN  
20 statement for positioning at its initial point, APPEND if the file is connected for positioning at  
21 its terminal point, and ASIS if the file is connected without changing its position. If there is no  
22 connection, *scalar-char-variable* is assigned the value UNDEFINED. If the file has been repo-  
23 sitioned since the connection, *scalar-char-variable* is assigned the value UNDEFINED.

24 9.6.1.17 ACTION= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
25 ACTION= specifier is assigned the value READ if the file is connected for input only, WRITE  
26 if the file is connected for output only, and READ/WRITE if it is connected for both input and  
27 output. If there is no connection, *scalar-char-variable* is assigned the value UNDEFINED.

28 9.6.1.18 DELIM= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
29 DELIM= specifier is assigned the value APOSTROPHE if the apostrophe is to be used to  
30 delimit character data written by list-directed or namelist formatting. If the quotation mark is  
31 used to delimit such data, the value QUOTE is assigned. If neither the apostrophe nor the  
32 quote is used to delimit the character data, the value NONE is assigned. If there is no con-  
33 nection or if the connection is not for formatted input/output, *scalar-char-variable* is assigned  
34 the value UNDEFINED.

35 9.6.1.19 PAD= Specifier in the INQUIRE Statement. The *scalar-char-variable* in the  
36 PAD= specifier is assigned the value NO if the connection of the file to the unit included the  
37 PAD= specifier and its value was NO. Otherwise, *scalar-char-variable* is assigned the value  
38 YES.

39 9.6. 2 IOLENGTH= Specifier in the INQUIRE Statement. The *scalar-int-variable* in the  
40 IOLENGTH= specifier is assigned the processor-dependent value that would result from the  
41 use of the output list in an unformatted output statement. Any DO variables have the scope  
42 of the implied-DO list, as in the DATA statement. The value must be suitable as a RECL=  
43 specifier in an OPEN statement that connects a file for unformatted direct access when there  
44 are input/output statements with the same input/output list.

45 9.6. 3 Restrictions on Inquiry Specifiers. A variable that may become defined or  
46 undefined as a result of its use in a specifier in an INQUIRE statement, or any associated  
47 entity, must not appear in another specifier in the same INQUIRE statement.

48  
49

1 The unit specified in an *inquire-by-unit-spec-list*  
2 need not exist or be  
3 connected to a file. If it is connected to a file, the inquiry is being made about the connection  
4 and about the file connected.

63

5 **9.7. Restrictions on Function References and List Items.** A function reference must  
6 not appear in an expression anywhere in an input/output statement if such a reference  
7 causes another input/output statement to be executed. Note that restrictions in the evalua-  
8 tion of expressions (7.1.7) prohibit certain side effects.

9 **9.8. Restriction on Input/Output Statements.** If a unit, or a file connected to a unit,  
10 does not have all of the properties required for the execution of certain input/output state-  
11 ments, those statements must not refer to the unit.

P. 442

Relay-Version: version Notes 2.7.5 (840 Contrib) 87/2/5; site hpclcdb.HP.COM  
From: kent@xanth.cs.odu.edu (Kent Paul Dolan)  
Date: Thu, 4 Feb 88 03:30:18 GMT  
Date-Received: Fri, 5 Feb 88 03:17:50 GMT  
Subject: FORTRAN 8X discussion  
Message-ID: <3885@xanth.cs.odu.edu>  
Organization: Old Dominion University, Norfolk Va.  
Path: hpclcdb!hpcllla!hpesocl!hpda!hplabs!sri-unix!husc6!cmcl2!brl-adm!umd5!ames!xa  
Newsgroups: comp.lang.fortran,comp.std.internat  
Reply-To: kent@xanth.UUCP (Kent Paul Dolan)  
Keywords: program maintenance costs, compiler performance worries  
Lines: 629  
Summary: mostly pro changes  
Xref: hpda comp.lang.fortran:411 comp.std.internat:250

64

*Suggested  
Response for  
a Public Review  
letter - from  
the Public!*

Presley,

Thanks for the long note. I have included it verbatim (except for spelling corrections and reformatting) here with my comments. Since the issues discussed are of interest to the larger community, I have taken the liberty of posting the discussion, long as it is.

I think the major difference we have is that I am less interested in the performance of compilers (I use, rather than sell them), than I am in the performance of programmers (being one). I see the committee's proposal, flawed though it undoubtedly is, as a major, correct step toward decreasing the life cycle cost of writing and maintaining FORTRAN code. In that larger perspective, taking a hit on compiler performance is pretty small potatoes.

Kent, the man from xanth.

>From psmith@convex.uucp Wed Feb 3 15:42:07 1988  
>To: kent@xanth.cs.odu.edu  
>Subject: Re: FYI - Thanks for the Input

[In response to a previous note of mine - Kent.]

>In reality, I tend to agree with you. In fact, IBM, DEC, and others  
>do also. They have said that the committee has produced a new language.

No, the committee has done major surgery on a very ill patient. Not the same thing at all.

>I have no problem with letting FORTRAN go the way of ALGOL and others.

I have no trouble with someone taking a gun to FORTRAN and saying "Bang, you're dead"; I just don't think it will happen. Letting the language linger on, looking like a viable language, continuing to be taught by FORTRAN using professors to unsuspecting engineering majors, costing probably billions in excessive maintenance costs due to problems which can be remedied only by such radical surgery as proposed by the committee, to this I do object. The problem is not that FORTRAN is dying, the problem is that FORTRAN is not dying, but it is bleeding us white.

>When you try to piece together old portions with new portions many times  
>what you get is a mess.

No quarrel, the committee has taken on an ambitious job, and has to do it well to have a successful result. I can just imagine how much fun it must be to be trying to make major

changes to the language with 1/3 of the committee screaming bloody murder and trying to impede the changes at every step.

64

Actually, I don't have to imagine it; I remember it from my X3H3 work, when every vendor's favorite construct had to be added to the language. If the Germans had not gone off in a corner and written GKS by themselves, and threatened to take it to ISO as a DIN proposal if we didn't get in gear, we would be there arguing still.

I can easily envision the maintenance responsibility for FORTRAN being wrested from the United States if the committee minority continues to stonewall against needed improvements. FORTRAN is too important to the international community to be long held hostage to the current petty nonsense.

>The committee has held on to all the old constructs and added all the >new and then trying to encourage people to move...

Not exactly true. The committee seems to have brought in the notation in common use in vectorized FORTRANs, added a lot of maintainability constructs, and used the usual deprecation route to tell FORTRAN users: "Don't use these old constructs any more, they cost more than they are worth, and, on a schedule herewith published, we plan to remove them from the language." They have NOT "held on to all the old constructs", they have used the committee's long standing standard practice for getting rid of them.

>People being people they don't want to go back and change the old dusty >decks ... they work ... why fix them.

This one I am sure is a red herring over the lifetime of the planned changes; at least 10 years, more likely 20, will pass before the changes that make the "dusty decks" uncompileable will finally take effect. I worked recently in a major programming shop (200 active coders) with a 6.5 million line inventory of production code, about 80:20 COBOL to FORTRAN. Essentially NONE of it sat for as long as 5 years without change, and with each change, we had to pay for the lack of maintainability constructs in two of the oldest languages in common use.

In fact, as part of a mainframe change that changed the word-size of our computers, we had to rewrite ALL of the code in 18 months (we hired expensive help). I think this experience is common; all of the FORTRAN dusty deck has had to undergo maintenance to run on 64 bit machines, then again to run on vector machines, then again to take full advantage of IEEE standard 80 bit floating point. Those decks don't sit still long enough to gather dust, and every time they move, they cost extra because FORTRAN lacks data abstraction, strong typing, records, type checking across compilations, adequate flow control mechanisms, and all else that the committee is offering in the way of maintainability improvements.

>One of our arguments has always been that when people are faced with >a major rewrite, they will go with a different language which solves >their problem instead of using one that is full of quirks...

My experience in the above major change was that we had Pascal, Ada, and many fourth generation language "COBOL generators" available for use, but we had a shop full of FORTRAN and COBOL programmers. Inertia and good sense meant that we upgraded our code to the latest standard (COBOL 74, FORTRAN 77) in the process of making the move, but did not change languages for

most of the code. I think the argument ignores the inertia of the mass of programmers and the reluctance of their managers to provide expensive training or take unquantified risks.

>We put together the following summary in response to some of the  
>other net traffic. I would appreciate your comments on it.

You couldn't hold me back! ;-)

>We look at the issue from two sides. The current customer base  
>and the complexity of keeping the old and the new in the same  
>compiler.

Unless you also look at the higher cost of "staying put", you have done yourselves and your customers a large disservice. The problem of carrying along deprecated features has always been with the FORTRAN community, but the community has, up to now, chosen to bear that cost because of the smooth transition it allows from the old to the new compilers.

>We also see holes in what is being proposed. Some of the things look  
>great on the surface ... but are not complete as in the case of array  
>notation or will not allow optimal compiler execution and code  
>execution as in derived data types.

Things which are incomplete should be addressed with example language showing how to extend them to be complete, in your comments, not recommended for exclusion because they are not perfect.

Again, if you keep this narrow focus, you do a disservice to your customers. The array notation you show below, and argue against in the cause of "excessive keystrokes", looks like the vector notation I have seen in common use in vectorizing FORTRANs; complying with the committee's mandate to standardize standard practice. The Ada committee was a lot wiser about the question of "extra keystrokes", they explicitly included in their goals a language that was harder to write but thereby easier to maintain.

Lack of record structures in FORTRAN causes nightmarish convolutions in code; either the pieces of a record must be moved and accessed in separate statements, increasing the bulk of the code, or the pieces are forced into a single array of inhomogeneous types of data, making a maintenance headache of immense magnitude.

>My thought is that the committee should reconfirm FORTRAN for  
>another 5 years, clean up this new proposed standard removing  
>the things they want to deprecate, etc., fix the problems in  
>areas of new functionality ... and call it language X. Let  
>FORTRAN stay FORTRAN and move the world to language X.

This won't work at all; names have a strong magic: it was not technical superiority or competitive pricing or excellent support that made their microcomputers outsell all others, it was the three letters "IBM". Similarly, if it isn't "FORTRAN", but it still contains FORTRAN's weaknesses, no one will go to your language X. It is the programming language going by the name of FORTRAN that needs improving. Your proposal says "do nothing" for FORTRAN. Yuk! FORTRAN is in desperate need of help; the committee proposal looks like, and is, radical surgery. The alternative is the continuation of the patient's lingering illness.

>Here's the summary. I would appreciate comments...



I took the liberty of reformatting this to ease comments, and correcting a raft of typos. GNUemacs is so nice!

64

> Summary of Major Additions Proposed in FORTRAN 8x

>This summary that follows is the Forward in the FORTRAN 8x draft:

>Array Operations:

>Computation involving large arrays is an important part of  
>engineering and scientific use computing. Arrays may be used as  
>entities in FORTRAN 8x. Operations for processing whole arrays and  
>sub arrays (array sections) are included in the language for two  
>principle reasons: (1) these features provide a more concise and  
>higher level language that will allow programmers more quickly and  
>reliably to development and maintain scientific/engineering  
>applications, and (2) these features can significantly facilitate  
>optimization of array operations on many computer architectures.

>The FORTRAN 77 arithmetic, logical, and character operations and  
>intrinsic functions are extended to operate on array-valued operands.  
>These include whole, partial, and masked array assignment,  
>array-valued constants and expressions, and facilities to define  
>user-supplied array-valued functions. New intrinsic functions are  
>provided to manipulate and construct arrays, to perform  
>gather/scatter operations, and to support extended computational  
>capabilities involving arrays.

>Numerical Computation:

>Scientific computation is one of the principal application domains of  
>FORTRAN, and a guiding objective for all of the technical work is to  
>strengthen FORTRAN as a vehicle for implementing scientific software.  
>Though nonnumeric computations are increasing dramatically in  
>scientific applications, numeric computation remains dominant.  
>Accordingly, the additions include portable control over numeric  
>precision specification, inquiry as to the characteristics of numeric  
>information representation, and improved control of the performance  
>of numerical programs (for example, improved argument range reduction  
>and scaling).

>Derived Data Types:

>"Derived data type" is the term given to that set of features in this  
>standard that allows the programmer to define arbitrary data  
>structures and operations on them. Data structures are user-defined  
>aggregations of intrinsic and derived data types. Intrinsic  
>operations on structured objects include assignment, input/output,  
>and use as procedure arguments. With no additional derived-type  
>operations defined by the user, the derived data type facility is a  
>simple data structuring mechanism. With additional operation  
>definitions, derived data types provide an effective implementation  
>mechanism for data abstractions.

>Procedure definitions may be used to define operations on intrinsic  
>or derived data types and nonintrinsic assignments for intrinsic and  
>derived types. These procedures are essentially the same as external  
>procedures, except that they also can be used to define infix  
>operators.

I have a problem with this; infix procedures should be  
importable from modules; this seems to say that they are not;  
reference in contrast the Ada standard. If I define a data

type in a module, and overload the '+' and '\*' operator for that data type, it does me little good if I cannot export that overloading for use by other compilation units.

64

>Modular Definitions:

>In FORTRAN 77, there is no way to define a global data area in only  
>one place and have all the program units in an application use that  
>definition. In addition, the ENTRY statement is awkward and  
>restrictive for implementing a related set of procedures possibly  
>involving common data objects. Finally, there is no means in FORTRAN  
>77 by which procedure definitions, especially interface information,  
>may be made known locally to a program unit. These and other  
>deficiencies are remedied by a new type of program unit that may  
>contain any combination of data object declarations, derived data  
>type definitions, procedure definitions, and procedure information  
>information. This program unit, called a MODULE, may be considered  
>as a generalization and replacement for the block data program unit.

This is poorly stated. The MODULE may supersede the BLOCK DATA subprogram, but they are not closely enough related to consider it a generalization of the BLOCK DATA. It would be better said that it gives FORTRAN for the first time a data abstraction mechanism.

>A module may be accessed by any program unit, thereby making the  
>module contents available to the program unit. Thus, modules provide  
>improved facilities for defining global data areas, procedure  
>packages, and encapsulated data abstractions.

>Language Evolution:

>With the addition of new facilities, certain old features become  
>redundant and may eventually be phased out of the language as use  
>declines. For example, the numeric facilities alluded to above  
>provide the functionality of double precision; with the new array  
>facilities, nonconformable argument association (such as associating  
>an array element with a dummy array) is unnecessary (and in fact is  
>not useful as an array operation);

This last is unclear; it may be language from the FORTRAN 77 standard, but it is compiler writers' terms of art, not programmers'.

>and BLOCK DATA program units are redundant and inferior to modules.

>As part of the evolution of the language, categories of language  
>features (deleted, obsolescent, and deprecated) are provided which  
>allow unused features of the language to be removed from future  
>standards.

Good. This is as it has been done in the past and will be in the future, one hopes.

> Our Comments

>Array Operations:

>Array operations allow the user to write simpler code in some cases.  
>There are also cases where the code may be more complicated to write  
>or may not be possible to write using the array operation notation.

You have missed the purpose of the notation; it allows the user to indicate in a simple way to a vectorizing compiler the

gather and scatter operations desired. The trade off is a little more typing to achieve a regular notation which can be compiled easily to (much) more efficient code.

64

>Examples:

```
> A = B * C      is what people think of as array operations
> DO I = 1,100,2
> DO J = 1,100,2
>   A(I,J) = B(I,J) * C(I,J)
>   D(I,J) = E(I,J) * A(I,J)
> ENDDO
> ENDDO
```

You can't say this in FORTRAN 77 (by my textbook). Since your proposal is to leave FORTRAN 77 unchanged and create a new language, you must compare against the FORTRAN 77 standard, not the parts of FORTRAN 8X which you most admire.

> becomes:

```
> A(1:100:2,1:100:2) = B(1:100:2,1:100:2) + C(1:100:2,1:100:2)
> D(1:100:2,1:200:2) = E(1:100:2,1:100:2) + A(1:100:2,1:100:2)
```

But you are indicating a complex situation where the gather/scatter operations need to be explicated to the compiler by the programmer. Is not  $A = B + C$  equally valid for conforming arrays, and does it not require the same amount of typing to create this loop in the old notation? Surely the gain is in the normal case, and more than makes up for the loss in the more complex case?

> The DO loop contains 72 characters and the array notation contains 112 characters.

But also, the DO loop to add the whole arrays contains (more like 84 characters in FORTRAN 77) the same number of characters, and yet  $A=B+C$  needs only 5? You are not giving a fair comparison here.

> If you used identified arrays to define dynamic aliases for the 5 arrays above, then the statements would be simplified, but you'd have 5 IDENTIFY statements and 5 new names to add to the program.

But you would then make this savings everywhere that you used the aliases. If that gather/scatter pattern were important to the problem being computed, it would be unusual indeed to see it used only once in the program. Again, properly analyzed, the gain is to the committee's proposal.

```
> DO I = 2,100,2
> A(I-1) = B(I) + C(I/2) * D(102-I)
> ENDDO
```

> becomes:

```
> A(1:100:2) = B(2:100:2) + C( :50) * D(100:2:-2)
```

> Which is easier to read and understand in the DO loop form. The bottom example saves 2 characters.

Again, you miss the point that the more regular notation immensely eases the task of a vectorizing compiler. It is nearly an AI problem (and probably provably NP complete), to

extract the assignment conflicts from a complex DO LOOP, while the committee's notation, in successful use in vectorizing compilers today, makes the task very easy.

64

```
>  
> DO I = 2,100  
> A(I) = A(I-1) + B(I)  
> ENDDO
```

> Cannot be represented in array notation.

> A(2:100) = A(1:99) + B(2:100) ! answer is different to the DO loop

Certainly, but this is not an array operation; this is an array element operation, and is a classical example of needing the result of the (i-1)st step before computing the (i)th. This is hardly an argument against the committee's proposal, but is instead just a textbook example of why it is difficult to analyze DO loops for vectorizability. In fact, it argues against you, since, by providing the array notation for use by the programmer where vectorization is appropriate, the compiler writer should be no longer obligated to analyze DO loops for vectorizability, immensely easing the task of writing a vectorizing compiler for FORTRAN.

> Thus, not all array constructs can be represented in array notation.

Again, that is not an "array construct", it is an elementwise operation on an array, not at all the same thing.

> Other issues include:

> 1. Array subscripts are not allowed in array notation.

Unclear.

> 2. Implied array temporaries may increase execution time.

If they were needed anyway, there is no loss. If they were not needed, then the compiler writer has not done his job correctly. If they allow vector operation where it was not possible before, and the execution slows, replace the hardware!

> 3. Dope vectors will be required with function invocations increasing call overhead.

Since most usage of FORTRAN today is moving toward vector hardware (why else use such an antique language with all its problems if you don't have big problems requiring fast execution times?), the use of "dope vectors" is already either standard practice or taken care of in hardware. This is a non-problem.

>Numerical Computation:

```
> REAL (PRECISION=10, RANGE=50) TEMPERATURE  
> D = DIGITS (TEMPERATURE)  
> E = EFFECTIVE_PRECISION (TEMPERATURE)
```

> Allows the program to determine attributes about the underlying implementation and allows support for more than 2 underlying REAL data types. Aids in moving programs to machines with different word size.

> Issues include:

> 1. May greatly increase the number of specific intrinsic functions

I think standard usage is to use the next bigger type that fits. Poof, no problem, you already have that intrinsic function written.

64

- > 2. "If no method exists that satisfies the specified precision and exponent range, the results are processor dependent" Ada compilers are required to flag such situations as an error.

And, certainly, so should FORTRAN compilers; the purpose is to let the programmer tell the compiler his real needs for numeric precision for this data type; unless he is lying, the program may be expected to fail, so why proceed to link and execute it?

>Derived Data Types:

- > TYPE PERSON
- >     INTEGER AGE
- >     CHARACTER (LEN=50) NAME
- >     REAL, ARRAY (2,2) :: RATES
- > END TYPE PERSON

This is very pretty. However, it fails to discriminate between typing to form an aggregate, and typing to differentiate usages of intrinsic types. To see the problem, consider defining just a person's name to be of TYPE character length 50. What I want to do is:

```
TYPE
  CHARACTER (50) NAME
END TYPE
```

but I doubt this would compile. There needs to be a keyword to indicate a typing declaration, and a separate keyword to indicate an aggregation. Standard practice is TYPE and RECORD in Pascal or Ada or Modula-2, typedef and struct in C. Probably the former would be preferable for FORTRAN.

- > TYPE PERSON CHAIRMAN

This, however, is not pretty. It should be clear in reading that this is a variable of type person being declared. It looks more like a redeclaration of person to use a data element of type chairman. The committee should take cognizance of reality; most future FORTRAN users will have had their computer language training in languages which are not line oriented, and so visual cues relating to line structure, easy for old FORTRAN programmers to grasp, will be imperceptible to students trained in modern, free format languages. A much better choice would be VAR, widely used, instead of TYPE. If I understand the use of "::" in the proposal:

```
VAR CHAIRMAN :: PERSON
```

would be better.

- > CHAIRMAN%AGE = 50

And this is an abomination. Almost every language that uses a record structure uses the period to indicate a substructural element, why make life hard for the multiple language user? The period provides a visually obvious break, easing reading and thus the maintenance task. The percent sign is non-standard and hard to find in the middle of a word. Yuk!

- > Provides a record data structure for FORTRAN (which is different from the record data structures currently available in VMS FORTRAN.)

No vendor is guaranteed that his particular implementation of a non-standard feature will become the one accepted in the next standard. All he can expect is that, if successful, something with the same functionality will probably make it into a future standard. Sounds like sour grapes to me. ;-)

64

- > Issues:

- > 1. Overloaded operators can make programs harder to read and understand.

Wrong! FORTRAN already massively overloads the arithmetic operators; this does nothing but make the language easier to use and read.

- > 2. Generic user functions can cause an explosion in the object code if the function has many arguments.

No worse than a procedure call; what's the problem?

- > 3. The compiler cannot optimize operations on derived data types in the same fashion as with intrinsic data types. For example, if the BIT data type is defined in the language, the compiler can generate optimized code to deal with this data type. You can do a BIT data type using derived data types, but the compiler will not have the same amount of information available for optimization since the derived data type is a generalized function.

Only partially true. Operations such as assignment can be done by a move bytes operation, which is usually quite fast, and this in turn can be done mostly as a move words operation if it is long enough to be worth the trouble. User defined operations must be done using the code in the user's MODULE, but if not done that way, they would be being done instead, by inline, user written code multiple places in the main routines, with concomitant larger code sizes, confusion, and major maintenance problems when the logical object being manipulated changes due to maintenance activity.

- > Modular Definitions:

```
> MODULE POOL1
>   INTEGER X(1000)
>   REAL Y(100,100)
> END MODULE
```

- > USE POOL1

> Allows for better modularization of programs. Interface errors between modules will be caught at compile/link time.

- > Issues:

- > 1. The dependent compilation model has not been tested in the FORTRAN arena and is not like the Ada dependent compilation.

Nevertheless, dependent compilation is now a well known, smoothly functioning technology which has an incredible payback in independent maintenance activities and reduced

maintenance costs. This is the foundation of data abstraction, sorely needed by FORTRAN for many years. Gaining the module construct for FORTRAN is worth a great deal of cost in compiler redesign.

64

- > 2. Increases compilation complexity and requires changes in other areas of the system software such as the linkers and loaders.

Yes, but the payoff is well worth the cost.

- > 3. Will cause compilers to be slower. Some argue that faster machines will overcome this, but if I have a FORTRAN 77 and a FORTRAN 8x compiler on the same machine, the FORTRAN 8x compiler will have to, by definition of the standard, do more work to compile the same program.

If the program uses only FORTRAN 77 constructs it will still compile under the old compiler. Poof! If it uses new constructs, it will only compile under the new compiler. Poof! If the old compiler is thrown away, and the old code is compiled under the new compiler, it will exercise only those portions of the compiler pertaining to the old constructs. Unless the compiler is HORRIBLY written, "Poof!" again; not a problem.

- > 4. The INCLUDE statement, which performs the function of allowing a single definition of common code, is NOT a part of this proposed standard.

There is no need to repeat code throughout the program: the sole use for that in FORTRAN 77 is to propagate COMMON declarations. COMMON is superseded by the MODULE; incorporating INCLUDE will only delay adoption of the MODULE. (I know for a fact that all the compiler vendors will continue to carry INCLUDE as a non-standard feature, but why put it in, just to deprecate it in the next standard release? Dumb.)

> Everyone has the right to form their own opinion about the proposed > FORTRAN 8x standard. Remember that if you are going to give your > opinion in writing to the committee, that letters must be received > in Washington DC by February 23, 1988. The address is:

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

Kent, the man from xanth.

## LANGUAGES

### Key Vendors Make Final Push To Kill FORTRAN 8X Proposal

But as the comment period draws to a close, user opposition to the proposed FORTRAN 8X standard has been mild, if not negligible, despite suppliers' urgings.

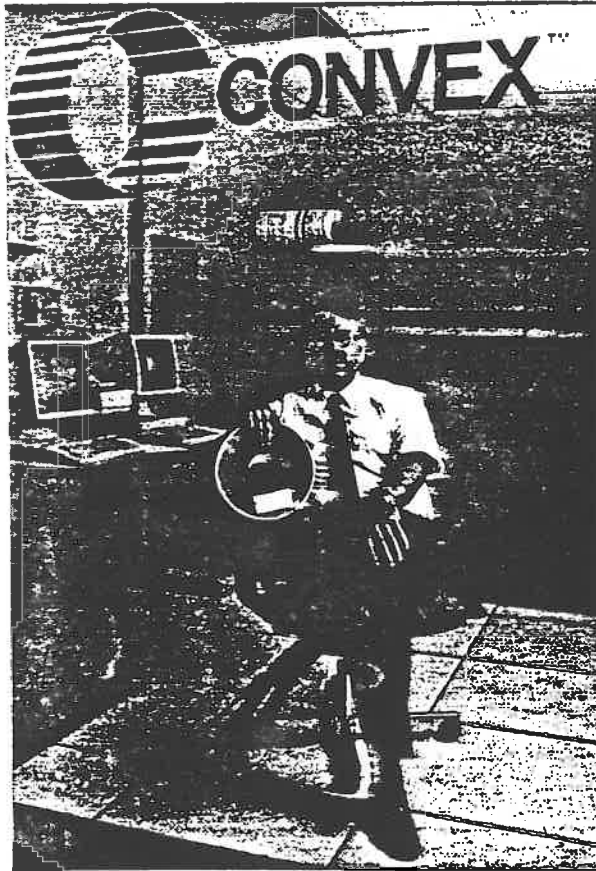
BY KAREN GULLO

As the public comment period for the proposed FORTRAN 8X standard draws to a close, vendors that oppose 8X hope their three-month effort to rally interest and support from their customers will unleash a flood of negative comments from the user community and send 8X reeling back to the American National Standards Institute (ANSI).

The vendors—IBM, Unisys, Digital Equipment Corp., Data General, and Convex Computer Corp.—haven't been shy about trying to persuade their customers to become active participants in the 8X debate. IBM, in an unprecedented move, sent letters to 2,500 customers, urging them to review the standard and send comments to the American National Standards Institute's X3 committee, which developed 8X.

At press time, Unisys was readying a letter to its 40,000 A-series and 1100 mainframe customers worldwide that aims to encourage them to obtain a copy of the proposed standard, review it, and respond to ANSI. DEC took the issue to its user group, and, in mid-December, the U.S. Chapter of DECUS drafted a comment paper that generally opposed the new standard. Convex, based in Richardson, Texas, has published articles in its user newsletter exhorting customers to write letters. Data General says it plans to send letters to its customers in the near future.

For all these efforts, response among end users to



SMITH OF CONVEX: He's not discouraged by the lack of user response.

8X has been less than overwhelming. Just 17 responses had been filed with X3 by late December.

Seventeen is not even close to the number of responses that were generated a few years ago in the debate over COBOL 85. However, FORTRAN 8X opponents say they are hopeful that many more than 17 comments will be filed with ANSI when the

comment period expires. The vendors in opposition to 8X contend that the proposed standard, which adds 33 new statements to FORTRAN 77, renders the language too large and complex—and therefore expensive—to implement. They also argue that the "decrementing" of 14 statements, some of which are widely used, would not be in the best interests of users

(see "The 14 Statements").

The whole issue came to a head at the last X3 committee vote, when IBM, DEC, Unisys and DG, all of whom have representatives on the committee, voted against the new standard and got very vocal on the issue (see "Opponents of FORTRAN 8X Are Facing an Uphill Battle," Oct. 15, p. 22). The proposed standard went into a four-month comment period, ending in February.

Some opponents are undaunted by the relative lack of response from the user community. Says Presley Smith, manager of development services at Convex and one of FORTRAN 8X's most vocal opponents, "I'm not discouraged as this point. IBM letters did not go out until mid-November, and that will influence the number of responses significantly. I think our efforts will fail if we don't get at least 2,000 responses, but let's put it this way, IBM sent out more than 2,500 letters. I'm not discouraged."

Others are not so sure. Roy Jones, program manager of the 1100 environmental software group at Unisys, says that time is running out. "Ordering the proposal document, reviewing it, and presenting it to your management to get approval for a response, that's going to take some time."

#### Users Opposed COBOL Standard

Although many observers in the industry liken the FORTRAN 8X debate to the controversy of the early '80s over the new COBOL standard, which dragged on for four years at the public comment stage, the difference is that users led the charge against the proposed new standard, inundating ANSI with almost 2,000 negative comments. COBOL was sent back to the drawing board, and a new standard was adopted in '85.

"I'm disappointed in the response we've gotten [for





PLYLE OF AMWAY: FORTRAN 77 code is running fine, so why change it?

FORTRAN 8X]. It doesn't compare to the amount we got with COBOL," says Cathie Kachurik, director of the X3 Secretariat at CBEMA. "It's just not getting the thrust we thought it would. But we could get an influx of comments in February. I think it's very likely we will."

Vendors aren't the only ones that want to mobilize users. GUIDE International has sent letters to a VIP list of its members who are in management. The letter, according to Frank Kirshenbaum, GUIDE representative to X3, does not take sides in the debate, but simply alerts users about the issue. "FORTRAN is not a language we have a formal program on," says Kirshenbaum. "We don't care if people are opposed to it or support it. We're saying, 'This is

something you should care about.'"

Although it's hard to say which side has the support of a majority of users, there's definitely a split in the user community over 8X. Generally, the division follows a pattern: users who are writing mostly new code and/or using parallel machines or supercomputers favor the new standard. Among the reasons why is their not having to worry about compatibility with existing FORTRAN 77 code and 8X's inclusion of features for array operations and numerical computations.

At the other side of the spectrum are users who are running mostly FORTRAN 77 applications. Although 8X is compatible with 77 (it contains all the same features of 77), it adds so many new fea-

## The 14 Statements

Vendors and users opposed to FORTRAN 8X take exception to the decrementing of the following 14 statements. Decrementing statements would be subject to removal in a future revision of FORTRAN. Removal of some statements would affect significantly the entire definition of data structures in a program.

**BLOCK DATA**—used for entering data and initializing common blocks.

**ENTRY**—defines the entry point to a subroutine.

**COMMON**—specifies global data pools.

**EQUIVALENCE**—states that two or more data objects of different types share the same storage region.

**DIMENSION**—allows for the definition of arrays of data.

**DOUBLE PRECISION**—allows greater precision in a data type.

**Old DATA**—an old form of data statement, considered redundant in functionality.

**Arithmetic IF**—an old form of IF from FORTRAN 66, which will be replaced in 8X by logical IF or block IF.

**ASSIGN**—a label to be dynamically assigned to an integer variable.

**ASSIGNED GO TO**—allows indirect branching through a variable.

**Alternate RETURN**—introduces a label into an argument list in order to allow the called subprogram to direct the execution of the caller upon return.

**PAUSE**—an old statement, unnecessary with new hardware.

**Computed GO TO**—to be replaced by a CASE construct.

**Statement Function**—to be replaced with an internal function definition.

tures that it may require programmer retraining and a short-term decrease in programmer productivity. In addition, 8X eventually will make certain statements obsolete, such as COMMON and EQUIVALENCE, which will require rewriting code to remove these features.

### Vendors Will Have To Speed

Falling into the first category is Jerrold Wagener, director of computing research at Amoco Production Co., Tulsa, Okla. He oversees 50 researchers running oil exploration programs on Sun and Apollo workstations, Alliant minisupercomputers, IBM 3090s with vector facilities, and Crays.

Wagener, an X3 member who voted in favor of 8X, acknowledges the opposition's

charges that the proposed standard is a "new language. This is not a minor revision. It's a fairly major one. It will require investment on the vendors' part to implement new compilers, but it's well worth it." He adds, "8X contains functions that scientific programmers need, such as the array facility. If the efforts of vendors like IBM and DEC delay the standard unduly, they will have done the scientific research community a great disservice."

Wagener says his group at Amoco has held regular meetings to follow 8X developments, and he says it is in favor of the new standard. He says, however, that the supporters of 8X are not mounting a mobilization campaign like that of the opposition.

Those in favor of 8X in-

NETWORKS

# Lloyd's of London Attempts To Insure Its Future

The world's largest insurance market is aiming to shed its stolid image by fine-tuning the way business is done worldwide, and IBM is helping and learning from it.

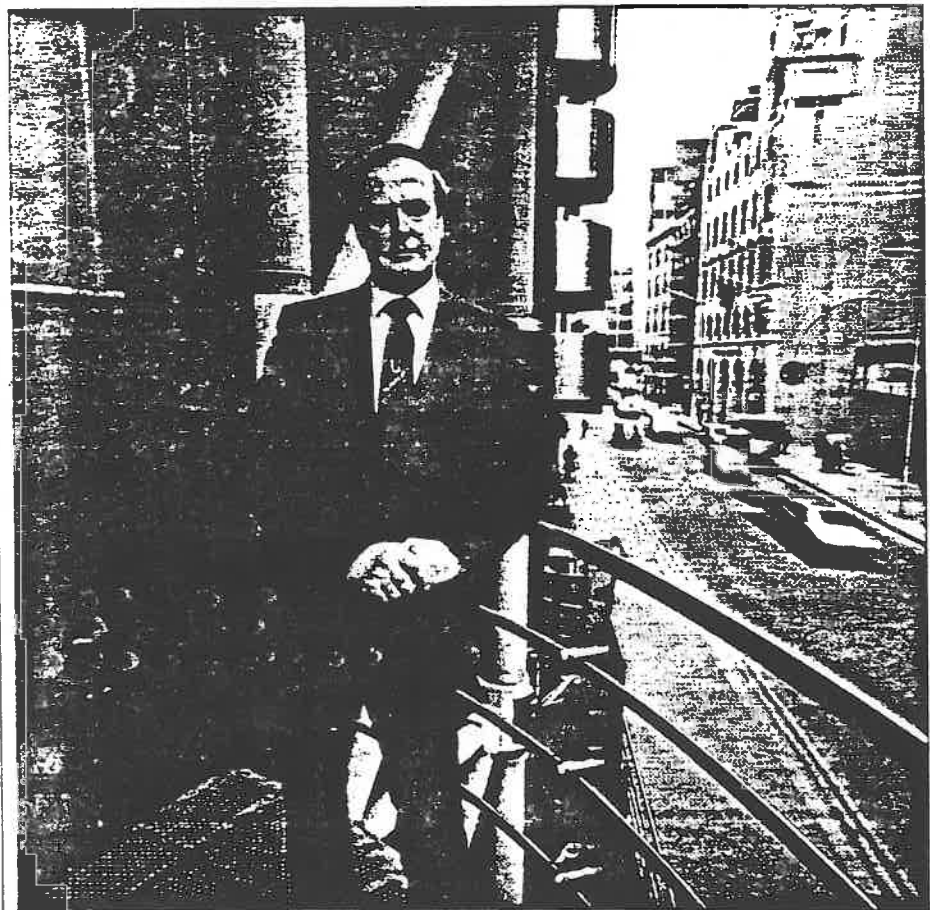
clude Grumman, the National Center for Atmospheric Research, and Hewlett-Packard.

Users such as Wagener seem willing to accept the complexity of 8X as a fair trade-off for the benefits the added features will give them. Wrote one user who sent ANSI a positive comment: "Implicit in the new proposed FORTRAN is that the current standard FORTRAN 77 is scheduled for a complete overhaul and upgrade. This makes me slightly uncomfortable, but that's the price of progress . . ."

On the other hand, there are many users like Patte Pyle, a technician in quality assurance at Amway Corp., Ada, Mich. "We have many applications written in FORTRAN 77. The new FORTRAN decrements 14 statements currently recognized by FORTRAN 77. We'd have to do a whole lot of rewriting if these statements are no longer supported," complains Pyle. "The code is running fine. Why change it?"

Next month, the X3 committee will have to respond to all public comments. It's not the quantity of comments that can influence the passage or revision of the standard, but the content. So far, the committee has chosen to proceed with 8X, despite the vendor opposition, but a ground swell of negative comments from users could send 8X back to the technical committee.

In the meantime, some observers feel that the most recent round of disagreement over 8X is unnecessary and serves neither vendors or users. "The X3 committee would have been doing itself a favor if it had gotten more consensus before going to public comment," remarks Kevin Harris, a DEC representative to the ANSI X3J3 subcommittee of 3X. "They are going to end up with a compromise that doesn't serve either party and screws the user to the wall." ■



LLOYD'S OF LONDON'S LEE: He hopes the network will attract foreign business to London.

BY JOHN LAMB

Lloyd's of London, the world's oldest and largest insurance market, is part of a group setting up a global electronic trading network that it hopes will streamline the way the world's international insurance business is run. In the process, the project is al-

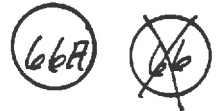
ready streamlining the way that IBM, which has won the Lloyd's contract—eventually to be worth \$18 million (£10 million) a year—is developing its managed data network service products.

Des Lee, the head of systems and communications at Lloyd's, expects that by the middle of the year, some 80%

of the Lloyd's volume of data will be carried by the network. Through this network, Lee says, Lloyd's hopes "to protect London as the number one insurance center, to administer the flow of premiums and the settlement of claims, and to cater for the bureaucracy of a 300-year-old tradition."

Photograph by Philip Ryan

P. 455



James H. Matheny  
41 Silver Spring Drive  
Rolling Hills Estates, CA 90274  
(213) 375-5940  
February 4, 1988

107JHM-02

Subject: Review. MOSI draft Standard, P855/draft 7, 11/1/87

*We* *X3J3 has (by a vote of 30-2)*  
In reviewing the MOSI (Microprocessor Operating System Interfaces) draft Standard, ~~I~~ have reluctantly reached the conclusion that the draft is not acceptable to Fortran (ANSI X3J3). ~~I~~ would require that Appendix F, Fortran Language Binding, be deleted from the proposed draft MOSI Standard. It does not adequately reflect Standard Fortran (ANSI X3.9-1978).  
9

Specific problems include:

1. Standard Fortran includes only one INTEGER type. The Standard does not specify the precision of an integer, but requires that it occupies the same memory as the single precision REAL data type when such a variable appears in a COMMON or an EQUIVALENCE statement. Thus, effectively, the integer type is long integer in MOSI terms.
  - a. Many Fortran processors extend the Standard to include two kinds of integer, a two byte integer and a four byte integer. The syntax of the extensions that I know about is INTEGER \*2 and INTEGER \*4, without the parentheses of MOSI.
  - b. As indicated above, the equivalence would be INTEGER\*4 to long integer and the default INTEGER, and INTEGER\*2 to the MOSI integer.
2. The Fortran type CHARACTER does not map into the MOSI (and PASCAL) string. The Fortran character datum does not contain a byte which contains the length of the item. Fortran character items are not limited to a length of 256 characters per item. Indeed there is no way in Standard Fortran to create a data structure that contains the length as a character position. The structuring capability of Fortran 77 consists only of COMMON and EQUIVALENCE, and these disallow the mixing of character and non-character items.
3. Standard Fortran provides for six character names only. There is no provision for a seventh character, meaningful or not.

66A 66

4. Standard Fortran does not provide for in-line comments. (The draft MOSI Standard description could take the position that this usage is for the description only. However, in this context, it should use the Fortran 8X (see below) exclamation (!) punctuation to indicate commentary, and not the semi-colon.)

5. There is no way in Standard Fortran to return a function result that is a pointer to an allocated area of memory. Thus sections F3.1 and F3.2 can not be achieved in Standard Fortran. They must be omitted.

6. I think that the result DATA of I/O reads and writes should be of type Fortran CHARACTER. Then the Fortran internal file capability can be used to convert fields of the item to or from character, real, integer etc. There is no way in Standard Fortran to consider character "under the guise of" integer. This comment includes the function calls OSRDDDA, OSWRDDA, OSREAFI, OSWRAFI, OSIREAD, OSIWRIIT, and the Message invocations etc. (There seems to be an inconsistency in the spelling of OSREAFI -- should it be OSRDAFI?)

Fortran 8X will resolve some of the problems cited above. Draft X3J3 /S8-104-June, 1987 is now in the "public review and comment" period, and may become an ANSI Standard replacing ANSI X3.9-1978 in the short-time future.

1. Fortran 8X now has a precision specification mechanism for type REAL. ~~Prior to the final stage of the standardization process, this language may be applied to integers.~~

2. Fortran 8X provides a convenient mechanism for derived types and data structures. ~~A STRING capability like PASCAL can be easily implemented if the integer precision is extended to include something like byte.~~

3. Fortran 8X names may be dual case, may be 31 characters long, may have embedded and trailing, but not leading, underscores (\_),

4. Fortran 8X provides in-line commentary, from the exclamation point (not in a character context) to the end of the line.

5. Fortran 8X provides a facility to ALLOCATE arrays, but not scalars or structures. ~~A pointer facility may be added to the Standard before its final release.~~

6. The comment #6 above should remain valid in Fortran 8X.

In the description of event handling, I did not see the

(left)

~~(right)~~

possibilities of continuation after the processing of an exception. If you expect continuation at the point of interrupt, this can inhibit optimization severely. I realize that MOSI describes only an interface, but I think that this area of concern should be surfaced.

We tried to get file and record locking into Fortran 8X, but failed totally. I am glad to see it here.

I am surprised that the COBOL appendix cites an obsolete Standard. The current COBOL Standard is X3.23-1985.

p. 459

67

~~43~~

107-EAJ-1  
15 January 1988  
Page 1 of 2

To: X3J3  
From: E. Andrew Johnson  
Date: 15 January 1988  
Subject: Fortran-77 Interpretations  
Ref.: 106-104(X3J3/216)

Enclosed is a request for interpretation from Richard R. Moore at Michigan State University. He is requesting that X3J3 make some statement about *reasonable* minimal requirements on standard conforming processors. His point is well taken, and we should consider adding such limits for Fortran-8x. The draft proposed "C" standard has many (perhaps too many) such limits defined, but only on the properties of source programs. No such limits are imposed on the operating environment.

I would like to discuss the possibility of adding such limits at the February meeting.

With regard to the interpretation request, section 12.2.2 File Properties only mentions processor-determined *access methods, forms, and record lengths*. There is no mention of processor-determined *file sizes* (number of records) in the standard, although it does discuss the *position* property of files.

If the limit of 50 direct access records is a function of the host file system, then I would agree that the implementation is standard-conforming. If, however, the limit is artificially imposed by the Fortran I/O system, then I would have to agree that the limit is arbitrarily small (perhaps an internal table size), but still standard-conforming, as long as the limit is clearly documented. Fortran applications developed on such an environment will easily port to a less restrictive environment.

Does anyone know the minimum number of direct access records required to pass FCVS (is it, perhaps, 50)?

P. 460

67

~~43~~

MICHIGAN STATE UNIVERSITY

COMPUTER LABORATORY  
COMPUTER CENTER

EAST LANSING • MICHIGAN • 48824-1042

December 30, 1987

E. Andrew Johnson  
Mail Stop 10C17-3  
Prime Computer Inc.  
500 Old Connecticut Path  
Farmingham Massachusetts 01701

Dear Mr. Johnson:

Jerry Wagener gave me your name as the person to request a ruling from X3J3 on whether an implementation conforms to the Fortran 77 standard.

Several years ago X3J3 stated that a processor that failed to accept 57 sets of nested parentheses still conformed to the standard. However, the purpose of the standard "is to promote portability of FORTRAN programs for use on a variety of data processing systems." from the purpose paragraph in the 1978 standard. In order for the standard to accomplish its purpose, there must be some minimal requirements on the size and complexity of a program that a standard conforming processor must accept. For example, if the processor in question had failed to accept three sets of nested parentheses, it surely would violate the purpose of the standard and therefore be non-conforming.

My case is a processor that does not permit a record number of a direct access file to be greater than 50. When writing a NEW direct access file, this processor requires the record number must be less than 50 unless some action outside of the processor is taken. This is a ridiculously low value and is the I/O equivalent of three levels of nesting. The vendor claims that this compiler conforms to the ANSI 1978 standard and uses the ruling noted above as justification. X3J3 should declare such a compiler non-conforming, in violation of the stated purpose of the standard. Neither DEC's VMS Fortran, SUN's Fortran, nor the various Fortran compilers written by CDC have such a restriction. Clearly, this restriction makes a mockery of portability.

If X3J3 chooses to declare this as standard conforming, then please consider adding "reasonableness" limits to the Fortran 8x standard.

Thank you for your time, help, and consideration.

Sincerely yours,



Richard R. Moore

P. 461

cc: J. Wagener, P. Wolberg, B. Moncrieff, L. Meissner (Forum)



~~68~~

682

107-EAJ-2

14 January 1988

Page 1 of 1

To: X3J3  
From: E. Andrew Johnson  
Date: 14 January 1988  
Subject: Deprecated Features  
Ref: WG5-L11 (Liverpool Resolution)

Resolution WG5-L11 (Decremental Features) states the following:

That WG5 recognizes the motivation for splitting decremental features in Fortran 8x into two different classes.

However, WG5 requests X3J3 to consider the possibility of identifying both obsolescent and deprecated features in the text of the standard, consistent with the requirement imposed on processors for detecting both classes of decremental features.

Since:

- the deprecated features are only described in Appendix B (which is *not* part of the standard, and
- the deprecated features list only serve as guidance for the next revision committee.

I would propose instead that a standard conforming processor only be required to contain the capability of detecting the use of deleted or obsolescent features (*not* deprecated). This would make the text consistent, in that features which may possibly be removed at the next revision are highlighted (lowlighted?) in the text of the standard and would be similarly flagged by a standard conforming processor on request.

#### Proposal

Page 1-2, line 6: Change ", obsolescent, or deprecated" to "or obsolescent". Also in lines 12-13.

End of Proposal

#### Discussion:

This is definitely a substantive change to the draft, despite the simple change of wording. I realize that this is actually weakening the effect of deprecated features somewhat, but it is presumptuous of X3J3 to mandate that the features which we have identified as deprecated must be detected in a standard conforming processor upon the adoption of this standard. In fact, such a change may actually increase the likelihood of public acceptance of this draft proposed standard.

p. 462

To: X3J3  
From: E. Andrew Johnson  
Date: 14 January 1988  
Subject: Re-work of Low-Level Syntax and Source Form  
Ref.: 106-29(X3J3/216); S8.104 Sections 3.2 and 3.3

In reviewing the S8 at meeting 98, the editorial committee detected some descriptive text that could be interpreted ambiguously when trying to process free-form source. The subgroup attempted to re-work the text, but discovered that descriptive text was inadequate to fully describe all possible situations. Therefore, we decided to use a formal description for both the Source Form and low-level syntax, and then only use text when required for clarification. This proposal unfortunately had problems of its own and was abandoned for lack of time. The steering committee has requested that it be re-introduced for consideration during the public comment period.

Note that this proposal is strictly editorial in nature. We have taken great pains to assure ourselves that no facility has either been added or deleted from that described in the current S8. However, we have discovered that one carryover restriction from Fortran-77 could be interpreted in two different ways. We have included our choice in the proposal. A second proposal will indicate the alternative which some implementors may wish to support.

**Proposal 1:**

Make the following changes to S8.104:

1. Renumber/Move Section 3.3 to Section 3.2 and Section 3.2 to Section 3.3.
2. pg. 3-4, line 40: Change "record" to "line".
3. pg. 3-5, lines 1-17: Replace all of sections 3.3.1.1 through 3.3.1.3 by the following text:

```

R324 program-name
      is source-line [source-line]...

R325 source-line is stmt-line
      or comment-line

R326 stmt-line   is initial-line [[continuation-line]... final-line]

R327 initial-line is character [character]... [continuation]

R328 continuation-line
      is [continuation-prefix] [character]... continuation
      or comment-line

```

R329 *final-line* is *prefix* [*commentary*]  
or [*continuation-prefix*] [*character*]... non-ampersand-char  
[*commentary*]

R330 *continuation*  
is & [*blank*]... [*commentary*]

R331 *continuation-prefix*  
is [*blank*]... &

R332 *comment-line*  
is [*blank*]... [*commentary*]

R333 *commentary*  
is ! [*character*]...

**Constraint:** *commentary* is recognized only when not in character context.

**Constraint:** In character context, the *continuation-prefix* of a *continuation-line* is required.

In order to process *stmt-line* under the low-level syntax in Section 3.2 as a *stmt-text-set*, the source characters of the *stmt-line* are viewed as a single source record that would be created by:

- a. replacing each *commentary* by a single blank,
- b. removing all characters in each *continuation-prefix* and
- c. removing each ampersand in each *continuation*.

- 4. pg. 3-4, line 27: Change last occurrence of "records" to "lines".
- 5. pg. 3-4, line 27: Change "Lines" to "Records".
- 6. pg. 3-4, lines 33, 34: Replace entire sentence by the following:

A **character context** means characters between the delimiting apostrophes or quotes of character constants, or within the format item list of a **FORMAT** statement.

- 7. pg. 3-2, lines 28-30: Replace with the following text:

The low-level syntax describes the formation rules of statements of the language from fundamental parts called lexical elements. The **lexical elements** of a statement consist of symbolic names, constants, labels,

69

format items, operators and other delimiters.

The source text of a FORTRAN program consists of program statements, commentary, and statement separators. There are two sets of rules for forming such source text, yielding what is called free source form and fixed source form.

### 3.2.1 Free Source Form

The low-level syntax rules for statements written for the free source form are:

R334 *stmt-text-set* is *stmt-text* [*multiple-stmt*]...

R335 *multiple-stmt* is *semicolon* [*semicolon*]... *stmt-text*

R336 *semicolon* is ;

R337 *stmt-text* is [*blank*]... *non-delim* [*delim-pair*]... [*delim*]...

R338 *delim-pair* is *delim* [*delim*]... *non-delim*

R339 *non-delim* is *symbolic-name*  
or *constant*  
or *label*  
or *format-item*

R340 *delim* is *operator*  
or , or ( or ) or [ or ] or =  
or *blank* or : or :: or % or =>

**Constraint:** *multiple-stmt* must not follow an END statement.

**Constraint:** *stmt-text* must not contain more than 1320 characters.

**Constraint:** *semicolon* is recognized only outside character context.

The syntax rules found in Sections 4 through 12 define the sequences of *stmt-text* which constitute legal FORTRAN statements. The ordering rules in Section 2 define the order in which source statements may appear.

8. pg. 3-2, line 32: Add sentence:

Keywords are formed in the same manner as symbolic names.

P. 465

69

107-EAJ-3  
14 January 1988  
Page 4 of 4

9. pg. 3-4, line 15: Change "statement." to "statement but separated from it by at least one blank."
10. pg. 3-4, lines 24-25: Delete.

**End of Proposal 1**

In reviewing the text, it was unclear whether the restriction of 1320 characters was to apply to a single statement as viewed by FORTRAN-8X or to an entire statement sequence (*stmt-line* in the source form). If the purpose of the restriction is to specify the maximum buffer space required to process a *stmt-line* then the following proposal should be passed. This would also make the relationship between processing fixed form source and free form source more regular.

**Proposal 2:**

Replace the second constraint on the low-level syntax with the following:

**Constraint:**     *stmt-text-set* must not contain more than 1320 characters.

**End of Proposal 2**

Pr 466

70

To: X3J3  
From: E. Andrew Johnson  
Date: 5 February 1988  
Subject: Fortran-77 Interpretation on Multiple DO Terminations  
Ref.: X3J3/S1 Issue 39

At meeting 106, the committee agreed by a straw vote of 27-0-1 that the following program:

```
DO 1 I = 1, 100  
...  
IF (X .LT. 0.5) GO TO 1  
...  
DO 1 J = 1, 50  
...  
1 CONTINUE
```

Therefore, I submit the following interpretation response. Note that the CYCLE statement in Fortran 8x does *exactly* what the user had intended to do.

## Transfer of Control in Nested DO-loops

### Question:

Consider the program excerpt:

```
DO 1 I = 1, 100  
...  
IF (X .LT. 0.5) GO TO 1  
...  
DO 1 J = 1, 50  
...  
1 CONTINUE
```

Is the transfer of control to a nested DO terminator from outside the inner loop a violation of the standard? (one could maintain that permitting the transfer into a loop was just an extension). In other words, is the programmer entitled to expect his transfer going to the outer loop only because the other possible transfer is not permitted, or is the action of his program undefined?

### Answer:

A program which transfers control to a nested DO terminator from outside the inner loop is not a standard conforming program. If the terminal statement of the DO were *any* statement other than a CONTINUE statement, the expected behavior for *any* transfer of control to that labelled statement would be to execute the statement. Consider the following program excerpt:

70

```
DO 10 I = 1, 10  
...  
IF (DONE) GO TO 10  
...  
DO 10 J = 1, 10  
...  
10 A = A + 1.0
```

If the "GO TO 10" statement transferred control to the outer loop, the statement "A = A + 1.0" would not be executed, even though it would be if there were a "GO TO 10" statement inside the inner loop. This is a clear violation of the properties of the DO statement as described in the standard, which states that "execution of the terminal statement occurs as a result of the normal execution sequence or as the result of transfer of control, subject to the restrictions in 11.10.8. (Transfer into the Range of a DO-loop)"

**References:**

American National Standard X3.9-1978

<u>Section</u>	<u>Heading</u>	<u>Page</u>	<u>Lines</u>
11.10.6	Terminal Statement Execution	11-8	10-14
11.10.8	Transfer into the Range of a DO-loop	11-9	15-17

P. 468



70

DEC 28 1984

American National Standard Institute  
1430 Broadway, New York  
N.Y. 10018, USA

Date 14 Dec. 1984  
Your ref.  
Our ref. PK/BA  
Dept. Computer Install.

Comments on FORTRAN, ANSI X3.9-1978.

I am writing to you about the interpretation of the rules and restrictions concerning transfer of control in nested DO-loops with a common terminal statement. Consider the program excerpt:

```
DO 1 I = 1, 100
...
IF (X.LT. 0.5) GO TO 1
...
DO 1 J = 1, 50
...
1 CONTINUE
```

RECEIVED  
DEC 20 1984  
ANSI

In our site, which is a large research center, we use FORTRAN compilers from several different manufacturers ("Processors" in your terminology). Now we have observed that processor A transfers control to the terminator of the outer loop, whereas processor B transfers control into the inner loop, when GO TO 1 is executed. Transfer into the range of a DO-loop from outside is not permitted (11.10.8). Therefore, processor A conforms with the "spirit" of the standard, and B does not.

But my question is now, is it so that the way B works is actually a violation of the standard? (one could maintain that permitting the transfer into a loop was just an extension). In other words, is the programmer entitled to expect his transfer going to the outer loop only because the other possible transfer is not permitted, or is the action of his program undefined?



We did not find any explicit answer of these questions from the text of the standard. Hence we are very interested in hearing your comments on the matter. We might also suggest that possible future editions should contain clarifications of the rules and their interpretation regarding the outlined problem.

With best regards,

Peter Kirkegaard  
Risø National Lab.  
Computer Installation

January 14, 1988

Mr. Bruce A. Martin  
MS/B02-106  
Grumman Aircraft Systems  
Bethpage, NY 11714

Dear Mr. Martin:

In reference to your telephone call this morning, Mr. Lahey would be delighted to have you distribute his article at the February meeting.

The copy of the article you requested is enclosed with this letter, along with an extra copy, in case you need it.

Thank you for your help.

Sincerely,

*Valerie Fuller*

Valerie Fuller  
Administration

/vf  
Enclosures

**The Fortran 8x Standard**  
By Thomas M. Lahey  
Lahey Computer Systems, Inc.

107-BAM-1

Page 246

71

## Introduction

The Fortran Standard Committee has submitted a draft of the next Fortran Standard, referred to as 8x, for public review. Copies of this draft are available for \$50 from Global Engineering Documents in Santa Ana, CA, 800 854-7179. Comments should be mailed to

**ANSI X3 Secretariat  
C3EMA  
311 First Street NW, Suite 500  
Washington, DC 20001**

During public review, a four month period ending 23 February, the Fortran community submits supportive and critical opinions about the standard. The committee will review and respond to every comment. If the public response on an issue reaches sufficient proportions, the standard will be changed. The 77 Standard went through important changes towards the end: IF/THEN/ELSE was added and the extended range of the DC and stream input/output were deleted.

This article consists of three parts: a history of how 8x evolved, my public review comments and suggestions on how future standards should evolve.

## History

After the committee finished the 77 Standard, a number of the members agreed to carry on to create the next standard. They recognized there was much to do: array operations, block structures (CASE, some form of REPEAT), and various types (POINTER, STRING, BIT, BYTE); and INCLUDE were some of the important features that were not in FORTRAN 77. The goal was to complete the 8x standard by 1982.

The committee's initial plan was to develop a core Fortran concept. Modules could be added to this core, e.g., array operations might be such a module. The advantages of this approach are incrementally new standards and incrementally new compilers. This core idea evolved into providing a module program-unit and structure definition in the language as the principal means of configuring global entities and providing extensibility.

The process of completing the 8x standard for public comment was eventful. In January '86, the current draft failed to pass an internal vote because the committee felt the language was too large. The solution was to delete features (principally: BIT data type, event/error handling and some array features) by moving them to Appendix F, Removed Extensions. Appendix F preserves the committee's work and suggests carefully thought-out approaches to implementors who wish to provide these capabilities. In the 77 Standard, Hollerith constants were described in such an appendix.

The June 87 Standard we are reviewing received several no votes, including IBM, DEC and Boeing Computer Services, all for essentially different reasons. Both DEC and IBM voted no on the 77 Standard.

## Accomplishments

FORTTRAN 77 source programs are fully compatible with Fortran 8x. 8x defines array processing, recursion, structured programming constructs and derived types as well as many new intrinsic functions. An important approach of the 8x standard is the one that preserves compiled-code compatibility with Fortran 77. Careful implementors can preserve FORTRAN 77 compiled libraries and program units; a programmer could continue to maintain old program units in the existing FORTRAN 77 and code new subprograms in 8x. This observation is made without a detailed study of input/output and other fine points. Also, installations that take advantage of this would have to maintain two runtime libraries.

P. 472

## Review

There are four types of criticisms in what follows: wrong in FORTRAN 77 and should be fixed in 8x, missing in 8x, in 8x but I propose a better way, and in 8x but shouldn't be.

### Wrong in 77

There were mistakes introduced in FORTRAN 77 which, if are not changed in 8x, may never be corrected. These mistakes and other 77 features which are no longer needed or are recognized as poor programming practices are identified in 8x Standard Appendix B, Decremental Features. Appendicies are not part of the standard, they are included for information purposes only. To maintain compatability with 77, the committee hasn't removed these features which have been in use for as long as programmers had a 77 implementation available. There is precedent for removing features without warning: FORTRAN 77 eliminated the extended range of the DC. The features I recommend removing are:

- DO index can be REAL,
- ENTRY statements in functions, and
- GO TO a labeled END IF from outside the BLOCK IF construct.

Strict adherence to the ASCII collating sequence has never been a part of the standard. If we have national standards for both character sets and languages, isn't it sufficient to state that the character set is ASCII? Intrinsic routines are defined in 77, i.e., LGT, LGE, ..., to handle ASCII comparisions in a portable way. These intrinsic functions are superfluous and should be deleted. 8x has two more: ACHAR and IACHAR. A standard processor should use the ASCII character set and any internal complications should be invisible to the programmer.

Although the INTRINSIC statement is not listed in Appendix B, Decremental Features, this seldom used feature should be eliminated. Users can write their own external function that does the same thing without the kludge that most, if not all, compilers use to provide this capability. Example:

<pre>* 77 &amp; 8x INTRINSIC SIN CALL SUB(SIN) END</pre>	<pre>* Lahey's 88 EXTERNAL SINEMINE CALL SUB(SINEMINE) END FUNCTION SINEMINE(X) SINEMINE = SIN(X) END</pre>
--	---

### Missing

The INCLUDE statement has not been standardized, partly because 8x defines a more general solution, MODULE and USE. I claim that every 8x implementation, since most 77's have it, will have the INCLUDE statement to preserve existing code. Isn't one of the purposes of standardization for the committee to look at existing implementations and standardize them?

8x needs more minimums of maximums for language implementation parameters. One that exists is the number of dimensions for an array is seven. Currently, a programmer has to survey all the systems a program is to run on and then program to the minimums he finds. Lack of these limits reduces portability and even results in poor implementations. The three minimums of maximums that I see missing are:

- the length of CHARACTER entities;
- the number of internal procedures allowed in a program unit;
- the number of derived types allowed in a program unit.

A feature that would increase portability is conditional compilation. Since its first standardization, Fortran has recognized the need to experiment with features by stating: "A standard-conforming processor may allow additional forms and relationships provided that such additions do not conflict with the standard forms and relationships." Another quote: "The purpose of this standard is to promote portability, ..." Many application programmers need to have their programs run on more than one system. Conditional compilation would permit one source file to run on all 8x-conforming systems.

## Different

8x introduces a free-form syntax for source files. The features are:

- upper and lower case, compiler translates nonCHARACTER constants to upper case
- multiple statements per line, delimiter is ';'
- trailing comments on each line, delimiter is '!'
- continued statements are indicated by '&' being the last character in the line that is not part of the trailing comment

Example:

```

10      a=1; b=a; c=0           ! Initialization
      IF ( a .EQ. b ) &       ! Loop
      THEN                     ! Main part
      ...

```

Continuation should be denoted by '&' being the first nonblank character on the continued line. The following example leads to a contradiction because one delimiter ends a statement, the other starts a statement.

```

10      FORMAT(2X13F10.2,'&
      ! Is the above blank line a part of the FORMAT &
      ! Is this a comment or continuation of the FORMAT? &
      ! where is the first source character?')

```

It is more "Fortranish," i.e., FORTRAN 77 continues statements by column 6 (the first nonblank character the reader sees) being a nonblank, nonzero character.

In 8x, functions can return arrays or defined-type structures in addition to the single value currently available in FORTRAN 77. The following code defines a real function that returns an array of 500 values:

* 8x	* Lahey's 88
TYPE many	REAL f
REAL X(500)	EXTERNAL f(500)
END TYPE	
TYPE (many) f	

Declaring dimensions in EXTERNAL context denotes it is a function that returns an array value - hardly any new syntax and a move towards putting all external references in an EXTERNAL declaration. There are so many new INTRINSIC functions that hits on existing user-defined procedure names are likely. As recommended in 77, programmers can solve this problem by using the EXTERNAL declaration in all program units.



Derived-type declaration syntax should be simplified.

* 8x	* Lahey's 88
TYPE (many) f	many f

An interface block defines subprogram interfaces and some 8x features require its use. Restrict interface blocks to MODULEs (or INCLUDE files) and reduce the likelihood of different declarations of the same interface block in different program units.

### Shouldn't be in

For a language that is already large, another syntax to declare attributes has been introduced and is therefore undesirable; e.g.,

**REAL, SAVE, ARRAY(10,10) :: X, Y**

Let's stick with the historical form of declarations, even if it is not ideal. Furthermore, if the committee introduces syntax that duplicates existing syntax and the Decremental Features Appendix exists, shouldn't the "old" form be in that appendix? Another introduced duplication of syntax that I object to is ">=" for .GE., "=" for .EQ., etc. Would 8x be approved if attributes and relational operators were the only changes to the 77 standard?

The CONTAINS statement delimits a procedure from its internal procedures - it is not needed. Fortran, unlike COBOL, has never delimited the transition from declaration to executable, why begin now?

Move RANGE and IDENTIFY, part of array operations, to Appendix F, Removed Extensions, and return vector-valued subscripts. Two companies that have implemented array processing use vector-valued subscripts. Further, FORTRAN 77 allows an array element to subscript an array so the extension seems natural. Example:

* 77	* Lahey's 88
INTEGER I(10), J(20)	INTEGER I(10), j(20)
DO 10 N=1,20,2	DO n=1,20,2
10 I(N)=N	· I(n)=n
DO 20 N=1,10	END DO
J(I(N))=	j(I)=...

### Standardization

I have the following criticisms of the standardization process:

- The only PR that I have seen stating that 8x was available for public review beginning October 23 appeared in *Fortran Forum*, which I received 17 December 87.
- Eleven years is too long between standards.
- A publicly reviewed charter is needed.

8x has taken more than eight years to formalize, I estimate more than 35 people-years, resulting in an inertia and a sacredness that makes it difficult to change. I doubt if the Fortran community can digest and comment on 8x in a four month period. There is also the consequence of making changes to 8x: features we have been looking forward to for eleven years will be delayed by the implementation of these changes.

Fortran has a flavor: statement labels, relational operators delimited by ':', column six for continuation, blanks are ignored, no reserved words, DATA, ... which distinguishes it from other languages. In addition, there is all that Fortran code that is successfully running.

P. 475

Before the next standard is started, the committee and the Fortran programming community need to agree on a few basics: What is Fortran? What is subject to change? What isn't? Under what conditions does it get changed? This approach, the creation of a charter, would clearly bless or condemn ideas such as the removal of EQUIVALENCE and COMMON statements and the introduction of alternate syntax for capabilities already defined.

If the charter was in place, the three major negative votes would not have occurred; or if they occurred, they would be accepted or dismissed on the basis of that charter. The Fortran community would not be wondering how much more there is to do before 8x is finished.

### Perspective

I programmed the compiler of the first implementation of FORTRAN 77 on the Honeywell 6600 under the DTSS Operating System. Mr. Robert Wegsten was responsible for the runtime package. We passed acceptance tests during November '75, and we changed the language system as the Standard changed. Customers include General Motors, Union Carbide, Hughes Aircraft and Citibank.

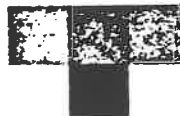
Under contract to GE Timesharing, I wrote the compiler for their System FIV which has local procedures, dynamic arrays, STRING data type, and free-format source files.

In February '83, Mr. Bruce Bush and I began to port the DTSS system to the PC and sold the first copy of F77L during September '84. Since then we have made two major releases, received favorable reviews including Editor's Choice of PC Magazine, and sold more than 5,000 copies. F77L is the leading implementor of 8x features: recursion, 31-character names with the underscore character, lower case letters, the quote character as a synonym for apostrophe in delimiting CHARACTER constants, and the IMPLICIT NONE and NAMELIST statements.

# Thinking Machines Corporation

72

157- MvAB-1



ANSI Technical Committee X3J3  
c/o X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N. W., Suite 500  
Washington, D. C. 20001-2178

Dear Sirs:

Under the Draft Proposed Revised ANS Standard for Programming Language Fortran, published for public review from October 23, 1987, to February 23, 1988, MAXVAL (section 13.12.60) has a very unfortunate property:

MAXVAL (ARRAY) is required to return the value -HUGE (ARRAY) in the case that ARRAY has size zero, and similarly for the cases that a MASK has no true elements (section 13.12.60).

Disclaimer: This letter does not necessarily reflect the official position of Thinking Machines Corporation, which is represented by Mr. Michael Berry. I am also an employee of Thinking Machines Corporation, but I present my arguments from what I hope is the viewpoint of a disinterested academic. (For your information, I was formerly a member of X3J11 (C language) and am presently vice-chairman of X3J13 (LISP language).) Please note that Mr. Berry has stated the official position of Thinking Machines Corporation on this matter in a separate communication.

Many computers use two's-complement representation for integers. While the model for integer data (section 13.6.1) requires the model for integers to be symmetric about zero, the draft standard elsewhere acknowledges that in practice it may not be so (section 4.3.1.1): "The set of values for the integer type... includes all of the integer values from some processor-dependent minimum negative value to some processor-dependent maximum positive value."

This discrepancy between the model and the reality does not have many bad consequences, but the case of MAXVAL cannot be ignored.



(2)

All other reduction intrinsics, when applied to an empty set of elements, produce an identity value for the operation, but this requirement may prevent MAXVAL from returning the identity value for a given implementation.

This is a nasty irregularity that prevents the programmer from relying on certain useful algebraic identities. For example, the programmer may assume that  $SUM(A(I:J))+SUM(A(J+1,K))$  is equal to  $SUM(A(I:K))$  for any J in the range I:K; but he may not assume that  $MAX(MAXVAL(A(I:J)),MAXVAL(A(J+1,K)))$  is equal to  $MAXVAL(A(I:K))$ , for the identity may fail in the boundary cases where J equals I or K.

The same difficulty can occur, in principle, for real data as well. Thanks to the increasing acceptance of the IEEE standard for floating-point formats and operations, however, asymmetry in the actual representations of real data is admittedly less of a problem nowadays than it used to be. (The DECsystem-20 computers, now no longer manufactured, have an asymmetric representation, for example.)

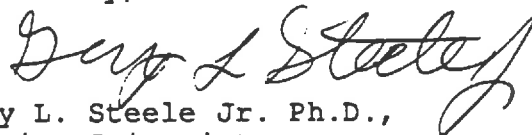
There is more than one way to remedy this difficulty.

(1) The simplest method is to modify the requirement on MAXVAL to return the minimum negative integer of the array's element type, rather than -HUGE(ARRAY), when the set of array elements is empty. (I do not exclude the possibility that other parts of the language may require similar adjustment, but I have not noticed any.)

(2) A more elaborate repair would explicitly recognize the possibility of asymmetric number ranges, introduce a new inquiry intrinsic (called, say, MHUGE) to find out the minimum negative value, and systematically use this new intrinsic in all relevant places in the standard.

I myself have no special desire to increase the number of intrinsics in the language. I do, however, argue for the importance of preserving algebraic identities as a tool for understanding and reasoning about programs, both as they are being written and as they are later maintained.

Sincerely,



Guy L. Steele Jr. Ph.D.,  
Senior Scientist

# Thinking Machines Corporation

ANSI Technical Committee X3J3  
c/o X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N. W., Suite 500  
Washington, D. C. 20001-2178

Dear Sirs:

I refer to section 7.2.3 and table 7.6 of the Draft Proposed Revised ANS Standard for Programming Language Fortran, published for public review from October 23, 1987, to February 23, 1988. I wish to suggest that the symbol "<>" newly introduced as a synonym for ".NE." is ill-chosen and should be either eliminated or replaced by the symbol "/=".

Disclaimer: This letter does not necessarily reflect the official position of Thinking Machines Corporation, which is represented by Mr. Michael Berry. I am also an employee of Thinking Machines Corporation, but I present my arguments from what I hope is the viewpoint of a disinterested academic. (For your information, I was formerly a member of X3J11 (C language) and am presently vice-chairman of X3J13 (LISP language).)

The symbol "<>" suggests the interpretation "less than or greater than". This may be regarded as equivalent to "not equal" only for domains where the law of trichotomy holds.

Trichotomy does not hold for complex values. It is sensible to say that two complex numbers are not equal, but it is nonsensical to speak of one complex number as being "less than or greater than" another.


Moreover, trichotomy does not hold for floating-point numbers as defined by ANSI/IEEE standard 754-1985. In that model the notions of "not equal" and "less than or greater than" are distinct and independently useful. Table 4 of that standard suggests the operators "?<>" and ".NE." for "not equal", and "<>" and ".LG." for "less than or greater than". It would not be good for Fortran 8X to come into conflict with this other standard.

I realize that there is some precedent in the programming language community for the notational abuse of "<>"; this stems primarily from the programming language Pascal. Now Pascal is admirable in many other respects and introduced a number of useful notions and notations. However, we must note that while Pascal does not have complex numbers, it contains the same egregious notational blunder with respect to sets: saying that two sets are not equal is not the same as saying that one must be a proper subset of the other. (Pascal does distance itself one dainty step from this difficulty by omitting to give "<" and ">" their obvious definitions on sets as tests for proper containment.)

The Ada programming language, which borrows heavily from Pascal in other respects, broke with tradition by substituting the symbol "/=" for "<>". The symbol "/=" is about as close the the traditional mathematical inequality symbol as "<=" is to the traditional less-or-equal symbol. The symbol "/=" was later adopted by Common Lisp (now being standardized by ANSI X3J13) for its inequality test. Note too that the C language (now being standardized by ANSI X3J11), from which we might speculate Fortran 8x has borrowed the symbol "==", uses the similar notation "!=" for "not equal".

I recommend that Fortran 8x, if it must adopt such symbols as "<=" at all, abandon "<>" and adopt "/=" as well.

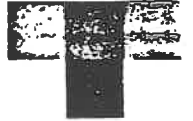
Sincerely,



Guy L. Steele Jr. Ph.D.,  
Senior Scientist

# Thinking Machines Corporation

ANSI Technical Committee X3J3  
c/o X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N. W., Suite 500  
Washington, D. C. 20001-2178



Dear Sirs:

I wish to argue for reinstatement of the FORALL statement, as described in Appendix F of the Draft Proposed Revised ANS Standard for Programming Language Fortran, published for public review from October 23, 1987, to February 23, 1988.

Disclaimer: This letter does not necessarily reflect the official position of Thinking Machines Corporation, which is represented by Mr. Michael Berry. I am also an employee of Thinking Machines Corporation, but I present my arguments from what I hope is the viewpoint of a disinterested academic. (For your information, I was formerly a member of X3J11 (C language) and am presently vice-chairman of X3J13 (LISP language).) Please note that Mr. Berry has stated the official position of Thinking Machines Corporation on this matter in a separate communication.

I have two main points: that FORALL provides a kind of abstraction not accommodated elsewhere in the language; and that many users will be more comfortable with FORALL than with certain of the more esoteric intrinsic functions. I also argue that compilation issues should not be a large problem.

(1) The FORALL statement allows one to give names to ranges. This promotes program readability and maintainability because a range is easily defined in one place and then referred to by name.

Consider a common sort of calculation, where every interior element of an M-by-N array A is to be updated as a function of its four neighbors (assume F to be elemental). It can be expressed through array sections:

$$A(2:M-1, 2:N-1) = F(A(2:M-1, 2:N-1), A(1:M-2, 2:N-1), A(3:M, 2:N-1), & \\ A(2:M-1, 1:N-2), A(2:M-1, 3:N))$$

But it is difficult to see the underlying pattern in the subscript expressions.

One might attempt to use the CSHIFT or EOSHIFT intrinsic:

```
A = F(A,CSHIFT(A,1,1),CSHIFT(A,1,-1),CSHIFT(A,2,1),CSHIFT(A,2,-1))
```

Unfortunately this does not mean quite the same thing, as it will update boundary elements as well as interior elements; this can be repaired by placing the assignment within a WHERE statement, but setting up the necessary mask expression is not straightforward. By comparison, the FORALL statement provides a clear solution:

```
FORALL(I=2:M-1,J=2:N-1)A(I,J)=F(A(I,J),A(I-1,J),A(I+1,J),A(I,J-1),A(I,J+1))
```

The pattern is easier to see than with the array sections all written out, and for a very good reason: one has given a name to sections such as 2:M-1 and 2:N-1, and then used the name everywhere. Moreover, the relationship between the ranges 2:N-1 and 3:N is easier to see in the form J and J+1, because there are fewer expressions to relate.

Suppose now that the calculation must be modified so as to leave a border of width 2 unchanged, rather than width 1. With explicit sections the change must be made in many places:

```
A(3:M-2,3:N-2) = F(A(3:M-2,3:N-2),A(2:M-3,3:N-2),A(3:M-2,3:N-2), &
  A(3:M-2,2:N-3),A(3:M-2,4:N-1))
```

In fact, I have intentionally made one error in this last example. (Hint: it is an error of problem intent, rather than of language semantics, and therefore an error that a compiler cannot catch.) See how difficult it is to spot!

With a FORALL statement many fewer places require change:

```
FORALL(I=3:M-2,J=3:N-2)
A(I,J)=F(A(I,J),A(I-1,J),A(I,J),A(I,J-1),A(I,J+1))
```

I have introduced the identical error into this FORALL statement. Is the error easier to spot?

Nowhere else in the language is it possible to give a name to a range. (Actually, if vector-valued subscripts were to be reinstated, one might declare I and J to be vectors and then write:

```
I=[3:M-2]
J=[3:N-2]
A(I,J)=F(A(I,J),A(I-1,J),A(I+1,J),A(I,J-1),A(I,J+1))
```

I suspect that compilers would have a more difficult time producing good code for that than for the FORALL statement.)

(2) FORALL allows an array notation that is closer in style both to previous Fortran usage (DO loops) and, which is perhaps more important, to standard mathematical notation than the intrinsic array functions that have been introduced into Fortran 8x. Now I happen to be a proficient APL programmer, and as a result I feel quite comfortable with the new array intrinsics. I certainly do not recommend removing them. However, many numerical programmers accustomed to Fortran 77 have expressed to me an unwillingness to adopt this unfamiliar style of programming. They prefer a notation with explicit indices.

Many of the array intrinsics, both included and removed, can also be expressed through FORALL. For example,  $A = \text{SPREAD}(V, \text{DIM}=2, \text{NCOPIES}=N)$  may be rendered as

```
FORALL(I=1:M, J=1:N) A(I, J) = V(I)
```

Similarly,  $A = \text{REPLICATE}(B, \text{DIM}=2, \text{NCOPIES}=K)$  may be rendered as

```
FORALL(I=1:M, J=1:N*K) B(I, J) = A(I, 1 + MOD(J-1, N))
```

Vector-valued subscripts may also be expressed:  $A(W) = B(V) + 3$  is simply

```
FORALL(I=1:N) A(W(I)) = B(V(I)) + 3
```

A converse relationship between FORALL and vector-valued subscripts has already been noted above.

Indeed, if FORALL is reinstated, it could provide a concise and precise notation for the definition within the standard of many array intrinsics. For example, in section 13.12.94 the paragraph "Result Value" for  $\text{TRANSPOSE}(\text{MATRIX})$  may be rendered simply as

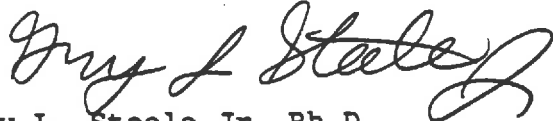
```
FORALL(i=1:n, j=1:m) result(i, j) = MATRIX(j, i)
```

This is remarkably close to the actual statement in English, which testifies to the naturalness of FORALL to programmers accustomed to standard mathematical notation.

The main advantage of FORALL over nested DO loops is in expressing the intent to perform the assignments in parallel rather than serially. On a serial machine, FORALL may be somewhat more difficult to implement because of the need to keep results in a temporary place (unless the compiler can prove it unnecessary). On a parallel architecture, DO loops may be somewhat more difficult to handle because of the need to communicate results from one iteration to later iterations (unless the compiler can prove it unnecessary). This is tit for tat. I think the programmer should

have the choice to express his intent as precisely as possible. Whichever of the two constructs is the more difficult for a particular architecture, the compiler technology needed to reduce it when possible to the other, simpler case is similar, and well understood today. In fact, transforming FORALL into nested DO loops is considerably easier than the reverse, because of the restriction that the body of a FORALL must be a single assignment statement. Any perceived need for exotic compiler technology is therefore incorrect and should not bar the adoption of FORALL.

Sincerely,



Guy L. Steele Jr. Ph.D.,  
Senior Scientist

ANSI Technical Committee X3J3  
c/o X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N. W., Suite 500  
Washington, D. C. 20001-2178

Dear Sirs:

I wish to argue for reinstatement of vector-valued subscripts, as described in Appendix F of the Draft Proposed Revised ANS Standard for Programming Language Fortran, published for public review from October 23, 1987, to February 23, 1988.

Disclaimer: This letter does not necessarily reflect the official position of Thinking Machines Corporation, which is represented by Mr. Michael Berry. I am also an employee of Thinking Machines Corporation, but I present my arguments from what I hope is the viewpoint of a disinterested academic. (For your information, I was formerly a member of X3J11 (C language) and am presently vice-chairman of X3J13 (LISP language).) Please note that Mr. Berry has stated the official position of Thinking Machines Corporation on this matter in a separate communication.

(1) On the merit of the feature itself: Vector-valued subscripts represent a concept that cannot be expressed nearly so easily or succinctly in any other way: elemental application of the "subscripting" operation. In most cases one must resort to writing a DO loop, with all its connotations of serial processing.

In some respects an array can be regarded as a mathematical function with an implementation peculiarly optimized to trade space for speed and to allow run-time modification, but with a restriction to integer arguments. When I need to use Fibonacci numbers, for example, I write FIBONACCI(N); whether FIBONACCI is a function or an array is an implementation tradeoff. Now let N be an array of integers rather than a scalar integer. The arbitrary decision of the current standard that, while most primitive



(2)

arithmetic operations may be used elementally, subscripting calculations may not, introduces an annoying language irregularity that complicates programming unnecessarily.

Moreover, the capabilities of hardware, present or future, are not a counterargument. Serial architectures should have no more difficulty with this feature than with other array sections; the main problem is the handling of array temporaries, which is provoked by many other features in the language. Furthermore, there are now many commercially available computer architectures, of both the parallel and vector varieties, that have hardware capable of supporting this operation with special efficiency.

As an aside, I would like to suggest that if vector-valued subscripts are reinstated then intrinsics called, say, DBOUNDS and EBOUNDS would be of great utility. The meaning of DBOUNDS(A,DIM=J) would be simply [DLBOUND(A,DIM=J):DUBOUND(A,DIM=J)], and similarly for EBOUNDS. I anticipate that subscripts of the form EBOUNDS(A,1) to stand for ELBOUND(A,1):EUBOUND(A,1) would be fairly common.

Another observation is that the FORALL statement includes subscript-like expressions called forall-triplet-specs (section F.2.3.1). Indeed, as I point out in a companion letter, FORALL may be viewed for some purposes as merely a means of naming subscript expressions. If vector-valued subscripts and the FORALL statement are both reinstated, a useful extension would be to permit the former within the latter: that is, allow a forall-triplet-spec to be a single vector-valued expression. In conjunction with EBOUNDS, a statement of the form

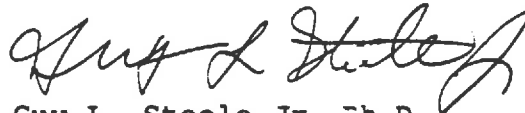
FORALL(I=EBOUNDS(A,1),J=EBOUNDS(A,2)) A(I,J)=...  
would be extremely convenient and frequently used. (End of aside.)

(2) In rebuttal to the committee's response to Mr. Willi Schonauer (page 143 of the submittal document): The committee's response argues primarily that the vector-valued subscripts would introduce an irregularity in the language because of certain restrictions that seem desirable when vector-valued subscripts are used on the left-hand side of an assignment statement.

I agree that such an irregularity occurs, and that the restriction is indeed desirable, but I argue that the irregularity is wrongly attributed. It is the fault of the assignment statement, not of vector-valued subscripts. One is not permitted to write  $\text{SQRT}(X)=5$ , even though that might plausibly modify  $X$  to have the value 25. One is not permitted to write  $\text{CSHIFT}(A,\text{SHIFT}=1)=B$  even though that might plausibly mean the same as  $A=\text{CSHIFT}(B,\text{SHIFT}=-1)$ . The assignment statement inherently treats its left-hand side in a peculiar manner that requires all manner of restrictions. Some of these restrictions are matters of semantics, some of readability, and some of implementation convenience.

Now observe that strides in triplet subscripts are required to be nonzero (section 6.2.4.4). This restriction is quite arbitrary. Offhand I can imagine two possible motivations for this restriction (and if there are others, please let me know). First, certain hardware may not support zero-length strides effectively. Against this I argue that certain hardware does not support zero-trip DO loops effectively but there they are (sections 8.1.4.4.2 and 8.1.4.5). Second, zero-length strides on the left-hand side of an assignment produce much the same aliasing difficulties as vector-valued subscripts. Zero-length strides produce no semantic difficulties on the right-hand side. If left-hand-side difficulties were the primary motivation for barring zero strides, then it seems curious to include one feature with an irregular restriction but exclude another, very similar, feature on the grounds that a very similar restriction would be required.

Sincerely,



Guy L. Steele Jr. Ph.D.,  
Senior Scientist

# **Thinking Machines Public Commentary on Draft Proposed Standard X3.9-198X Programming Language Fortran**

January 29, 1988

## **1 Scope of Commentary**

Thinking Machines Corporation is currently engaged in the implementation of a Fortran Compiler for the Connection Machine® data parallel computing system. The Fortran implementation is based on Fortran 77 and the array features of Fortran 8x. In order to take full advantage of the Connection Machine architecture, a programming language must address its two most salient features:

**Elemental operations** — Each array element is assigned to a separate Connection Machine virtual processor, (which may be physical or virtual). Arrays of like shape reside in the same set of processors, allowing elemental operations to be done in constant time.

**Interprocessor Communication** — Any processor may access the memory of any other processor and all processors may communicate in parallel. Operations are most efficient among neighbors in the conceptual n-dimensional lattice of processors in which the arrays are embedded. There are specialized instructions for rapidly propagating data along new or existing array axes.

In designing Connection Machine Fortran, we have worked from the draft Fortran 8x standard. For the near term, we have adopted only those portions of the proposed standard that directly relate to data parallel array-oriented programming. These features all come under the heading "array extensions." Although we consider many of the other extensions in the proposed standard (e.g., user-defined data types, module interfaces) to be useful ideas that we will eventually adopt if they become standard, we would be content with a standard that included nothing beyond the array extensions to Fortran 77.

## 2 General Comments

We heartily approve of making all Fortran 77 intrinsic functions elemental. Not only are array valued expressions more concise, direct, and readable when unencumbered by indices and loop counters, the 8X style array expressions directly express the desired result, rather than suggesting an implementation method. On data parallel architectures compilers will no longer have to undo "scalarization" which did not exist in the original problem, but is present in the code due to a deficiency in the expressive power of the language. At the same time, compilers for scalar architectures will have no difficulty translating the elemental expressions into iterative ones for execution.

We question the removal of the array features listed in appendix F as "removed extensions." These features provide important functionality that is hard to express without them. We ask that the committee reexamine each of these removed extensions in light of the arguments for their inclusion that we put forward later in this document.

We recognize that major alterations to the language definition provided by the proposed draft standard are best left for a future standardization or language design effort. There are, however, a number of minor changes which could be made to the definitions of transformational intrinsics which already exist in the draft proposed standard which would increase their generality. For the most part, these changes are simply relaxations of unnecessary restrictions.

The clarity of the document would be improved by the provision of better examples. In many cases, the examples given currently illustrate only the simplest and easiest to understand uses of a feature. For instance, the example given for the FORALL statement does not illustrate its power. Similarly, the example for RESHAPE does not show the effect of the ORDER optional argument.

Another problem with the examples is that the relationship between the display form used for arrays throughout the document and the defined array element order is not made clear. The descriptions of CSHIFT, RESHAPE, COUNT, EOSHIFT, MAXLOC, MINLOC, and SPREAD, among others, include arrays displayed according to the usual conventions of linear algebra. That is, an array with shape  $(/3,4/)$  is drawn with three rows and four columns. This presentation seems natural to mathematicians and to users of other programming languages, but it is not the same order one would normally get when supplying a whole array to a Fortran WRITE statement.

It seems natural to test a new implementation of Fortran against examples in the standard. If an array appears as an item in an output statement, the values will be printed in array element order from left to right across the page. The resulting printout will not match the examples in the standard.

### 3 Comments on Individual Extensions and Definitions

These comments are presented in the order in which the features to which they pertain are defined in the draft proposed standard. The relevant section numbers appear in brackets at the end of each subheading.

#### 3.1 The RANGE attribute [5.1.2.8, 5.2.8, 6.2.5]

The RANGE attribute is unnecessary and, we believe, ill-considered. A programmer may refer to an array section as if it were a whole array by using the IDENTIFY statement to alias the section to a new name. This method is preferable to the use of ranges because it avoids the confusion caused by having the same name refer to different subarrays at different places in the source program. The use of RANGE seems likely to lead to errors in reading and writing programs. Getting rid of the RANGE attribute would also allow for the elimination of the ESHAPE, ESIZE, ELBOUND, and EUBOUND intrinsics. The initial D's could then be dropped from the names of the DSHAPE, DSIZE, DLBOUND, and DUBOUND.

#### 3.2 Storage Association of Data Objects [5.5, 5.5.1, 5.5.2]

Language features that make a particular computer architecture explicitly visible at the source program level should be labeled as *obsolescent* or *deprecated*. As data parallel and other distributed architectures become increasingly common, the notion of "storage order" will become meaningless on an increasingly large set of machines. On the Connection Machine System, for instance, each element of an array is stored in a separate processor. The processors containing an array of rank N are arranged in a lattice also having rank N. Two arrays having the same number of elements, but different shapes, are stored in different sets of processors. A linear memory model is assumed by the EQUIVALENCE statement and by uses of the COMMON statement that change the shapes of arrays or cross array boundaries. This linear model is difficult to support efficiently on newer architectures, and in our view is not worth maintaining.

#### 3.3 Logical operators [7.1.2]

Given the wide use of overloading elsewhere in the draft proposed standard, there is no reason not to allow .EQ., .NE., = =, and < > to take arguments of type logical. This would make .EQV. and .NEQV. redundant.

**3.4 HUGE [13.9.6, 13.12.43]**

The numeric inquiry functions provided are not sufficient to determine the value of the negative number of greatest magnitude for a particular implementation. The numeric model needs to include distinct values for the positive number limit and the negative number limit. These quantities will have different magnitudes on machines where, for example, a two's-complement representation is used for integers.

**3.5 DOTPRODUCT [13.12.27]**

This intrinsic would be redundant if the definition of MATMUL were to be changed as suggested in section 3.7 below. The *name* DOTPRODUCT would be more appropriate than MATMUL for the combined intrinsic.

**3.6 INDEX [13.12.46]**

This facility would be useful for vectors of any type, not just character strings. In fact, it would be useful to be able to determine the location of a subarray of any shape in a parent array. Furthermore, the INDEX intrinsic should have an optional DIMENSION argument so that one could find the first occurrence of a certain string in each row or each column of a matrix.

**3.7 MATMUL [13.12.56]**

The restrictions on the ranks of the arguments should be relaxed. If both arguments are vectors, the result will be the scalar which would have been returned by DOTPRODUCT. In general, the arguments should be allowed to have any rank so long as the length of the last axis of the first argument is the same as the length of the first axis of the second argument.

**3.8 MAXLOC, MINLOC [13.12.59, 13.12.64]**

The following suggestions are not mutually exclusive. Any one alone would be sufficient to address our concern.

### 3.8.1 Replace with a general INDEX

While MAXLOC and MINLOC are useful, many similar functions are equally useful, but not provided. (e.g., location of the first .TRUE. in a logical vector). If the domain of INDEX were expanded as suggested in 3.6 above, these special case intrinsics would not be necessary since they could be replaced by expressions of the form

INDEX(V, MAXVAL(V))

### 3.8.2 Provide LOCATION and allow optional DIMENSION arguments

LOCATION(ARRAY, TARGET, MASK, DIMENSION, BACK)

Optional Arguments: MASK, DIMENSION, BACK

**Description:** Determine the location of an element of ARRAY having the same value as TARGET.

**Kind:** Transformational function.

**Arguments:**

**ARRAY:** may be any type.

**TARGET:** a scalar having the same type as ARRAY

**MASK:** must be type logical and conformable with ARRAY.

**DIMENSION:** An integer scalar in the range (/ 1:SHAPE(SHAPE(ARRAY)) /)

**BACK:** A scalar of type logical

**Result Value:**

If DIMENSION is absent, the result is as for MAXLOC(ARRAY, MASK) with TARGET substituted for "maximum value" throughout the description. If the DIMENSION argument is absent, the BACK argument has no effect.

If the DIMENSION argument is present, the result is an array of rank one less than the rank of ARRAY. The values returned are the first locations of TARGET along the axis specified by DIMENSION.

The BACK argument does exactly what the BACK argument to INDEX does. If it is absent or .FALSE. the values returned are the first locations of TARGET when traveling in the direction of increasing indices in DIMENSION. Wherever TARGET is not found, a 0 appears in the result. When BACK is .TRUE, the search is in the

reverse direction and wherever TARGET is not found, the value in the resulting array is 1 more than the extent of DIMENSION.

### **3.8.3 An Even More General LOCATION function**

The definition of 3.8.2 could be generalized to allow TARGET to be an array of rank one less than the rank of ARRAY and shape conformable to the result shape. Each element of TARGET would be sought in the one-dimensional subarrays along axis DIMENSION. This would allow one to, for example, find the location of the first 1 in the first row, the first 2 in the second row, the first 3 in the third row, and so on.

### **3.9 MAXVAL [13.12.60]**

The current draft proposed standard states that if MASK does not contain any true elements, the result is -HUGE(ARRAY). The result should instead be the negative number limit. See comment 3.4.

### **3.10 TRANSPOSE [13.12.94]**

The restriction of the argument rank to two should be removed. The function should allow one to transpose an array of rank N by simply reversing the order of the shape vector. For example, if M has shape (/3,4,5/) then TRANSPOSE(M) has shape (/5,4,3/).

## **4 Removed Extensions Needed for Data Parallel Architectures**

Although we would like to see all of the removed array extensions restored, two of them, vector valued subscripts and the FORALL statement play particularly important roles on data parallel systems.

### **4.1 Vector Valued Subscripts**

Vector valued subscripts allow the explicit use of interprocessor communication in arbitrary patterns on a data parallel computer. Array sections support rectilinear



patterns of communication only. Other patterns, such as trees, can be achieved only through vector-valued subscripts.

These kinds of communication are fundamental to algorithms designed to take advantage of an architecture consisting of an array of processors. Fortran must have a way of expressing general parallel data motion or it cannot be used for this style of programming. On serial machines the constructs can be straightforwardly broken down into a series of scalar assignments.

#### **4.2 FORALL statement**

On a serial machine, the FORALL statement may be regarded as another form of loop control which should not cause much difficulty for users or compiler writers. On a data parallel computer, the FORALL statement may be seen as a request from the programmer to have the assignment expressions executed in parallel. On the Connection Machine, this is accomplished by using SPREAD (there is a Connection Machine instruction analogous to the transformational intrinsic of the same name) to add a new dimension for each *subscript-name* in the *forall-triplet-spec*. The *forall-assignment* expression is then executed in parallel on the temporary which represents all combinations of the *forall-triplet-specs*.

### **5 Other Removed Extensions**

Experience with APL suggests that when arrays are first class objects in a language, the structure of arrays is as important as the contents and all kinds of transformational functions are used frequently. All of the removed array intrinsics except RANK are transformational functions. Whether or not the set of transformational functions in appendix F represent the best possible collection, each one is useful individually, and together they provide a good vocabulary for array formation and data movement.

We do not have specific comments on DIAGONAL, PROJECT, FIRSTLOC, and LASTLOC, but we would like the committee to reconsider their removal in light of the need to make FORTRAN a more expressive and flexible array programming language.

#### **5.1 REPLICATE**

If one of replicate or SPREAD were to be removed, SPREAD would be the more likely candidate for removal because it is more natural to define SPREAD in terms of a

**RESHAPE** to insert a 1 in the shape vector of the argument followed by a **REPLICATE** along the new axis than to define **REPLICATE** in terms of **SPREAD** and **RESHAPE**. The latter requires adding a new dimension just to remove it. In any event, we feel that these functions are sufficiently different from one another that both should be kept.

## **5.2 RANK**

This is the only array intrinsic from appendix F whose removal does not bother us. We suspect, however, that the expression **SHAPE(SHAPE(A))** will be so common that compilers will special case it.

## **6 Concluding Remarks**

The draft proposed standard represents a monumental effort by many dedicated people. We applaud the committee for its diligence and care. There is a clear and present need for a standard Fortran which is suitable to modern array oriented programming. We feel that the timely publication of such a standard is more important than the details of the definitions it includes. We urge the speedy passage of some version of the Fortran 8x standard. The requirements of new parallel architectures and the demands of the marketplace will cause vendors to continue adding array features to Fortran whether or not there is a Fortran 8x standard to work from. With a standard there is a much better chance that the wheels we are all re-inventing will fit the same car.

To: X3J3  
From: Bill Leonard  
Subject: Interpretation of FORTRAN 77

I have been asked by a user for an interpretation of the FORTRAN 77 Standard concerning Direct Access I/O error conditions. The question is: Does attempting to READ a record of a direct access file that has never been written cause an error condition? Does it matter whether the requested record lies before or after the last record that has been written? In particular, if I have an IOSTAT= specifier on the READ, what value should it receive?

In researching this question, I first referred to Section 12.6 of X3.9-1978, which enumerates the events that can cause an end-of-file condition. Since I interpret this list to be all-inclusive, I infer that attempting to READ a record beyond the last in a direct access file is NOT an end-of-file condition. I then referred to Section 12.9.5, which states that, "On input, an attempt to read a record ... that has not previously been written causes all entities specified by the input list to become undefined." However, it does not say that this is an error condition.

I believe the answer to the question should be: yes, it is an error condition, so the IOSTAT= variable should receive a positive integer value. If there is an ERR= specifier, control should be transferred to the specified label.

As far as I can tell, this ambiguity also exists in S8.104.

Ω

To: X3J3  
From: Miles Ellis  
Subject: British Computer Society Fortran Forum Report

The British Computer Society (in conjunction with the British Standards Institute) held a full-day Fortran Forum in London on November 23rd, 1987. The primary purpose of the Forum was to provide input to the BSI in formulating its response to the ISO review, and in order to ensure that the audience was well-prepared all those attending were issued with a copy of the draft standard which had been reprinted by the BCS for this purpose. All participants were also asked to submit any questions, comments, etc. in advance, or during the morning of the Forum. Over 150 people attended.

The morning was taken up by an overview of Fortran 8x, presented by John Reid and Lawrie Schonfelder, preceded by a short summary of the (ISO) review process given by David Muxworthy, who is chairman of the BSI Fortran Panel.

The afternoon was mainly given over to a panel discussion of the Fortran 8x proposals, based on the questions and/or comments which had been submitted by those attending the Forum. The panel consisted of all the UK X3J3 members (Miles Ellis (chairman), Geoff Millard, John Reid, Lawrie Schonfelder and Alan Wilson). There were a large number of questions - certainly far more than it was possible to discuss - covering all aspects of Fortran and its environment, and these provoked a lively and well informed discussion.

The final part of the afternoon was devoted to discussing and taking a number of straw votes concerning the detailed technical issues that had raised most interest during the earlier part of the Forum, or which David Muxworthy felt to be important, together with two looking at the language as a whole.

The remainder of this paper consists of the results of the straw votes, followed by a summary of the comments and questions received before and during the Forum which was produced by David Muxworthy.

STRAW VOTES TAKEN AT THE CONCLUSION OF THE FORUM

Should there be a BIT data type? 56 yes / 37 no /27 undecided

Should there be a set of BIT intrinsic procedures? 56/18/32

Should there be a pointer facility? 54/38/18

Should blanks be significant in the new source form? 33/38/35

Should there be a facility for input/output of numbers using more than one radix (e.g. base 2, 8, 10 or 16)? 62/5/32

Should there be provision for multi-byte characters? 19/34/53

Should there be a facility for stream I/O? 44/23/33

Should there be a facility for varying length strings? 36/26/37

Is the proposed language too big? 9/86/3

The final vote was concerned with timing and assumed that the inclusion of the various facilities already voted for (BITS, pointers, general radix I/O, stream I/O, varying length strings) would lead to a delay of 2-3 years. The vote was as follows:

Standardise current proposals quickly: 57  
Add extra features with 2-3 year delay: 45  
No opinion: 3

#### COMMENTS AND QUESTIONS SUBMITTED BY PARTICIPANTS

(Note that some of these comments and questions were submitted by people who had clearly read and studied the draft standard, while others were from people whose knowledge of Fortran 8x had largely been gleaned during the morning session of the Forum. No attempt has been made to differentiate between these.)

The section references have typically been added later; in only a few cases were they explicitly specified. Reference 0 indicates 'general' or 'unclassified'; 'F.Miss' indicates a specific topic not in the document. For convenience, only one reference has been given to each entry although in some cases more than one is relevant.

0

Does the team really expect a Fortran 9y standard to be produced eventually?

0

Efficiency of resulting executable code seems to have suffered relative to Fortran 77:

1. Array section handling must be slow and encourage misuse of virtual memory.
2. Array operations are not helpful to machines with vector facilities - they have vectorizing compilers, can handle vector temporaries etc.
3. Bit handling vital for efficient use of space IOR, IAND.
4. Pointer extremely useful for sparse data.

0

I support engineers who use Fortran because it is relatively simple, efficient, and is supported by good libraries. The simplicity and compile time efficiency seem to be lacking in 8X. Current features which cause difficulty have not been improved, why?

0

In view of the long development time of Fortran 8X, what account has been taken of recent concepts and ideas of Software Engineering Techniques. Are there any impacts on Quality Assurance aspects of code written in Fortran 8X?

0

Is there any available indication of the acceptance of a new Fortran standard by the major computer manufacturers and software houses?

0

The facilities seem fairly good for the writing of large programs and in particular, large numerical programs. The semantics seem as clean as can be expected, given Fortran's ancient definition/undefinition rules. However, many aspects of the syntax are made much more confusing by the remains of earlier Fortrans. This is likely to make Fortran 8X compilers have rather unfriendly diagnostics on syntax errors. I would like to see some incompatible changes here.

0

What facilities are there regarding character handling, data-handling and file handling?

0

What is the estimated date for ratification of the next Fortran standard?

0

You've told us F77 will become a subset of 8X, but what about those facilities only available in Fortran 66 (e.g. bit manipulator, equivalence of char/non-char items)?

1.3

Limit: Why not specify that the maximum no. of characters in an identifier, subscripts to all array characters in a source line, etc. be - HUGE(I) (I integer)?

1.3

What is the maximum length of a character string in a portable program?

1.4  
Back to conformance. Surely the standard should make some attempt to define minimum acceptable actions for each error condition (whether compile or tune time). (I do realise this is likely to be difficult, but consider the (VMS/DEC) example of complaining about a syntax error on a line, which returns a random part of the line - not the whole line, and not the offending part. Some statement of style in error detection (e.g. at least give back the whole line, preferably specify the exact error) might help.)

1.4  
Conformancy Issues: In general, if the processor has determined that an error condition exists, then it should be required to report this.

1.4  
Insufficient attention seems to have been paid to the problem of making it difficult to write bug-ridden code. E.g.

1. Possibility of different global data entities and subroutines having the same name.
2. No BLOCK structure - simply to restrict code tangles.

1.4  
Some indication of what can go wrong at run-time would be good. This would help compiler writers who could guarantee that they had checked every case. I see the point of not insisting that run-time errors be detected; but help for those who wish to detect them would be useful.

1.4  
At run time, the ability (presumably user selectable, possibly at compile time) to perform checks on arrays should be required. In particular, in the WHERE statement an example was given and we were told that it did not check that the shape of the source and destination arrays should match. The WHERE statement should be generalised to include more than BOOLEAN choice as,

```
WHERE (cond1)
  action 1
ELSE WHERE (cond 2)
  action 2
ELSE WHERE (cond 3)
  action 3
OTHERWISE
  action 4
END WHERE
```

!ELSE WHERE cannot be  
!used here

is clearer than

p. 500

WHERE (cond 1)  
    action 1  
END WHERE  
WHERE (cond 2)  
    action 2  
END WHERE

!etc - and how to cope with the "OTHERWISE" clause above.

Note that it would presumably be acceptable to require that various conditions not overlap (as for SELECT and the CASE clauses). Would it also be sensible to introduce a "THEN" as syntactic sugar to make things clearer? (I think it would.)

SELECT CASE (cond)  
    CASE (-)  
    -  
    -  
    -

#### 3.1.4

Could the Array Constructor ([]) be avoided? Programs written with square brackets will not be readable in languages which use these as national alphabetic characters. Could then all ASCII characters 64-127 be allowed in names?

#### 3.2.6

Would it be possible to use either names or numbers for statement labels?

#### 3.3.1

Fortran 8X is good, but the new source form is an opportunity not to be missed! Freeze Fortran 77 into the old source form and make blanks significant in the new source form. Make the Fortran 8X enhancements dependent on the new source form (this would allow syntax improvements). Standardize a compiler directive for switching between Fortran 77 and 8X modes on a per routine basis. Take the opportunity to delete unwanted features now. The benefits would be two smaller, more self consistent languages than the current proposal; evolution of Fortran is more rapid yet complete compatibility with existing software is maintained. Transfer to Fortran 8X (and any routine that might be using 8X features) is then immediately obvious from the source form. Verification of Fortran 77 to Fortran 8X 'spaghetti unscramblers' would be welcomed.

#### 3.3.1

Why are significant spaces in free source form so controversial for X3J3?



3.3.1

Why the extension to more than one statement per line when the draft justifies 31ch. names and 132 ch. lines on the grounds of readability and indentation. I can see Fortran follow APL into the one-line syndrome.

4.3.1

Can one define a type MYREAL with certain properties and use it in place of REAL for isolating precision attributes?

4.3.1

Specified Precision is useless if the outcome can be machine dependent

4.3.1.1

Will '8X' contain the equivalent of INTEGER\*1 (or BYTE) as part of the standard? If so, what? If not, why not?

4.3.1.2

In particular, if the precision or exponent range requested for a real cannot be met by the implementation, the processor should either warn the user, or possibly it should abort (if it is not considered feasible for it to perform some correcting action).

4.3.1.2

It appears that precision can be defined at will - but surely, realistically, we still only have a choice of single and double precision because of the dependence on hardware.

4.4.1.1

It appears to be impossible to define a derived type which contains two parameterized precisions. This makes it impossible to define a parameterized SSP or SVD type with an extended precision trace.

5.

Would the Panel comment on the view that it is premature to standardize array features to the extent this has been done in S8 when parallel architectures are still evolving so rapidly.

5.1

Are there any intrinsic functions which require an array to be specifically declared by the "ARRAY" attribute? (See page 5-7)

5.1

Could the use of the "DOUBLE COLON" delimiter (see R419, Page 4-6) be explained more clearly, with examples of where it is, and is not, needed. For example, on line 17 page 4-7, is it optional? Similarly, on line 16 page 4-7, is it mandatory?

5.1

Is `REAL(6,10):: X,Y,Z(7,20),W` legal? If so, why? It isn't very nice! If not, why allow `CHARACTER(20):: K1,K2(10)`?

5.1

It was said during the session that a statement like `REAL,ARRAY(10,5):: X,Y,X(5,3)` declares `X(10,5)` `Y(10,5)` and `Z(5,3)`. Given that the Standards Committee arbitrates on style elsewhere (a good thing), surely this example is confusing, and serves no purpose - it could be done more clearly by declaring `Z` separately.

5.1

Why is the attribute "ARRAY" necessary? For example is there any difference between `REAL W(10,10)` and `REAL,ARRAY(10,10)::W`? If there is no difference, doesn't the "ARRAY" attribute make 8X unnecessarily complicated?

5.1.2.3

Why is there a need for an `INTENT(OUT)`, example P5-7 has `SUBROUTINE TRANSFER(FROM,TO)` with `INTENT(OUT)::TO`. We could equally have `SUBROUTINE TRANSFER(FROM)` without changing anything? I.e. `TO` is simply a local subroutine variable in both cases.

5.1.2.4

Why limit array dimensionality to 7 dimensions? There should be a very good reason for any particular dimension chosen.

5.2.6

Why not lists of lists for initializing multi-dimensional arrays (cf C, ALGOL).

6.1.2

Could the use of the "Percent Sign" notation be explained more, with analogies to coding in Fortran 77, if possible? (See R608, Page 6-2.)

6.1.2

Since the full stop (.) is the standard delimiter in many other languages, could this be used instead of the %, for example 1.2.5.B instead of 1Z2Z5%B. The full stop is not only a de-facto standard, it is also easier to read characters separated by a full stop than those separated by a %. I recognise that there could be some confusion with constructs such as "A..NOT. B", but surely this could be avoided by making words such as NOT and LE reserved words.

6.2.2

Allocate/Deallocate must be implemented with garbage collection if portability problems are to be avoided.

6.2.2  
Can ALLOCATE be used to implement variable-length character strings?

6.2.4.3  
If I pass an array section to a subroutine, what are the on-going ranges to be used inside the subroutine?

6.2.5  
The IDENTIFY, SET RANGE, ALIAS etc statement are rather clumsy; has thought been given to some form of explicit manipulation of array descriptors?

6.2.6  
Does an ARRAY IDENTIFY statement always produce a constant stride across the parent object?

7.1  
Why is there no conditional expression (e.g. A=IF(X.EQ.0,0,1))?

7.5.2  
I agree that the WHERE construct should be iterated for consistency. A practical example, to evaluate the gamma or factorial function (which has poles at negative integral values) is

```
WHERE(A<>AINT(A))
  B=REAL FACTORIAL(A)           ! function
ELSE WHERE (A>=0)
  B=REAL FACTORIAL(NINT(A))    !by iteration?
ELSE
  B=0                           !bombout
END WHERE
```

8.1.2  
Why is no 'EXIT' facility available for passing control to the end of an IF block? Such a facility would make it possible in a great number of cases to avoid the dreaded GOTO.

8.1.3  
The "CASE" after "SELECT" is redundant and confusing. There are no other types of SELECT, and CASE is used in two senses.

8.1.3  
The case selectors in a Case construct are not required to be constants. If the expressions supplied involve Function evaluation is there any specified order in which these are executed?

15

8.1.3

The word CASE in SELECT CASE is redundant. If present, shouldn't the "match" be END SELECT CASE rather than just END SELECT?

8.1.4

DO construct. This construct is overcomplicated, especially when compared with the DO loop in Fortran 77. Specifically (a) since array manipulation in Fortran 8X will be often performed by array operations, surely the ability to have a real DO variable should be retained? Alternatively, (b) why not "abolish" the stop controls, just leaving the TIMES and unlimited DO, leaving the user to write explicitly the limit or 'while' or 'until' etc.

8.1.4

EXIT does not appear to be obligatory in an infinite DO END DO. Comments?

8.1.4 Is it practical to standardize commands to vectorize (i.e. parallel process) certain DO loops, or force them to be scalar processed, so that in this regard, CRAY fortran, CONVEX fortran etc would be identical?

8.1.4

Why is no simple conditional loop included?  
E.g. DO WHILE(condition)...END DO. Admittedly such a structure may be built with an unconditional DO - END DO and IF(condition)EXIT but this implies the condition must be evaluated to stop looping rather than the more common - continue condition. This structure is surely more commonly required than the DO-n TIMES structure given in the standard.

8.1.4

Why were WHILE and UNTIL loops not included as self documenting similes for DO;...BREAK and BREAK;...;END DO respectively?

8.1.4.1

In R817, if free format is not being used, does the do-construct-name start in column 7?

8.1.4.1

In R823, I would like to suggest that the construct name come before the END DO. This would make the END DO easier to find. For example:

```
FOURTH: DO I= 1,10
        FIFTH: DO J=2,15
        .
        .
        FIFTH: END DO
        .
        .
FOURTH: END DO
```

p. 505

9  
I/O statements in Fortran 77 are almost completely non-portable, largely because of the perversity or incompetence of many compiler suppliers. The Fortran 8X specification does not seem to improve this very much. The worst areas are error handling, and file existence and allocation. Numeric conversion is also poor.

9  
More thought to stream I/O would be nice, even as an extension. The UNIX implementations of Fortran typically have to jump through hoops to simulate record I/O. Some support would be nice.

9  
What was the reason for the absence of support for keyed Access Files in F8X I/O?

9.3.4  
How many characters should a standard conforming program reserve for the FILE= and NAME= parameters?

9.4  
In an input-list variable of type CHARACTER I would like a way of discovering how many characters were actually supplied with the READ was satisfied.

9.4.1.5  
Can the standard recommend a few error status code for I/O operations, like 0 -operation good; 1 - unexplained error, do not attempt further I/O; 2 - conflict between ACCESS and OPEN; 3 - conflict between declared and actual record length; 4 - illegal data in numeric input field; 5 - as four, but file connected to terminal so can issue read request etc.

9.4.5  
I don't think saying that it is up to the processor to decide whether an output device is a printer is adequate. I would like the application program to state how the first character of each record on a given I/O unit is to be interpreted. (cf VAX CARRIAGECONTROL='LIST' on OPEN).

9.4.5  
It seems a great pity that the Draft has eliminated all the most archaic restrictions of Fortran except one: the need for a "printer" to chop off the first character of each record and use it for carriage-control. Present compilers deal with this in a variety of ways, all of them messy and non-portable. Should there not be support for pure text files using something like the VAX/VMS Fortran extension of an additional option for the OPEN statement (CARRIAGECONTROL='LIST')?

10.2.1

It would be useful if edit descriptors (3.1.1) were allowed to be either upper case or lower case, e.g. lX, e12.5. If this is not possible, this should be caught and flagged as an error during compilation.

10.5.1

For similar {bit manipulation} reasons it would also be useful to have format edit descriptors for octal and hexadecimal I/O with integers: again most current compilers recognise the need.

10.5.1

The EDIT descriptor Z (hexadecimal) is not in the language. Was it considered and rejected or is it not thought a suitable EDIT descriptor for a formula language?

10.5.1.2.2

It would be preferable if, in the Ew.d edit descriptor, the form of the output was +x1.x2...xd exp rather than 0.x1x2...xd exp if no scale factor is given.

10.6.5.1

Why is F format affected by the scale factor?

10.8.1

Would it be possible to have some sort of free format input for character variables? For example, if A is a character variable, and B and C are numeric, allow READ (5,\*) A,B,C. The character variable could be delimited by quotes, or could be restricted to a single word, where a word is the collection of characters between a pair of blanks or commas. For example, in the example above, the input string might be: Jones,30,45 or 'Tom Jones', 30, 45

11.3

How many of the linking requirements are satisfied by existing linkers? In particular, 11.3.2 allows the export of imported objects; this adds several nasty problems, especially with linkers that cannot support 31 character external names (not to say generics).

11.3

If the use of 'COMMON' is to be a deprecated feature, then what is its replacement in the '8X' standard and what considerations, if any, have been given to issues such as performance?

11.3

Is the name 'MODULE' used to define the new type of program unit, first mentioned under modular definition p.ii foreword, going to be confused with the name module which is output with error messages under present (VAX) compilers? E.g. missing operand or delimiter symbol in module n at line n. Why not define it as 'GLOBAL'?

11.3

Why has the "Include" statement, which appears in the DEC and IBM extensions to Fortran 77, not been included in 8X?

11.3.2

Please give an example of how you expect external references in modules to be resolved. It seems that the compiler needs some indication of where to find the compiled code (or source) of modules mentioned in USE statements? I would expect to see something like USE A MODULE FROM "A FILE" but the filename cannot be given with the specified syntax.

12

It would be useful if the number of parameters passed to a subroutine is checked to make sure it is the same as the number of parameters in the subroutine call. This could be done at link time.

12.1.2.3

Ability to store a dummy procedure argument in a variable is a requirement of GINO-F implementations (of more use to us than pointers)

12.3

Does the procedure interface facility provide for default values?

13.10

Intrinsic procedure DATE-AND-TIME refers to obsolete timescale GMT. Local times are now referred to as UTC (coordinated Universal Time) and include leap seconds (at July 1 and/or January 1 as required). Thus the SECOND argument ought to be allowed to go to values > 59 (e.g. there is going to be a 1987 Dec 31 23h 59m 60s). There are several equally unpalatable options, including use of the UT or TAI timescales, which don't have leap seconds.

13.9.10

Re. new array intrinsic functions: I would like the facility to use SUM on a MASK array to find out how many elements are true (and similarly on a BIT array when available): this appears not to be allowed by the standard (real, int, or complex only). (Similarly for other functions no doubt.)

13.9.12

Has any consideration been given to the manipulation of sparse arrays, if not, why not?

13.9.5

Would it be possible to have LENGTH as well as LEN as the length function?

13.9.7

What restrictions are there on the use of the TRANSFER function, particularly for data packing/unpacking applications?

13.9.9

Why can MATMUL only accept arguments with 1 or 2 subscripts? MATMUL (A,B)=..... is perfectly well defined.

14.1.2.1

What is the reason for prohibiting a name clash between a common block name and a named constant while allowing a clash with a local variable. (14.1.2.1)

14.7.2

Does the standard define how the elements of a two or more dimensioned array are to be laid out in memory?

14.8

Initialization: the initialization of variables should be an explicit part of the standard. At present it is machine dependent - IBM machines zero all variables automatically at the start of execution; others, such as DEC machines do not. Either all variables should be initialized to zero or allow this as an option, or, if initialization is not done, have the linker list all variables which are not explicitly initialized.

B.2

Obsolescent Features (B.2 page B.1). Specifically, why are Alternate return and Assigned GOTO (with ASSIGN) included here instead of Deprecated? Neither of the suggested alternatives is available in Fortran 77.

B.3

I am disappointed to see that the alternate RETURN mechanism (12.4.1, 12.5.2.6) is deprecated. Just as ERR=14 and END=14 are valuable in I/O statements, some mechanism for dealing with unusual, but predictable, occurrences within subroutines (without explicit testing after every call) is desirable. The keyword argument facility could make the use of alternative returns in subroutine calls much more akin to the ERR=14, END=14 in I/O statements.



5

B.3

Why is fixed form source code now deprecated? This may mean that all present Fortran programs will be redundant in 20 years or so.

B.3.1.6

The ALLOCATE, IDENTIFY etc. features are not a complete replacement for EQUIVALENCE. The latter is heavily used in packages (Genstat, GLIM, SPSS, Minitab etc) to enable the package to write its own compacting garbage collector. ALLOCATE is not adequate for this, largely because it can lead to uncontrolled fragmentation and unacceptable performance or even program failure.

F.1.1

BIT FIELD (BIT SUBSTRING) intrinsics would be of more use to us than BIT TYPE

F.1.1

Since the BIT data type has been removed from the Draft, would it be possible to provide intrinsic functions for bit-wise logical operations on integers? There is a clear need for this in programs which interface directly to digital hardware and most current compilers support it with functions such as IAND and IOR.

F.1.2

Why were variant-types rejected?

F.4

Are there any facilities for event handling/exception handling?

F.Miss

I would like POINTERS - what is the possibility of their appearing in 8x?

F.Miss

If public comment places sufficient pressure on X3J3 to include pointer functionality what is the estimated delay to ratification of the standard?

F.Miss

Proper varying length CHARACTER strings would be more welcome than a host of INTRINSIC functions.

76

To: X3J3  
From: Miles Ellis  
Subject: UK response (NO) to the ISO Review of X3J3-S8

The following response has been sent by the British Standards Institute to ISO as its formal response to a review of the Draft Fortran 8x Standard.

\_\_\_\_\_ Start of Response \_\_\_\_\_

The UK votes 'no' in this ballot. While we approve the draft proposal in general, we believe that it is too flawed at this stage to be allowed to proceed to the next stage of processing without substantial changes. The changes we wish to see are significant enough to warrant a negative vote.

REQUIRED CHANGES

The UK believes that the proposal meets many of the needs of current users of Fortran and considers that the size of the proposed language to meet these needs and to allow for language evolution is reasonable and justifiable. The UK also believes it very important that the draft be progressed to a full standard in a timely manner, and hence gives notice that its vote would be changed to 'yes' if the following objections to the present draft were removed (note: the order shown is not a priority ordering and all of these objections will need to be removed to produce a positive vote):

- the proposed standard must contain bit manipulation facilities
- the proposal must contain a strongly typed pointer facility (see note)
- the proposed standard must specify the action to be taken by a processor when real type parameters specifying required precision cannot be satisfied
- the SETRANGE and RANGE statements must be deleted from the proposal
- the ALIAS attribute and the IDENTIFY statement must be deleted from the proposal
- the proposal as issued contains too many typographic errors and terminological and other editorial inconsistencies to be acceptable; these include the fact that the proposal does not contain all of ANS X3.9-1978.

Note: the UK would be prepared to change its 'no' vote to 'yes' even if this objection is not removed, provided all of the following conditions are satisfied:

- all of the other UK objections have been removed
- no other SC22 member body objects at this ballot on the basis of the absence of this facility
- a convincing case has been made out that incorporating this facility into the standard would cause a delay in the adoption of the standard which would be unacceptable to the UK.

HIGHLY DESIRABLE

The UK considers the following further changes to the proposal to be highly desirable and would support any other member body proposing their adoption:

- the proposal should allow integers to be parameterized in a manner analogous to that for reals
- there should be a format code to allow for input and output conversion for radices other than 10, notably 2, 8 and 16
- there should be additional conformance rules to enable the user to request detection and reporting of certain run-time errors such as array bound violation and numeric domain errors
- the restrictions on specification expressions should be removed.

DESIRABLE CHANGES

The UK considers the following changes to be desirable as improvements to the draft and would support their inclusion if adopted by the committee responsible for the draft:

- inconsistencies between the various control structures should be removed
- the WHERE construct should be removed and a suitable intrinsic array-valued function introduced in its stead
- the term "MODULE" should be replaced by a less confusing term
- the word "CONSTANT" should be introduced as a synonym for "PARAMETER" and the latter should be deprecated
- the word "CASE" after "SELECT" should be removed

- the blank character should be significant in the free source form
- the proposal should contain a facility for stream input/output.

ASPECTS OF THE DRAFT WHICH SHOULD NOT BE CHANGED

The UK approves of and would oppose any significant alteration to the following aspects of the draft:

- language architecture
- free form source (other than possible addition of significant blanks) - static conformance rules
  - parameterized reals
  - derived data types
  - array functionality
  - dynamic array allocation
  - control constructs
  - internal procedures
  - explicit interfaces and keyword calls
  - recursion
  - module/use concept.

Note: "would oppose" means that any departure, which the UK would regard as significant, in the revised draft from any of the above aspects would result in a continued negative vote from the UK even if all of the required and desirable changes listed earlier were included.

RATIONALE

Detailed justification, if required, for the above UK position may be found in detailed comments submitted by individuals in the UK which are being forwarded separately to the convenor of SC22/WG5, and can be confirmed or expanded upon if necessary through normal SC22/WG5 channels. The UK asks that all of the severalhundred such comments from the UK Fortran community be taken into account in revising the draft.

\_\_\_\_\_ End of Response \_\_\_\_\_

You should also read 107.TMRE-6 which contains the comments of the British Computer Society's Fortran Specialist Group concerning the above response. They were not very happy with it!

To: X3J3  
From: Miles Ellis  
Subject: British Computer Society comments on UK vote to ISO

I have been asked to forward the following comment from the Chairman of the BCS Fortran Specialist Group to X3J3. You will see that the Group does not agree with several of the comments in the official UK response to the ISO review of S8.

Message Starts

Date: 5-FEB-1988 12:50:42 GMT  
From: JDW@UK.AC.LEICESTER.VAX  
To: CTCMILES@UK.AC.OXFORD.VAX  
Subject: BCS FSG comments on S8

The BCS Fortran Specialist Group, at a meeting on the 21st January 1988, discussed the text of the letter vote from the BSI Fortran Panel on the S8 document.

The general response from the FSG (there were about 22 present) was that people concurred with the essence of the points made but were very unhappy that the vote had been NO and with the rather negative tone of the letter. I explained the BSI's reasons for this as best I could, and I think most people accepted this from BSI's point of view.

However, the meeting felt that the tone of the document was far too negative given the positive straw vote at the Forum. I was mandated to write to both CBEMA and the BSI expressing the view that the BCS Fortran Specialist Group is in favour of the 8X document (editorial, typographical and consistency errors having been corrected) and urge X3J3 to progress it with expediency.

Many people were also concerned that, according to the letter, if X3J3 does everything we ask except remove SETRANGE and RANGE, or ALIAS and IDENTIFY, then the UK will still vote NO ! Certainly that seems nonsense. Speaking personally I now feel it was a mistake to include these items in the top category.

Other points made were:

- the POINTER request should not be included in the top category as there was considerable debate on this and a rather split vote,
- the BIT statement should say we would find BIT intrinsics acceptable.

John Wilson  
5 Feb 1988

Message Ends

P. 514

**Japanese Proposal to FORTRAN8x  
-- NCHARACTER type for  
National Character Handling --**

**CONTENTS**

- **Comparison between NCHARACTER and CHARACTER(KIND=n)**
- **Reasons of the change of the Japanese position**
- **Summary of language Specifications**
- **Example of NCHARACTER feature**

**FORTRAN WG of Japanese National Committee  
for ISO/IEC JTC1 SC22/WG5**

**February , 1988**

*p. 515*

## 1. Comparison between NCHARACTER and CHARACTER(KIND=n)

Discussion Point	NCHARACTER	CHARACTER(KIND=n)	Notes
Functionality - Hiding of escape sequences from users  - Target kinds of character sets	{ good { bad  CHARACTER and NCHARACTER	good  bad  n=1(CHARACTER) and n=2, 3, 4, ..., n	almost all operation (type declaration, substring etc.)  formatted record length and column position
Efficiency	good	good	
Implementation	easy	easy ( $n \leq 2$ ) difficult ( $n \geq 3$ )	easy to implement only one national character set.

■ Major difference is the number of the target character sets.

NCHARACTER :2 (current CHARACTER and NCHARACTER)

CHARACTER(KIND=n) :more than 1 (maximum number is processor dependent)

## 2. Reasons of the changes of the Japanese position

We had been proposing 'CHARACTER(KIND=n)' until Liverpool meeting, but now we have changed to 'NCHARACTER' for the following reasons:

■ Practical needs during the FORTRAN8Xs is only one national character.

- In our investigation the needs of manipulating 3 or more kinds of characters (that is  $n \geq 3$ ) in a program will be small.

- Hardware such as handling input/output of 3 or more kinds of characters will presumably not be supplied during the next decade. Even if it were supported, the implementation cost would be high and would not be widely spread out.

■ NCHARACTER is a de facto standard in Japan.

- It is the draft proposal of Japanese standard on FORTRAN77 with extended character type and is now on pending to become standard.

- It is implemented by most vendors and widely used in Japan.

■ NCHARACTER supplies good consistency with other languages.

- We need good data-binding with other languages and we consider it is realized by NCHARACTER concept.
- Japan is now also proposing national character feature to ISO/SQL2. The concept of the proposal in SQL2 is consistent with NCHARACTER in FORTRAN.

■ Program portability by means of 'CHARACTER(KIND=n)' is not good.

- Registration mechanism for 'n' should be needed for ensuring program portability, but it will be impossible.
- No upward compatibility with current 'NCHARACTER' implementation.

■ The technical committee of Japanese language feature has recommended that all languages should include national character support by 'NCHARACTER' concept.

3. Summary of Language Specifications

Intrinsic data type - CHARACTER and NCHARACTER

■ Class of type character

- Type character is divided into two classes.

class of type character	type specification keyword
class-1	CHARACTER
class-2	NCHARACTER

- Characters included in the Fortran character set are of the class-1.
- The class-2 of type character is supplied for the use of alternate characters.

■ Source record length

- If a source record contains the class-2 characters, the maximum number of characters that the source record may contain is processor dependent.

P. 517



4. Collating sequence and Storage

■ Collating sequence

- There are no constraints on the location of the class-2 characters in the collating sequence.

■ Storage unit

- A character storage unit is also divided into two classes according to the class of type character.
- The relationship between the class-1 character storage unit and the class-2 character storage unit is not defined.

■ Storage association (COMMON, EQUIVALENCE etc.)

- Storage association is permitted only between the data objects of the same class of character storage unit.

5. Constant and Type specification

■ Constant

- class-1-char-literal-constant is  
' [class-1-character ] ... ' or " [class-1-character ] ... "
- class-2-char-literal-constant is  
NC' [class-2-character ] ... ' or NC" [class-2-character ] ... "

■ Type specification

- R502 type-spec is ...  
or CHARACTER [length-selector ]  
or NCHARACTER [length-selector ]
- CHARACTER specifies class-1 character type.
  - NCHARACTER specifies class-2 character type.
  - length-selector specifies the length of character object in each class.

p. 578

## 6. Use of character data objects

### ■ Substring

- The class of type character of a substring is same as that of the parent-string.

### ■ IDENTIFY statement (for scalar and array)

IDENTIFY (alias-name-parent)

▲ ▲

### ■ Expression (intrinsic operation)

op1 { concat-op } op2  
▲ { rel-op } ▲

concat-op : //

rel-op : .EQ. .NE. .LT. .LE. .GT. .GE.  
== <> < <= > >=

If one side is of type character, both(▲ and▲) classes must be same.

### ■ Intrinsic assignment

variable = expr

▲ ▲

### ■ Case construct

SELECT CASE (case-expr)

CASE case-selector

END CASE

If one side is of type character, both(▲ and▲) classes must be same.

### ■ STOP/PAUSE statement

STOP stop-code

PAUSE stop-code

If stop code is of type character, its class must be class-1.

## 7. Input and Output

### ■ Formatted record

- A record may consists of both class-1 and class-2 characters.
- The length of class-2 characters included in a formatted record is measured in class-1 characters, in a processor-dependent manner.

### ■ Internal file

- An internal file must be of class-1 character type.
- I/O list should not include class-2 character data.

### ■ I/O specifier (UNIT=~, STATUS=~, ACCESS=~ etc.)

- The specifiers must be of class-1 character type.

### ■ PAD

- Padding characters are blank characters of appropriate class of type characters. (blank character of class-2 for N format specification)

### ■ DELIM

- If APOSTROPHE or QUOTE is specified and the character item is of class-2 character, an NC precedes the leading delimiter.
- IF NONE is specified, the character item does not have preceded NC or surrounding delimiter.

### ■ A character indicating vertical spacing of formatted record

- The first character of the record must be of class-1 character.

### ■ Edit descriptor for class-2 character

R1005 data-edit-desc is N [w ]

where w is the length of a class-2 character

R1016 char-string-edit-desc is NC'class-2-char-str'

or NC" class-2-char-str"

### ■ Numeric editing and Logical editing

- All characters processed by numeric editing or logical editing are of class-1 character type.

### ■ Format specification (A editing, H editing etc.)

- Format specification must be of class-1 character.

### ■ Position editing (T, TL, TR, X editing)

- Position editing for records including class-2 characters is not allowed. (results in being undefined.)

### ■ List directed formatting and Namelist formatting

- Value separator etc. (.r\*...) must be class-1 character.

## 8. Character intrinsic functions

■ ADJUSTL, ADJUSTR, INDEX, LEN, LEN\_TRIM, SCAN, TRIM, VERIFY, REPEAT, ICHAR

- These generic functions can deal with both class-1 and class-2 characters.

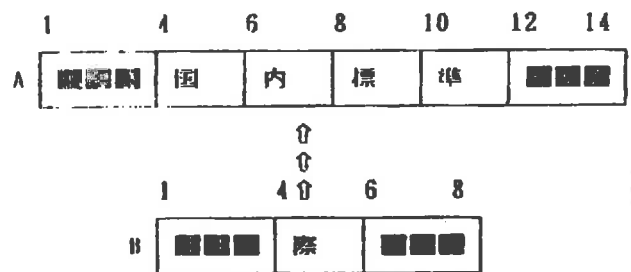
■ NCHAR

- The functionality of NCHAR corresponds to that of CHAR of class-1 character.

## 9. Example of NCHARACTER feature

■ When KANJI feature is not supported, user must take care of the escape sequences.

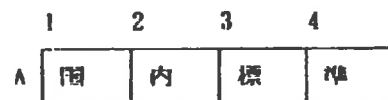
```
PROGRAM KANJI1
CHARACTER A*14, B*8
A= '国内標準'
B= '際'
WRITE(2, 100) A
A(6:7)=B(4:5)
WRITE(2, 100) A
100 FORMAT (IX, A14)
STOP
END
```



■:escape sequence code(3 bytes)

■ When KANJI feature is supported, user can handle KANJI strings easily.

```
PROGRAM KANJI2
NCHARACTER A*4
A=NC'国内標準'
WRITE(2, 100) A
A(2:2)=NC'際'
WRITE(2, 100) A
100 FORMAT (IX, N4)
STOP
END
```



(2)

Japanese Proposal for  
Handling Multi-octet Character Sets  
in DP 1539 FORTRAN

SECTION 2     FORTRAN TERMS AND CONCEPTS

75

1. Page 2-1, line 41

"CHARACTER, LOGICAL" should read "CHARACTER, NCHARACTER, LOGICAL".

2. Page 2-7, line 26, 27

"CHARACTER, and" should read "CHARACTER, NCHARACTER, and".

3. Page 2-9, line 5

Add the following sentences after "numeric and character." :

There is one class of character storage units to one class of type character(3.1). So, there are two classes of character storage units: the class-1 character storage unit and the class-2 character storage unit.

4. Page 2-9, line 15

Add the following sentence after "associated." :

Objects having different classes of character storage units must not be storage associated(3.1).



class-1 character. The class-2 character has a blank character.

There are no constraints on the location of the characters in the collating sequence not included in the Fortran character set.

78

11. Page 3-4, line 41

Add the following sentence after "characters." :

Note that if a source record contains the class-2 characters, the maximum number of characters that the source record may contain is processor dependent.

12. Page 3-5, line 17

Add the following sentence after "characters." :

Note that if a statement contains the class-2 characters, the maximum number of characters that the statement may contain is processor dependent.

13. Page 3-5, line 20

Add the following sentence after "long." :

Note that if a source line contains the class-2 characters, the number of characters that the source line may contain is processor dependent.



3

SECTION 4    INTRINSIC AND DERIVED DATA TYPES

1. Page 4-5, Line 23

Add the following sentences after line 23:

The class-1 of type character has a set of values composed of character strings of the class-1 of type character. The class-2 of type character has a set of values composed of character strings of the class-2 of type character. Each character string must be a sequence of characters of the same class of type character.

2. Page 4-5, Line 24

"The type specifier for the character type is the keyword CHARACTER."  
should read as follows:

The type specifiers for the character types are the keyword CHARACTER and NCHARACTER.

3. Page 4-5, Line 25

"A character literal constant is written as a sequence of characters delimited by either apostrophes or quotation marks."  
should read as follows:

A character literal constant is written as a sequence of characters delimited by either apostrophes or quotation marks. A character literal constant is a class-1 character literal constant or a class-2 character literal constant. A class-1 character literal constant is a sequence of class-1 characters delimited by either apostrophes or quotation marks. A class-2 character literal constant is an NC followed by a sequence of class-2 characters delimited by either apostrophes or quotation marks.

4. Page 4-5, Line 27

"R414 char-literal-constant is '[character]...' or "[character]..." "  
should read as follows:

R414.1 char-literal-constant is class-1-char-literal-constant  
or class-2-char-literal-constant

R414.2 class-1-char-literal-constant is '[class-1-character]...' or "[class-1-character]..."

R414.3 class-2-char-literal-constant is NC'[class-2-character]...' or NC"[class-2-character]..."

5. Page 4-6, Line 34

"CHARACTER" should read "CHARACTER or NCHARACTER".

SECTION 5 DATA OBJECT DECLARATIONS AND SPECIFICATIONS

1. Page 5-1, Line 23

Add the following line after line 23:

or NCHARACTER[length-selector]

2. Page 5-4, Line 4

"The CHARACTER type specifier " should read "The CHARACTER or NCHARACTER type specifier ".

3. Page 5-18, Line 25

Add the following sentences after "...only with other objects of type character.":

A data object of the class-1 of type character may be equivalenced only with other objects of the class-1 of type character. A data object of the class-2 of type character may be equivalenced only with other objects of the class-2 of type character.

4. Page 5-19, Line 44

Add the following sentences after line 44:

If an object of the class-1 of type character is in a common block, all of the entities in that common block must be of the class-1 of type character. If an object of the class-2 of type character is in a common block, all of the entities in that common block must be of the class-2 of type character.

SECTION 6 USE OF DATA OBJECTS

73

1. Page 6-1, line 43

Add the following sentence after line 43:

The class of type character of a substring is same as that of the *parent-string*.

2. Page 6-7, line 42

Add the following sentence after "type parameters.":

When the *scalar-alias-name* and the *parent* are of type character, both classes of them must be same.

3. Page 6-8, line 38

Add the following sentence after "eters.":

When the *alias-element* and the *parent-array-element* are of type character, both classes of them must be same.

SECTION 7 EXPRESSIONS AND ASSIGNMENT

1. Page 7-5, line 4

Add the following sentence after line 4:

A character intrinsic operation is an intrinsic operation where the operands are of the same class of type character.

2. Page 7-5, line 7

"where the operands are of type character" should read

"where the operands are of same class of type character".

3. Page 7-5, line 34

Add the following sentence after line 34:

Note that when an intrinsic operator is a character intrinsic operator (//), the type of the result of the operation has the same class of type character as that of the operands.

4. Page 7-14, line 6

"type character." should read

"type character where the operands are of the same class of type character".

5. Page 7-14, line 7

"type character." should read

"type character having the same class of type character as that of the operands."

6. Page 7-14, line 28

Add the following sentence after "< > .":

And when two operands are of type character, both classes of them must be same.

7. Page 7-15, line 19

"blanks" should read

"blanks of same class of type character as that of the operands".

8. Page 7-18, line 26

"type character" should read

"same class of type character".

9. Page 7-18, line 41

Add the following sentence after line 41:

Note that when the type of *variable* and *expr* are character, both classes of them must be same.

10. Page 7-19, line 26

"blanks" should read

"blanks of same class of type character as that of *variable*".

## SECTION 8 EXECUTION CONTROL

1. Page 8-3, line 38

"allowed." should read

"allowed, but each *case-value* must be of the same class of type character as that of the *case-expr*."

2. Page 8-10, line 38

Add the following sentence after line 38:

Constraint: The *scalar-char-constant* must be of *class-1* of type character.

SECTION 9 INPUT/OUTPUT STATEMENTS

1. Page 9-1, line 27  
Add the following sentence after "zero.":  
If the record includes class-2 characters, the length of that character portion is measured in class-1 characters, in a processor-dependent manner.
2. Page 9-4, line 37  
Add the following sentence after line 37:  
(3) An internal file must be of class-1 character type.
3. Page 9-4, line 44  
Add the following sentence after line 44:  
Constraint: The char-variable must be of class-1 character type.
4. Page 9-6, line 48  
Add the following sentence after line 48:  
Constraint: The scalar-char-expr must be of class-1 character type.
5. Page 9-7, line 45  
"characters" should read "class-1 characters".
6. Page 9-8, lines 20 & 22  
"apostrophe(s)" should read "apostrophe(s) of the class-1 character type".
7. Page 9-8, lines 22 & 24  
"quotation mark(s)" should read "quotation mark(s) of the class-1 character type".
8. Page 9-8, line 24  
Add the following sentence after "doubled.":  
If APOSTROPHE or QUOTE is specified and the character constant is of the class-2, an NC precedes the leading delimiter.
9. Page 9-8, line 30  
"padded with blanks" should read "padded with blanks of the appropriate class character type".
10. Page 9-8, line 47  
Add the following sentence after line 47:  
Constraint: The scalar-char-expr must be of class-1 character type.
11. Page 9-11, line 15  
Add the following sentence after line 15:  
Constraint: The char-expr must be of class-1 character type.
12. Page 9-16, line 36  
"padded with blanks" should read "padded with blanks of appropriate class character type".
13. Page 9-16, lines 43-44  
"blank characters are added to fill the record." should read as follows:  
class-1 blank characters are added to fill the record, unless the record includes class-2 characters.
14. Page 9-17, line 7  
"is not printed." should read "must be of the class-1 and is not printed".
15. Page 9-19, line 30  
Add the following sentence after line 30:  
Constraint: The scalar-char-variable must be of class-1 character type.
16. Page 9-21, line 3  
"characters" should read "class-1 characters".

73

## SECTION 10 INPUT/OUTPUT EDITING

1. Page 10-1, line 30  
"expression" should read "expression of the class-1".
2. Page 10-2, line 23  
Add the following line after line 23:  
or N[w]
3. Page 10-2, line 30  
"A, and" should read "A, N and".
4. Page 10-3, line 10  
Add the following sentence after line 10:  
Constraint: The characters following H must be of class-1 character type.
5. Page 10-3, line 25  
Add the following sentence after "the field."  
All characters in a field must be of the same class character type.
6. Page 10-5, line 14  
Add the following sentence after line 14:  
(6) All characters processed by numeric editing are of class-1 character type.
7. Page 10-8, line 20  
Add the following sentence after "logical."  
All characters processed by logical editing are of class-1 character type.
8. Page 10-8, line 27  
"character." should read as follows:  
class-1 character. The N[w] edit descriptor is used with an input/output list item of type class-2 character.
9. Page 10-8, lines 28, 29, 31 & 35  
"A" should read "A or N".
10. Page 10-9, line 8  
"blanks" should read "blanks of class-1 character type".
11. Page 10-9, line 13  
Add the following sentence after line 13:  
All characters skipped by a T, TL, TR or X edit descriptor must be of class-1 character type.
12. Page 10-11, line 2  
"c characters" should read "c class-1 characters".
13. Page 10-11, line 10  
Add the following sentence after "value separators."  
All characters in a list-directed record are of class-1 character type, except within a class-2 character constant.
14. Page 10-11, line 11  
"a blank" should read "a blank of the class-1 character type".
15. Page 10-11, line 48  
Both "constant" should read "class-1 constant".
16. Page 10-12, line 4  
"the characters" should read "the class-1 characters".
17. Page 10-12, line 15  
"blanks" should read "blanks of the same class character type as the constant".
18. Page 10-12, line 28  
"blanks" should read "blanks of class-1 character type".

19. Page 10-13, line 10  
"marks," should read "marks and are not preceded by an NC,".
20. Page 10-13, line 14  
"a blank" should read "a blank of the class-1 character type".
21. Page 10-13, line 18  
"quotes," should read "quotes, are preceded by an NC in the case of class-2 constant,".
22. Page 10-13, line 21  
"apostrophes," should read "apostrophes, are preceded by an NC in the case of class-2 constant,".
23. Page 10-13, line 29  
"blank" should read "blank of the class-1 character type".
24. Page 10-13, line 32  
Add the following sentence after "value separators.":  
All characters in a namelist record are of class-1 character type, except within a class-2 character constant.
25. Page 10-13, line 34  
"a blank" should read "a blank of the class-1 character type".
26. Page 10-15, line 20 & 22  
Both "character constant" should read "class-1 character constant".
27. Page 10-15, line 31  
"blanks" should read "blanks of the same class character type as the constant".
28. Page 10-15, line 35  
"The character" should read "The class-1 character".
29. Page 10-16, line 5  
"blanks" should read "blanks of the class-1 character type".
30. Page 10-16, line 37  
"marks." should read "marks and are not preceded by an NC,".
31. Page 10-16, line 41  
"a blank" should read "a blank of the class-1 character type".
32. Page 10-16, line 45  
"quotes," should read "quotes, are preceded by an NC in the case of class-2 constant,".
33. Page 10-17, line 2  
"apostrophes," should read "apostrophes, are preceded by an NC in the case of class-2 constant,".
34. Page 10-17, line 16  
"blank" should read "class-1 blank"





- NCHAR(I)        NCHAR(I)        integer
- NINDEX(S,T)    INDEX(S,T)     class-2 character

11. Page 13-10, line 31

Add the following lines after line 31:

- NLEN(S)         LEN(S)         class-2 character
- NLEN\_TRIM(S)    LEN\_TRIM(S)    class-2 character
- NREPEAT(S,N)    REPEAT(S,N)    class-2 character
- NSCAN(S,S,B)    SCAN(S,S,B)    class-2 character
- NTRIM(S)        TRIM(S)        class-2 character
- NVERIFY(S,S,B)  VERIFY(S,S,B)  class-2 character

12. Page 13-11, line 27

"character" should read "class-1 character".

13. Page 13-11, line 31

Add the following sentences after line 31:

13.12.4-1 NADJUSTL(String).

Description. Adjust to the left, removing leading blanks and inserting trailing blanks.

Kind. Elemental function.

Argument. STRING must be of type class-2 character.

Result Type and Type Parameters. Character of the same length as STRING.

Result Value. The value of the result is the same as STRING except that any leading blanks have been deleted and the same number of trailing blanks have been inserted.

Example. NADJUSTL(NC' 四五') has value NC'四五'.

14. Page 13-11, line 35

"character" should read "class-1 character".

15. Page 13-11, line 39

Add the following sentences after line 39:

13.12.5-1 NADJUSTR(String).

Description. Adjust to the right, removing trailing blanks and inserting leading blanks.

73

Kind. Elemental function.

Argument. STRING must be of type class-2 character.

Result Type and Type Parameters. Character of the same length as STRING.

Result Value. The value of the result is the same as STRING except that any trailing blanks have been deleted and the same number of leading blanks have been inserted.

Example. NADJUSTR(NC'五六 ') has value NC'五六'.

16. Page 13-14, line 34

"character" should read "class-1 character".

17. Page 13-14, line 38

"characters" should read "class-1 characters".

18. Page 13-14, line 39

"Character" should read "Class-1 character".

19. Page 13-14, line 40

"character" should read "class-1 character".

20. Page 13-15, line 1

"character" should read "class-1 character".

21. Page 13-15, line 4

Add the following sentences after line 4:

13.12.15-1 NCHAR(I).

Description. Returns the class-2 character in a given position of the processor collating sequence. It is the inverse of the function INCHAR.

Kind. Elemental function.

Argument. I must be of type integer with a value in the range  $0 \leq I \leq n$ , where n is the processor-dependent number.

Result Type and Type Parameters. Class-2 character of length one.

Result Value. The result is the class-2 character in position I of the processor collating sequence. INCHAR(NCHAR(I)) must have the value I for  $0 \leq I \leq n$  and NCHAR(INCHAR(NC'四')) must have the value NC'四' for any class-2 character NC'四' capable of representation in the processor.

P. 535

22. Page 13-27, line 11  
"a character" should read "a class-1 character".

23. Page 13-27, line 13  
"character" should read "class-1 character".

24. Page 13-27, line 23  
"character" should read "class-1 character".

25. Page 13-27, line 25  
"character" should read "class-1 character".

26. Page 13-27, line 26  
"character" should read "class-1 character".

27. Page 13-27, line 29  
"characters" should read "class-1 characters".

28. Page 13-27, line 30  
"characters" should read "class-1 characters".

29. Page 13-27, line 35  
Add the following sentences after line 35:

13.12.45-1 INCHAR(NC'五').

Description. Returns the position of a class-2 character in the processor collating sequence.

Kind. Elemental function.

Argument. NC'五' must be of type class-2 character and of length one.

Its value must be that of a class-2 character capable of representation in the processor.

Result Type. Integer.

Result Value. The result is the position of NC'五' in the processor collating sequence and is in the range  $0 \leq \text{INCHAR}(\text{NC}'五') \leq n$ , where n is the processor-dependent number.

30. Page 13-27, line 40  
"character" should read "class-1 character".

31. Page 13-27, line 41

"character" should read "class-1 character".

32. Page 13-28, line 15

Add the following sentences after line 15:

13.12.46-1 NINDEX(String, SUBSTRING, BACK).

Optional Argument. BACK

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

Kind. Elemental function.

Arguments.

STRING must be of type class-2 character.

SUBSTRING must be of type class-2 character.

BACK(optional) must be of type logical.

Result Type. Integer.

Result Value.

Case(i): If BACK is absent or present with the value .FALSE., the value returned is the minimum value of I such that STRING(I:I+NLEN(SUBSTRING)-1)=SUBSTRING or zero if there is no such value. Zero is returned if NLEN(STRING) < NLEN(SUBSTRING) and one is returned if NLEN(SUBSTRING)=0.

Case(ii): If BACK is present with the value .TRUE., the value returned is the maximum value of I such that STRING(I:I+NLEN(SUBSTRING)-1)=SUBSTRING or zero if there is no such value. Zero is returned if NLEN(STRING) < NLEN(SUBSTRING) and NLEN(STRING)+1 is returned if NLEN(SUBSTRING)=0.

Examples. NINDEX(NC'四五六', NC'五') has value 2.

NINDEX(NC'四五六', NC'四', BACK=.TRUE.) has the value 3.

33. Page 13-28, line 31

"character" should read "class-1 character".

34. Page 13-28, line 33

"character" should read "class-1 character".

35. Page 13-28, line 39

Add the following sentences after line 39:

13.12.48-1 NLEN(String).

Description. Returns the length of a class-2 character entity.

Kind. Inquiry function.

Argument. STRING must be of type class-2 character. It may be scalar or array valued.

Result Type and Shape. Integer scalar.

Result value. The result has value equal to the number of character in STRING if it is scalar or in an element of STRING if it is array valued.

Example. If C is declared by the statement

```
NCHARACTER=11 C(100)
```

NLEN(C) has value 11.

36. Page 13-28, line 41

"character" should read "class-1 character".

37. Page 13-28, line 41-line 42

"characters" should read "class-1 characters".

38. Page 13-29, line 2

"character" should read "class-1 character".

39. Page 13-29, line 5

"characters" should read "class-1 characters".

40. Page 13-29, line 7

Add the following sentences after line 7:

13.12.49-1 NLEN\_TRIM(String).

Description. Returns the length of the class-2 character argument without trailing blank class-2 characters.

Kind. Elemental function.

Argument. STRING must be of type class-2 character.

Result Type. Integer.

Result Value. The result has a value equal to the number of characters after any trailing blanks in STRING are removed. If the argument contains no nonblank class-2 characters, the result is zero.

Examples. NLEN\_TRIM(NC' 五 四 ') has value 4. and NLEN\_TRIM(NC' ') has value 0.

41. Page 13-41, line 18  
"a character" should read "a class-1 character".

42. Page 13-41, line 18  
"characters" should read "class-1 characters".

43. Page 13-41, line 21  
"character" should read "class-1 character".

44. Page 13-41, line 22  
"character" should read "class-1 character".

45. Page 13-41, line 38  
Add the following sentences after line 38:

13.12.80-1 NSCAN(String, SET, BACK).

Optional Argument. BACK

Description. Scan a string for a class-2 character in a set of class-2 characters.

Kind. Elemental function.

Arguments.

STRING                    must be of type class-2 character.

SET                        must be of type class-2 character.

BACK(optional)        must be of type logical.

Result Type. Integer.

Result Value.

Case(i): If any of the characters of SET appears in STRING, the value of the result is the position of the leftmost character of STRING that is in SET. The result is zero if STRING does not contain any of the characters that are in SET or if the length of STRING or SET is zero. The default value of BACK is .FALSE. and its inclusion is optional when processing starts with the first character or STRING.

Case(ii): If STRING is to be processed starting with the last character, BACK must contain the logical value .TRUE., the value of the result is the position of the rightmost character of STRING that is in SET. The result is zero if STRING does not contain any of the characters that are in SET or if the length of STRING or SET is zero.

Examples. NSCAN(NC'五十步百步', NC'百十') has value 2.

NSCAN(NC'五十步百步',NC'百十',BACK=.TRUE.) has value 4.

73

46. Page 13-47, line 2

"characters" should read "class-1 characters".

47. Page 13-47, line 4

"character" should read "class-1 character".

48. Page 13-47, line 5

"Character" should read "Class-1 character".

49. Page 13-47, line 7

Add the following sentence "(class-1 character)" after "blanks".

50. Page 13-47, line 8

"characters" should read "class-1 characters".

51. Page 13-47, line 9

Add the following sentences after line 9:

13.12.95-1 NTRIM(String).

Description. Returns the argument with trailing blank class-2 characters removed.

Kind. Transformational function.

Argument. STRING must be of type class-2 character and must be a scalar.

Result Type and Type Parameters. Class-2 character with a length that is the length of STRING less the number of trailing blanks in STRING.

Result Value. The value of the result is the same as STRING except any trailing blanks(class-2 character) are removed. If STRING contains no nonblank class-2 characters, the result has zero length.

Example. NTRIM(NC' 四 五 ' ) has value NC' 四 五 '.

52. Page 13-47, line 33

"of characters" should read "of class-1 characters".

53. Page 13-47, line 33

"the characters" should read "the class-1 characters".

p. 540

54. Page 13-47, line 36

"character" should read "class-1 character".

55. Page 13-48, line 1

"character" should read "class-1 character".

56. page 13-48, line 12

Add the following sentences after line 12:

13.12.97-1 NVERIFY(String, SET, BACK).

Optional Argument. BACK

Description. Verify that a set of class-2 characters contains all the class-2 characters in a string.

Kind. Elemental function.

Arguments.

STRING must be of type class-2 character.

SET must be of type class-2 character.

BACK(optional) must be of type logical.

Result Type. Integer.

Result Value. The value of the result is zero if each character in STRING is in SET or if STRING has zero length; otherwise, there are two cases as follows:

Case(i): If BACK is absent or present with the value .FALSE., the value of the result is the position of the leftmost character of STRING that is not in SET.

Case(ii): If BACK is present with the value .TRUE., the value of the result is the position of the rightmost character of STRING that is not in SET.

Examples. NVERIFY(NC'三三五五', NC'七五三') has value 0.

Case(i): NVERIFY(NC' $\pi \leq \phi \leq \infty$ ', NC' $\leq \pi$ ') has the value 3.

Case(ii): NVERIFY(NC' $\Delta \odot \otimes$ '  $\odot$ ', NC' $\odot$ ', BACK=.TRUE.) has the value 4.



79

79a

10/(\* )LWC-4  
January 31, 1988  
Page 1 of 2

To: X3J3  
From: L. Campbell and L. Rolison

Subject: Corrections and Edits for S16 dated December 1987

To agree with the minutes of the Nov. 87 meeting and our notes, the following corrections to S16 are needed:

1. In third line of introduction, "came" should be "are".
2. Item 5, "procedure" should be "statement".
3. Item 8, "subroutine" should be in italics.
4. Item 27, delete "a" before "function".
5. Item 32, first "character" should be in italics. Add space after "to".
6. Item 33, add comma after first "PRIVATE".
7. Item 35, ending comma should be a period.
8. Item 38, "assumed type parameter" should be in bold.
9. Item 39, "lowerbound" and "upperbound" should each be two words.
10. Item 39, add comma after "negative".
11. Item 47, "Constraint" should not be in bold and "parens" should be "parent".
12. Items 47 & 121, the vote (13-0) should be (15-0).
13. Items 50 & 73, the round black dots should be square.
14. Item 67, add a comma after "DO construct".
15. Item 71, delete comma after "fragment", delete period after "8), and add space after "!".
16. Item 72, "loop" should be in bold in "called a loop".
17. Items 73, 82, & 83, "Constraint" should not be in bold.
18. Item 73, in first line of R821, delete space before the second comma.
19. Item 73, in first two constraints, "default real, or double precision real" should be in small font. (twice)
20. Item 75, all text in 8.1.4.1.2 should be in small font.
21. Item 73, in second constraint of 8.1.4.1.2, "stmts" should be "stmt".
22. Item 73, in next to last paragraph, add "a" before "nonblock".
23. Item 73, in last paragraph, "share" should be in bold.
24. Item 74, "range" should be in bold at beginning of both paragraphs.
25. Item 74, whole second paragraph should be in small font.
26. Item 75, second "loop" should be in bold.
27. Item 78, old and new text for lines 18 & 19 should be in small font.
28. Item 79, add the vote "(22-1)" at end.
29. Item 82, in first line of paragraph after constraint, "belong" should be in bold.
30. Item 82, need small font for last sentence of next to last paragraph and last two sentences of last paragraph.
31. Item 83, in first line of paragraph after constraint, "belong" should be in bold.
32. Item 83, list "a-f" should be "(1)-(6)".
33. Item 84, in both DO statements in Ex. 1, the character after the "=" should be a numeric "1".
34. Item 84, in Examples 4 & 8, add period after "SUM = 0".
35. Item 84, add space after "!" in 5th & 6th lines of Ex. 4 and in 5th line of Ex. 8.
36. Item 84, in second line of paragraph after Ex. 4, delete comma after "MAX(N,0)".
37. Item 85, "Statement labels" and "statement label" should be in bold.

792

38. Item 89, "correct" should be "current".
39. Items 101 & 103, "file" should be in italics.
40. Item 103, line "28" should be "18".
41. Item 111, "r" should be in italics.
42. Item 135, "Constraint" should not be in bold.
43. Item 172, add large parentheses around matrix. (See new edit below.)
44. Item 197, line "34" should be "44".
45. Items 197 & 198, "return" should be in italics.
46. Item 205, "length" should be in italics.
47. Item 214, "lines 1 and 20" should be "line 20".
48. Item 224, add a comma after "conforming".
49. Item 230, list "a-c" should be "(1)-'3)".
50. Item 230, add "The" before "CASE" in title of C.8.2.
51. Item 231, add "Page C-4, line 43+, add the following:" at beginning.
52. Item 231, move the vote "(22-1)" to the last line.
53. Item 233, add period after "Files".
54. Item 265, move ending period inside the quote mark.
55. Item 269, delete "(1)" and "(2)" in subroutine SAVE\_VAR.
56. Item 278, delete item. (apparently wasn't voted on)
57. Item 86, "label" should be in italics.

The following new edits to S16 are suggested:

61. Items 3, 4, 163, 260, & 273, delete "(Ed.)".
- ~~62.~~ Item 32, add second sentence of new text to beginning of existing paragraph at line 29 of S8.
63. Item 46, change line "12" to "13".
64. Interchange items 57 & 58.
65. Items 53 & 224, delete "(Tech)".
66. Item 63, delete ", provided the ... change made?]", (see item 171)
67. Item 70, change "(80) LINE" to "(80) :: LINE".
- ~~68.~~ Item 73, "construct" in title of 8.1.4.1 & 8.1.4.1.2 should be "Construct".
69. Item 73, in first line, change "can" to "may".
- ~~70.~~ Item 74, "construct" in title should be "Construct".
71. Item 83, in list item "c" (should be (3)), add "a" before "CYCLE".
- ~~72.~~ Item 84, "constructs" in title should be "Constructs".
73. Item 84, in third line from end of Ex. 2, change "I/O" to "input/output". Also in last line of Ex. 7.
74. Item 84, in last line of example 2, add "statement" after "READ".
75. Item 84, in next to last line of Ex. 7, delete "an" before "ENDDO".  
~~(or also add "a" before "CONTINUE")~~
76. Item 135, add period after "-list".
77. Item 148, delete " & 32" in first line.
78. Item 172, enclose matrix in large brackets (instead of parentheses).
79. Item 187, Change "S" to "s".
80. Item 269, in last two lines on page 23, change "machine's" to "processor's" and change "machine" to "processor".
- ~~81.~~ Vote on item 278 (deletion of App. F) and keep item if it passes.
82. Item 230, move quote mark to follow "logical-expr" after "DO WHILE" and "DO UNTIL".
83. Item 7, lines "27-28" should be "27 and 28". Also add "(twice)".
84. Item 35, lines "11-12" should be "11 and 12". Also add "(twice)".

107-CDB-7  
February 5, 1988

SD

From : Carl Burch

To : X3J3

Subj : Subgroup Nominations for Public Review Letters 65-91

Please find attached the annotated copies of Public Review letters 65-91, marked with my recommendations for subgroup assignments. All subgroup assignments are negotiable between the Subgroup Chair and the Public Review Working Group. Any omissions noted should be brought to the attention of the Public Review Working Group.



#65

79

DECUS Meeting  
Dallas, Texas  
December 15, 1987

Review Letter # 65			
Subgroup Nominations :			
1	DATA	11	
2	PROC	13	
3	DATA	13	
4	GEN	14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. (65.1)
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. (65.2)
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. (65.3)
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. (65.4)

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

*James R. ...*

Company: INTERNATIONAL POWER MACHINES  
Address: 2975 MILLEX PARK N  
GARLAND TX 75042  
(214) 272 8000

#64



# King Computer Search, Inc

Technical Recruiting and Executive Se.  
9221 LBJ Freeway, Suite 208  
Dallas, Texas 75243  
(214) 238-1021

80

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

66.1

I believe that the CASE construct is a needed addition. I believe that the numerical precision control is poorly designed modification the the language. (66.2)  
(66.3) → I am disappointed that the DO WHILE statement was not included. I am also  
(66.4) → shocked that the VMS RECORD structures was missing. I am unhappy that the  
(66.5) → INCLUDE statement was not defined. I feel that the COMMON statement must not (66.6)  
be removed ever.

Sincerely,

David Atkinson

X3 SECRET  
'88 JAN 20 P1:52

Review Letter # 66			
Subgroup Nominations :			
1	CIO	11	
2	DATA	13	
3	CIO	13	
4	DATA	14	
5	PROC	15	
6	GEN	16	
7		17	
8		18	
9		19	
10		20	

#67

-V-

### Accredited Standards Committee 3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/87-11-075-X, S  
Comment 167  
Date: January 22, 1988  
Project: 67-R  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT 167

Attached is Comment 167 on BSR X3.9-198x from L. Matton of Programming Research Ltd.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part a proposal, the response should provide the rationale for

The comment should be discussed at the next TC meeting, responded to at once, an interim acknowledgment should be sent to the commentors of their need to notify the Secretariat of their dissatisfaction with the committee's response. The comment should be sent to the Secretariat a written statement indicating acceptance of the response within fifteen working days. The comment should be sent to the Secretariat within fifteen working days that failure to respond within fifteen working days that will indicate to the Secretariat that the comment is to

*Catherine A. Kaphurik*  
Catherine A. Kaphurik  
Administrative Secret

Attachment: Comment 167  
cc: Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

P. 547

### Programming Research Ltd.

"Oakwood", 11, Cutham Rd., New Malden, Surrey, KT3 3AJ, England  
Tel: 01-949-3537 Telex: 8950511 ONEONE G Quoung Ref. 13724001

X3 Secretariat/CBEMA  
Fortran Public Review  
311, First St. N.W.  
Suite 500  
Washington D.C. 20001-2178  
U.S.A.

88 JAN 20 1988

### Comments on the proposed revision to the FORTRAN standard

I welcome the fact that the standard has now appeared after so much obvious hard work amidst difficult conditions. My main interest as a practising scientist and former project leader for a very large FORTRAN system, (>600,000 lines), is not a plethora of features but the underlying reliability and portability of the language itself which as far as I can see needs some urgent attention.

I would like to make the following comments:

Reliability is inversely proportional to size and the size of FORTRAN 8X is an immediate worry. The new notion of planned obsolescence is very laudable but the necessity of being downwards-compatible puts grave strains on the development of the language and can be compared to Intel's efforts with microprocessors and the modern trend to RISC. Might I suggest tying the incremental deprecation, obsolescence and deletion features of FORTRAN to the old source form and the new features to the new source form, thus creating two more manageable languages.

I find it extraordinary that a language can support modern reliability features like MODULES and allow non-significant blanks in variables and "keywords", a source of many famous FORTRAN software errors over the years. Although an emotive issue, perhaps the protagonists of non-significant blanks would like to pay for the costs of errors arising from their existence. Please, please can we disallow blanks in variables and "keywords", restricting their legality to constants only, where they serve a very useful and safe function.

### FORTRAN7I example

DATA A, B, C, D, E, F .....

which has a comma missing between C and D. This code was ported to five

Review Letter # 67	
Subgroup	Nominations:
1	GEN 11
2	GEN 13
3	GEN 13
4	DATA 14
5	GEN 15
6	CIO 16
7	PROC 17
8	DATA 18
9	
10	

80

67.1

67.2

~V~

Programming Research Ltd.

"Oakwood", 11, Carlton Rd., New Malden, Surrey, KT3 3AJ, England  
Tel: 01-949-3537 Telex: 8950511 ONEONE G Quoting Ref: 13724001

different machines before it was spotted and led to a correction cost of some \$10,000.

It disturbs me that the compiler writer is not forced to report errors whenever a dubious or non-standard construct is used. As an example, an ARRAY assignment statement involving two entities of dissimilar shape does not lead to automatic rejection. Rather, its result is undefined. What this means in practice is that every machine will respond in its own peculiar way and not bother telling the user. Debugging numerical software for which the only clue is that the numbers are wrong is no joke. If FORTRAN 8X is to be a competitor to Ada, then it must report errors accordingly.

I'm afraid that I consider the programmable floating-point precision features are no more than a toy. In the benign case where a particular precision is available, the underlying hardware will in general only have 32 and 64 (and possibly 80) bit arithmetic. So that is what the user is most likely to get whether they want it or not. (Surely no FORTRAN user would want an extremely slow software 33 bit equivalent in preference to a hardware 64 bits !!). In the case where a precision is not available, the result is processor dependent. This is extremely dangerous and will lead to many problems. Where something is not available, the user should be informed, not given something which may be inadequate.

Given that the IEEE 754 standard defines 32, 64 and 80 bit arithmetic and defines them well, why not just use these? Most modern chip sets can't do anything else.

Please reconsider allowing multiple assignments on the same line, an infamous contributor to unreadable (and therefore unreliable) code.

Please can you force a compiler warning for potentially infinite DO loops which do not contain an EXIT. I realise occasionally that a user may wish to do this deliberately such as in event-loops, but in general, I'm sure that it will be omitted accidentally leading to a great deal of wasted computer time.

I would just like to mention that the plan to pass single array elements by value not reference will wreck probably every major FORTRAN 77 code in existence.

I subscribe to the view that "pointers are the GOTO of the '80s", but an implementation seems far more regular than the features offered by IDENTIFY/SET RANGE.

I do not think that the pressure of urgency after a long gestation period is a good enough reason for releasing the language in its present form and would not vote for its acceptance as such.

67.2, cont.

67.3

67.4

67.4 cont.

67.5

67.6

67.7

67.8

~V~

Programming Research Ltd.

"Oakwood", 11, Carlton Rd., New Malden, Surrey, KT3 3AJ, England  
Tel: 01-949-3537 Telex: 8950511 ONEONE G Quoting Ref: 13724001

I'm sorry that I do not have the time to go through the standard in detail at this stage (three months is far too short a review period for such a major enterprise) and these are just general comments from a user. I believe that reliability is justifiably becoming the most important language feature and if FORTRAN 8X can introduce the concept of maximising the number of errors which can be detected along with all the good work which has been done already, then it has a future. It is however worth remembering that for the first time ever, a whole generation of scientific and engineering students are graduating without ever having seen FORTRAN and they may never do so without rapid progress. Many major software houses now standardise on C and its development, C++.

Yours sincerely,

L. Hutton

L. Hutton  
Managing Director

80

p. 548

January 13, 1988

'88 JAN 20 P2:29

William C. Leonard  
Harris Corp., Computer Systems Division  
2101 W. Cypress Creek Road  
Fort Lauderdale, FL 33309

Catherine Katchurik  
X3 Secretariat/CBEMA  
311 First Street NW, Suite 500  
Washington, DC 20001

cc: Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York City, NY 10018

Review Letter # 68			
Subgroup Nominations :			
1	GEN	11	PROC
2	GEN	12R	GEN
3	GEN	13	GEN
4	GEN	14	PROC
5	GEN	15	CIO
6	GEN	16	CIO
7	GEN	17	GEN
8	GEN	18	
9	GEN	19	
10	GEN	20	

Subject: Comments on Draft Proposed Revision of X3.9-1978 (FORTRAN)

The following are my comments on the Draft Proposed Revision of the American National Standard for Information Systems Programming Language FORTRAN (X3.9-198x), based on the S8, Version 104 document (S8.104).

**General Comments** I am opposed to the proposed revision of FORTRAN, for many reasons, the most important being that S8.104 is not a revision, it is a rewrite. Standards are supposed to control change; language standards especially were instituted to facilitate portability of programs, training of programmers, and some degree of uniformity among vendors' implementations. Major changes, such as S8.104 propose, negate these goals.

The proposed revision is also a major departure from the goals for the FORTRAN language in particular. From the beginning, FORTRAN implementations have had at least the following goals:

- fast compilation;
- high-quality generated code; and
- ease of use by programmers not trained in computer science, especially physical scientists and engineers.

The proposed revision is a language almost as large and complex as Ada®. Many hundreds of man-hours have been invested in Ada implementations; yet there are very few (if any) implementations that exhibit both fast compilation and high-quality generated code. There is no reason to expect FORTRAN to be different. Some might respond, "But FORTRAN can benefit from the experience of Ada." As a compiler implementor, I do not believe that claim. Every language has its idiosyncrasies, which it takes many attempts to discover before an implementation really conforms to the standard. Only after they have been discovered can the implementor turn his attention to performance. To do otherwise risks the entire investment in the implementation. To put this simply, "You can't take the shortcut until you know where you

• Ada is a registered trademark of the U.S. Department of Defense (Ada Joint Program Office (AJPO)).



Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/87-11-075-X, S  
Comment #68  
Date: January 22, 1988  
Project: 67-R  
Ref. Doc.:  
Reply to:

80

A  
C  
T  
I  
O  
N  
  
R  
E  
Q  
U  
E  
S  
T  
E  
D

To: X3J3 -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 68

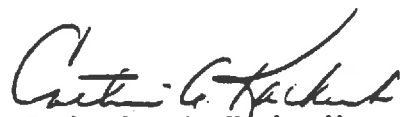
Attached is Comment #68 on BSR X3.9-198x from William C. Leonard of Harris Corporation, Computer Systems Division.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

  
Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: Comment #68

cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

\*Operating under the procedures of The American National Standards Institute.  
X3 Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-2178

80

are going."

As for ease of use, the proposed revision violates this goal also. In fact, any major change to a language, regardless of its nature, makes that language much harder to use by existing users. Moreover, S8.104 defines a language that will be much harder to learn than FORTRAN-77 ever was, not least because it is a hodge-podge of features, many of which do not fit the original language model.

4

Now I will present some specific comments on various aspects of the proposed revision.

**Obsolete, Obsolescent, and Deleted Features** The concept of removing features from an existing language, while laudable in theory, is absolutely unworkable in practice. For example, FORTRAN-77 removed the Hollerith character constant from FORTRAN-66; yet every commercial FORTRAN implementation I know of still implements Hollerith. Why? Because there are still FORTRAN programs in production use that use this feature. Any vendor who expects to be able to remove a language feature quickly finds that the market will not allow it; users expect their programs to work, and continue working, or they find another vendor.

5

One may well ask, "Why don't those users continue to use FORTRAN-66?" Because vendors have not, in the past, been able (for very long) to sustain two FORTRAN compilers. Vendors were thus forced to make their FORTRAN-77 implementations include all facilities from FORTRAN-66 to satisfy their customers. The same will apply to FORTRAN-8x, but I expect the process to be quicker, both because FORTRAN-8x implementations will be much more costly and because vendors have learned their lesson well.

In short, the expectation, built into the proposed revision, that some features of FORTRAN-77 will eventually be removed is unrealistic. Even if they are removed from future revisions of the standard, they will remain in implementations practically forever.

**Array Facilities** I am generally in favor of the array facilities in the proposed revision. However, I think there is danger here, in that such facilities are biased towards vector and parallel processors. Implementations for scalar processors will require much more analysis to generate high-quality code for array facilities. Consider, for example, the following FORTRAN-77 program fragment:

6

```
DO 10 I=1,N
  A (I) = B (I) + C (I)
  IF (I .LE. M) C (I) = D (I) + E (I)
10 CONTINUE
```

The equivalent FORTRAN-8X code, using the array facilities, is:

7

```
A (1:N) = B (1:N) + C (1:N)
C (1: MIN (M,N)) = D (1: MIN (M,N)) + E (1: MIN (M,N))
```

A simple translation of the FORTRAN-8x version for a scalar processor would produce two loops, rather than the one loop that the original version had. The analysis required to produce the first version given the second is far from trivial; I would not expect most implementations to attempt it. This will be even more true of implementations on smaller processors. Users of those implementations will thus be tempted to avoid the array facilities in order to get efficient code.

Conversely, implementors for vector processors will likely be tempted to gradually curtail efforts at "automatic vectorization" as use of the array facilities increases. Users of those implementations will thus be further encouraged to use the array facilities in order to get efficient code.

The result is that the array facilities may very well segregate users into two classes: those with vector processors and those without. This does not promote program portability, especially

80

between vector and scalar processors. In particular danger will be third-party vendors of software written in FORTRAN and targeted for a wide range of machines in both classes.

**Language Size** As I have said, the language proposed in S8.104 is much too large. Implementations of FORTRAN-77 have only recently appeared on the popular micro-computers, although many micros have been on the market for years. The situation with FORTRAN-8x will, I believe, be even worse, since the increase in size is much greater than it was for the transition from FORTRAN-86 to FORTRAN-77.

One might argue that microprocessors are rapidly increasing in speed and capacity, and that that will more than overcome the increase in language size. I do not believe this. A user who buys a new microprocessor, purported to perform twice as fast as the old one, expects his FORTRAN compiler to run twice as fast (or at least significantly faster), not the same speed.

Also, existing FORTRAN-77 users who have no need for the many new features of FORTRAN-8x will be penalized in two ways. First, they will pay more for the FORTRAN-8x processor than for FORTRAN-77. Second, they will pay in increased compilation time. For at least a few years, they will probably also suffer decreased code quality, since FORTRAN-8x processors will be much less mature than existing FORTRAN-77 processors.

Likewise, vendors whose customers have no need for these features will also be penalized. Because many of these vendors sell to the U.S. Government or to government contractors, they are forced to comply with the standard. As I pointed out before, it is economically infeasible to maintain multiple FORTRAN implementations. Essentially, then, the proposed revision would force those vendors to "attack a gnat with a sledgehammer".

**MODULE Facility** I have two primary objections to the proposed MODULE facility: it places a much greater burden on the processor than did FORTRAN-77, and it is incomplete. The proposed MODULE facility requires the processor to maintain, in some unspecified form, a program library. That is, the processor must, while compiling one program unit, be able to access the result of compiling a different program unit. This idea is quite new to FORTRAN (although not to compiler implementors); it is certainly a big step for a language to take.

The requirement of a program library also imposes constraints on the order of compilation of program units. However, the draft standard does not seem to require the processor to enforce these constraints, nor does it specify the behavior of a program in which a MODULE specification is modified and recompiled after a unit that USEs it. As far as I can determine, such a program is standard conforming, yet its behavior is likely to be surprising.

**Justifying the Revision** I do not believe that the new features in the draft proposed revision have been adequately justified. They are, with few exceptions, not needed by anyone modifying an existing FORTRAN-77 program. I can see no reason why someone who is writing a new program and wants the proposed new features should not use Ada instead. Except for array sections, Ada has all those facilities (including array manipulation) and more; in many cases the Ada version is more natural and better integrated into a consistent whole. Furthermore, Ada is by now a more mature and better understood language than FORTRAN-8x. By inventing "yet another Ada", we would merely subject users to another painful and costly learning process for no real gain.

Rather than these many new features, many of which are untried and not really well-understood, what is needed is standardization of existing extensions that are already in wide use. Take, for example, the INCLUDE facility. Many vendors implement such a capability; yet the lack of standardization is a hindrance to portability. Despite repeated claims to the contrary, MODULE/USE in FORTRAN-8x is NOT equivalent nor better than INCLUDE; MODULE/USE requires an entirely different model of programming. It gives no encouragement to users to convert existing programs, yet these are the very programs most in need of portability aids. Let me also answer those who claim INCLUDE is inherently non-portable:

80

among those implementations with INCLUDE, many, many programs have been easily transported. The C language has had an INCLUDE facility for many years, and the X3J11 committee has so far seen no reason to remove it from the proposed ANSI standard for C. There are, of course, differences in file naming between computer systems; yet C users have been very successful in avoiding such problems in transporting programs among various implementations. Besides, file systems themselves are evolving so that arcane limits on the length and content of file names are disappearing. UNIX™ has influenced the market so profoundly that many other computer systems are adopting UNIX-like file naming conventions. Hence, I expect it will become progressively easier for users to choose file names that are widely transportable. FORTRAN-8x should anticipate this and adopt an INCLUDE facility now.

15

NAMelist I/O, which is included in S8.104, is another feature I would support. It is already widely implemented.

The DO WHILE and DO UNTIL constructs, which are not included in S8.104, should also be included in any revision of FORTRAN-77. These also are widely implemented. I note further that the X3J3 committee has already adopted text for inclusion in the Section Notes that admits that these constructs are popular, yet they have not been added to FORTRAN-8x. Why?

16

There are very few other features I believe to be needed in a revision of FORTRAN-77. I believe that, instead of the draft proposed revision, FORTRAN-8x should be a minor revision of FORTRAN-77. It should correct existing errors and inconsistencies and clarify any ambiguities of FORTRAN-77; it should contain an INCLUDE facility and any other *widely-implemented* feature that is consistent with the spirit of FORTRAN. With some reservations, I would also support the inclusion of some array manipulation capability, albeit somewhat less far-reaching than what is proposed in S8.104.

17

Sincerely,



William C. Leonard  
 Harris Corporation, Computer Systems Division

™ UNIX is a trademark of AT&T Bell Laboratories.

1-69

Lawrence Livermore National Laboratory



NOT A PUBLIC REVIEW COMMENT

Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3J3-11 075-X, S  
Comment #69  
Date: January 22, 1988  
Ref. Doc.: 67-R  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT #69

Attached is Comment #69 on BSR X3.9-198x from Jeanne T. Martin of Lawrence Livermore National Laboratory.  
In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.  
If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: Comment #69  
cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

ACTION REQUESTED

P. 554

November 24, 1987

Catherine A. Kachurik  
X3 Secretariat  
CBEMA, Suite 500  
311 First Street, NW  
Washington, D.C. 20001

Dear Cathy:  
I have had an opportunity to look over the new book "Fortran 8x Explained" by Metcalf and Reid. In my opinion the authors have performed a critical service for the Fortran community. As you know, draft proposed standards are not written as tutorials and are frequently difficult to read and understand in a short period of time. The book could contribute to public understanding and reduce the comments based on misinterpretation and confusion.

Both authors have made significant contributions to the work of the committee - writing proposals, working on Subgroups, and offering advice based on their extensive experience. John Reid took on the time-consuming and unrewarding task of Secretary of X3J3 when my employer, after 5 years, could no longer support me in this position. Neither author has complained about paying the service fees; both expend much more money and physical effort on travel to attend X3J3 meetings than is required from US members. I understand that preparation of the book was time consuming and the royalties will provide small returns per hour of effort. The authors were certainly not motivated by profit alone. Many X3J3 members feel that an informed description is sorely needed during the public comment period for the Fortran draft.

\*Operating under the procedures of The American National Standards Institute.  
X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001 2178

Tel. 202/737-6888  
Fax. 202/638-4922

11/24/87 10:45 AM X3J3-11 075-X, S Comment #69 - Lawrence Livermore National Laboratory - PD Box 2200 (Lawrence Livermore National Laboratory) 94550 - Telephone (415) 422-1801 - Fax (415) 422-1148



I do not believe your conclusion was warranted by the actual circumstances.  
Sincerely,

Jeanne T. Martin  
L-300  
Lawrence Livermore Nat. Lab.  
Livermore, CA 94550  
USA

cc: Fran Schrotter  
Bob Kearney --  
Bob Follett

80

# 70



NOT A PUBLIC COMMENT

Catherine Katchurik  
X3 Secretariat/CBEMA

13.11.87

TEKLA OY is a Finnish software company, the majority of program code written in Fortran. Since 1966 we have been quite deeply involved with Fortran and naturally we are very interested in the development of the new standard on that language. As it appears that the proposed, revised Fortran 8X standard has now been announced for public review, we would very much like to purchase the document also for our company. However, we don't know the connections through which to order the standard. Therefore, could You kindly send to us the addresses (and telefax numbers) of those authorities (or companies) from which we could get that document (=Global Engineering Documents ?). We would also like to know the price and the payment procedure, i.e. if a cheque is needed in advance (banking connections). If it's impossible to order the draft directly from the U.S., we would appreciate the ANSI number of the standard so that we could try to order it through the Finnish Standardization Association.

Thank You.

On behalf of TEKLA OY

  
Tero Tommila

Our address: Koronakatu 1  
02210 Espoo  
Finland

Our telefax number: + 358 0 8039489

X3 SECRETARIAT  
'87 NOV 23 AM 12:25

p. 556

# 71

80

1500 Johnson Drive  
Room 1008, CAE  
Madison, WI 53706

X3 Secretariat/CBEMA  
311 First Street, N.W.  
Suite 500  
Washington, DC 20001-2178

This is a public review comment on the proposed Fortran standard.

71.1

Since you changed the statement in section 12.2.5.2 of the Fortran 77 standard from:

Reading and writing records [for internal files] is accomplished only by sequential access formatted input/output statements that do not specify list-directed formatting.

to the following in section 9.2.2.2 of the proposed standard:

Reading and writing records [for internal files] must be accomplished only by sequential access formatted input/output statements that do not specify namelist formatting.

I assume that you intend to permit list-directed formatting on internal files. For this I am thankful, since a list-directed READ on a piece of a command line is an easy way to decode a variable length numeric token.

71.2

Here is yet another proposal for putting pointers into the language: A name declared with the ALIAS attribute is logically equivalent to a pointer. Let the number of times the ALIAS attribute is declared for the same name specify the level of the pointer. When the name is used outside of an IDENTIFY statement, only the object ultimately pointed to is used or changed. In the IDENTIFY statement, the two sides must have the same type and rank, and the right hand side cannot have more than one level of ALIAS less than the right hand side. (If the right hand side has extra ALIAS levels, one follows the pointers to find the object that the left hand side will point to.) To pass a pointer value to a subprogram, one must make the ALIAS name a component of a derived data type, and pass a name of that type.

Sincerely Yours,

*David L. Wilson*  
David L. Wilson

CC: Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY 10018

X3 88 JAN 14 P 2:57

Review Letter # 71			
Subgroup Nominations :			
1	CIO	11	
2	DATA	13	
3		13	
4		14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	





University of San Francisco

San Francisco, CA 94117, URS

Department of Computer Science  
 Bureau Science Center (413) 666-6330

19 December 1987

Accredited Standards Committee  
 X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/87-11-075-X, S  
 Comment #72  
 Date: January 22, 1988  
 Project: 67-R  
 Ref. Doc.:  
 Reply to:

To: X3J3 -- FOR ACTION  
 Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 72

Attached is Comment #72 on BSR X3.9-198x from Loren F. Helmsner, Ph.D. of University of San Francisco, Department of Computer Science.  
 In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
 Catherine A. Kachurik  
 Administrative Secretary, X3

Attachment: Comment #72  
 cc Jeanne Adams, X3J3 Chair  
 Lloyd Campbell, Technical Editor

\*Operating under the procedures of The American National Standards Institute.  
 X3 Secretariat: Computer and Business Equipment Manufacturers Association  
 3111 Hill Street, N.W., Suite 600, Washington, DC 20001-2178

ACTION REQUESTED

P. 558

ANSI X3 Secretariat  
 CBEMA, Suite 500  
 311 First Street NW  
 Washington DC 20001

ATTN: Public Review of dpr ANS X3.9-198x Fortran  
 I have reviewed the draft Proposal document, and I have the following comments:

1. The Type Extensions in Appendix F, Section F.1, relating to BIT data type, should be reinstated as part of the full language.

A BIT facility in some form is commonly available as an extension to Fortran 77 in almost all existing implementations, because such a facility is needed in many Fortran applications. Examples of use include graph theory, bit-mapped graphics, and data transmission.

The same functionality is available in most other languages in one way or another, for example in the form of "bits" in Pascal. Bit handling is not a novel or untested area of programming language design.

A standard bit handling facility in Fortran is especially needed, since otherwise the current anarchy will continue, with consequent lack of portability. Apparently X3J3 has spent considerable time working out the best possible way to conform such a feature to the other parts of the proposed language; so it may be assumed that the description in Appendix F gives the most reasonable way to introduce such a feature into Fortran in a standard manner.

2. The feature "Significant Blanks in Free Source Form", described in Appendix F, Section F.3, should be included as part of the full language.

This feature has some impact in simplifying lexical analysis, by making it possible to more easily recognize the boundaries between tokens.

Significant blanks also contribute to the desirable goal of eventually providing tools for the analysis of program source text that will work with more than one language. For example, one would like to have a word processor tool (text editor) that could globally change occurrences of "x" in a text file to "y", but only when "x" appears as a data object identifier (and not a comment, for example); and such that the same tool could work for several languages including Fortran, Pascal, Ada (M), etc. Although other features of the current languages (as well as of the proposed Fortran 8X) preclude the possibility of such tools at present, now is the only foreseeable opportunity to make this change that will be necessary (even if not sufficient) for reaching the eventual goal.

Tel: 202/721-6888  
 Fax: 202/658-4922

80

72.3

3. A pointer facility should be added to Fortran along with the other features in the draft proposal.

I have studied recent working papers prepared by members of XJJ, and it appears to me that a pointer facility could be added as a relatively minor extension to the existing ALLOCATE and ALIAS features. This facility would work in much the same way as the pointer facility in Pascal, which is useful for creating linked data structures, binary trees, etc.

Pointers are provided in all modern programming languages, with a syntax not too different from that recently suggested. It would be unfortunate to add user-defined derived (record) types to Fortran without any way to link them together into data structures such as linked lists.

SUMMARY:

I am greatly impressed by the obviously tremendous amount of effort that has gone into the development of this draft proposal. I believe that the new proposed language will be accepted by the Fortran user community. I also believe that, given current trends in hardware development, the new language will not be "too large" for implementation on even the smallest scientific computers that will be available in one or two years. By careful measures, the proposed language (even with the additions that I have endorsed above) is smaller than Ada [TM], and it has the advantage that at every step the developers of Fortran III have kept clearly in mind the possibility of efficient implementation "in the spirit of Fortran".

I hope that my comments will prove to be of value in shaping the further processing of the proposed revised Standard language, and in the general future development of the language.

Sincerely,

*Loren P. Meisner*  
Loren P. Meisner, Ph.D.  
Professor of Computer Science

72.4

Review Letter # 72	
Subgroup Nominations :	
1	DATA 11
2	GEN 13
3	DATA 13
4	GEN 14
5	15
6	16
7	17
8	18
9	19
10	20

Subgroup Nominations for Public Review Letter #73

80

1. GEN	26. DATA	51. PROC
2. DATA	27. DATA	52. PROC
3. DATA	28. ED	53. PROC
4. DATA	29. DATA	54. PROC
5. GEN	30. DATA	55. PROC
6. ED	31. DATA	56. PROC
7. ED	32. DATA	57. PROC
8. ED	33. DATA	58. PROC
9. ED	34. DATA	59. PROC
10. ED	35. DATA	60. PROC
11. DATA	36. CIO	61. PROC
12. DATA	37. DATA	62. PROC
13. GEN	38. DATA	63. PROC
14. GEN	39. DATA	64. PROC
15. GEN	40. DATA	65. PROC
16. GEN	41. DATA	66. PROC
17. GEN	42. DATA	
18. ED	43. GEN	
19. ED	44. GEN	
20. ED	45. GEN	
21. GEN	46. GEN	
22. GEN	47. CIO	
23. GEN	48. CIO	
24. DATA	49. CIO	
25. DATA	50. CIO	

redlined Standards Committee  
INFORMATION PROCESSING SYSTEMS'

Doc. No. X3J37-11-075-X, S

Comment 873

Date: January 23, 1988

Project: 67-R

Ref. Doc.:

Reply to:

To: X3J3 -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC  
REVIEW COMMENT 873

Attached is Comment 873 on BSR X3.9-198x from Steven O. Siegfried.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: Comment 873

cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

acting under the procedure of The American National Standards Institute.  
Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-3178

Tel: 202/337-8088  
Fax: 202/638-4822

11/73

Public comment of Steven O. Siegfried on X3J3/58

Date: January 12, 1988

To : X3 Secretariat, CBEMA  
311 First Street, N.W.  
Suite 500  
Washington, D.C.  
20001-3178

From: Steven O. Siegfried  
1607 Stanford Avenue  
Saint Paul, Minnesota 55105

Re : A comment on the new X3J3 dp Fortran standard

Size,

The X3J3 committee is charged with developing a new FORTRAN standard.

Some of the normally considered aspects when extending a language are:

- \* Is this particular extension common in industry usage?
- \* Is there a demand from the user community for this extension?
- \* Does this particular extension
  - duplicate existing function?
  - make the most general change in the most minimal manner?
- \* Does this particular extension create a cost benefit for:
  - the users of the language, or
  - the suppliers of language processors?
- \* Do the various extensions fit together to form a cohesive language?
- \* Are user programs that use these extensions made:
  - easier to write, maintain and understand, and
  - more efficient?
- \* Do these extensions advance the "state of the art"?
- \* Does the standard specify a language that is portable across a number of implementations?
- \* And finally,
  - Does the finished product look enough like the original language to warrant calling it by the same name?

As a FORTRAN compiler writer for the company that pays my way in life (I'm the guy who's supposed to implement BXL, as a FORTRAN programmer since the late '60's, and as a language designer currently working on the X3J9 and X3J3 (Extended Pascal) standards, it's my opinion the committee has failed in the worst possible way to discharge its duty. I strongly urge this language not be adopted as a FORTRAN standard.

A number of serious technical flaws exist in the draft. The language proposed for standardization by the draft is clearly flawed and does not, in my opinion, deserve serious consideration as a standard for FORTRAN.

I really wouldn't mind so much if you called it FORBOL or something else, but please... don't call it FORTRAN, because it isn't FORTRAN anymore. Some of these features don't belong in FORTRAN, others don't fit well with the existing FORTRAN, others aren't general enough to be usable, while still other common vendor extensions to FORTRAN 77 are

-- 1 --

January 12, 1988

80

missing altogether. The language is so large that I fear we'll never see microcomputer versions (even reasonable subsets) for machines with less than a megabyte of memory. It would appear that a very long shopping list of features to be added was compiled by the committee, then every third item was accepted without regard to normal language extension practices, de facto vendor extensions, or the existing language. FORTRANX seems to have ended up as some sort of Frankenstein of marginal features. Nothing fits together well, the stitches show, and, when taken as a whole, some anti-social personality problems exist. Well Doctor, you've got us local peasants reaching for our pitchforks and torches.

FORTRAN is a scientific language, a lingua franca used by scientists the world over. The "creeping elegance" you've introduced won't add any new tools to the quantum mechanic's toolbox, he's too busy to even read your catalog, let alone subscribe to your theories of programming. You've misread your target audience. Don't turn FORTRAN into programming's Esperanto.

I am not a lawyer, and don't pretend to offer legal advice. However, it seems to me that any change of this magnitude will bring out the lawyers. Need I remind the committee of just how many million dollar codes are out there in the real world being maintained by folks having no control over their compilers? Need I remind the committee of just how much protection an X333 membership brings each of you when the owners of these codes find out they don't work anymore, or worse yet, give answers that cost lives? Since, by definition, each of you is an "individual expert" in your field, representing an employer with deep pockets might not be enough. As I told a member of your committee a year or so back, my advice to you is "vote no, and stock up on liability insurance".

Attached, please find comments on specific areas within the dPANS as well as general comments on the language taken as a whole. These are not as fully fleshed out as I would hope, it's too depressing to work on that hard. In as much as a good deal of my future rides on your decisions with respect to FORTRAN, I would ask for a timely reply to my comments.

The opinions and views expressed herein are my own. In voicing these opinions and views, I have not asked for, nor have I received, an official okay-dokay from any organization of which I am a member or employee.

Thinking-about-joining-X3-just-tj-vote-no'ly yours,

General Comments on what's Missing from the Draft

Subset FORTRAN

Since the language is so large, where is the subset definition?

Bit Datatype and Operators

Virtually every FORTRAN compiler I've seen in the last 5 years has had the LOGICAL, BOOLEAN or some such type that allowed bit-wise AND, OR, NOT, XOR, and NAND as well as INTERIOR operations. So how come you're ignoring common practice?

Pointers

FORTRAN has to be the only commonly used algorithmic language left that doesn't have explicit pointers. Many vendors supply a FORTRAN compiler that contains pointers with explicit or implicit dereferencing. Having used a FORTRAN with implicit dereferencing, I strongly suggest explicit references like "A -> B" be included, mainly because implicit dereferencing isn't as useful (it's tough to build a binary tree with implicit dereferencing). Contrary to what most FORTRAN programmers think, the study of data structures has come a long way since FORTRAN-IV and this type of feature is badly needed.

I don't know if I can declare fields in derived types as ALLOCATABLE, since I can't find the section on the ALLOCATABLE attribute. If I can, then how do I access the value in a single expression of the second element in a linked list of integers without something like ANCHOR->NODE->NODE->NODE? If your answer is something like ANCHOR->NODE->NODE, my eyes glaze over.

Non-ASCII Character Sets

How about adding support for non-english character sets, like kanji? I might even be able to sell supercomputers in Tokyo if you did.

Comments on the Text of the Draft

Editorial

Why have you chosen to separate the syntax into different subsections from the semantics for most constructs? This is confusing and error prone for both the authors and readers of the dpANS. The subsections on Expressions and Assignment are truly horrid and should be rewritten from the ground up.

73.6

Editorial

By my understanding, forewords, notes, examples, and appendices are not officially part of an ANSI standard. A paragraph is needed in 1.5 detailing this. Also by this reasoning, a large chunk of the appendices properly belong in the specification itself.

73.7

All references to the appendices in the body of the standard (the specification) should appear in notes. For an example of where such a "bad" reference is made, see paragraph 1 of 1.6. In all cases, these references should be made into notes. Also, I would recommend a special type font for notes and examples be used to make this distinction more clear.

73.8

73.9

Editorial

Any standard should be precisely worded. This one falls far short of a precise definition of it's requirements. Requirements of the form "The mumble shall dither" should be used in all places where something is required. "Must" and "is" are also acceptable. Avoid using terms like should, will, can, might and may. At all costs (remember you have to enter a maintenance phase too!), don't use terms like inferred, recommended, assumed, and derived. Terms like this leave too much room for interpretations different than those the committee had in mind.

73.10

Editorial

The syntax seems more cluttered than it has to be, remember most FORTRAN programmers can't touch type. For example:  
PROGRAM N, "END IF", "END CASE", "END DO", "END DATA", "END PROGRAM", and "END SUBROUTINE" Don't you expect programmers to be able to tell which END matches which construct, or have you so totally hoed the syntax that this is actually required? In other programming languages that use only a single END, these extra noise words are normally used by programmers, where needed for clarity, in comments directly after the END.

73.13

73.14

1.4 Conformance

The conformance clause contains some rather large holes.

Paragraph 1 states that "The requirements, prohibitions, and options specified in this standard refer primarily to permissible forms and relationships for a standard-conforming-program rather than for a processor. ... The requirements, prohibitions, and options for a standard-conforming processor usually must be inferred from those given for programs." What are your standardizing here? It should be processors, not programs. The wording of this clause alone is enough to invalidate the entire standard. How does a vendor validate a processor for a standard that specifies programs it should accept (an infinite set), and not the processor itself? Wrong, Wrong, Wrong!

73.15

73.16

73.17

Paragraph 5 allows a standard conforming processor to not detect conflicts with locally defined intrinsics. This vastly hampers program portability and should be fixed.

Paragraph 7 defines what constitutes an "extension" to the standard with an example. How about a more standard-like statement of what an extension to the standard is, along with a statement of where extensions stand with regard to the core language?

73.18

Paragraph 8 properly belongs in 1.3 (Scope), not here.

80

Public comment of Steven O. Siegfried on XJ3/58

1.5.1 Syntax Rules

This is the strangest BNF I've seen in some time. Is there some valid reason for departing from more traditional BNF forms like that below?

expression ::= term ( "|" term ) .

term ::= factor | factor | .

factor ::= non-terminal | terminal | "-" expression "-" | "-" expression "-" | "-" expression "-" | .

rule ::= non-terminal ":" ":" | expression | "." .

language ::= rule | rule | .

73.19

1.5.2 Assumed Syntax Rules

Wrong, Wrong! If for some semantic reason, you need a syntax rule on the order of

BY-name is name.

by all means insert it at that point in the BNF, then add some verbiage nailing the semantics to the syntax. Likewise, lists should be explicitly stated as lists, not assumed, since all lists also have semantic actions associated with them, even if it's nothing more than a one to one ordered mapping. In the real world, people get assumptions wrong all together too often, so don't assume anything!

73.20

1.6 Deleted, Obsolescent and Deprecated

Just because you good folks think (probably rightly) that features of older FORTRAN standards are out of date, don't for a minute believe that the rest of the world will run out and change their code. Old programs that use these features will continue to be worked on by users and compiled with new FORTRAN compilers for some time to come. Vendors will continue to support these features regardless of their status in the latest FORTRAN standard. Standardized features, once standardized should stay that way. A standard still exists for punched cards and for labeling features "deleted" or "obsolescent" opens holes for vendors of low quality goods to break with established common practice, exactly what standards embody. As for "deprecated", well I don't know about you good folks, but when I arise in the morning, I shave, I shower and deprecate. Don't deprecate on FORTRAN.

73.21

Public comment of Steven O. Siegfried on XJ3/58

1.2.5 Statement Labels

Can we at least bring the major control flow hinge-point into the late 1960's by allowing alphanumeric label names?

73.22

1.3 Source Form

Other than changing things for the sake of changing them, is there any reason for such an obtuse source form? Fixed source form served the user community not wisely but too well for too long to change now. The new form is clearly a wart that should be removed.

73.23

4.3.1.2 Real and Double Precision Real Types

Since you've gone to all of the trouble to add precision-type-parameters and exponent-range-type-parameters, and since you've bitten the bullet on deleted and obsolescent features, what possessed you not to get rid of DOUBLE PRECISION? DOUBLE PRECISION is one of the most non-portable features I've run across in any language. The 64 bit single precision real I use at work corresponds to quad precision on my 6 bit micro, yet all I really want for most applications is mumble digits of precision. Either dump variable precision arithmetic altogether, or dump DOUBLE.

73.24

73.25

73.26

1.4.1 Derived-type Definition

What I can write in Pascal and

```

TYPE
  Movie_queue = ARRAY[1..10] OF RECORD
    age : INTEGER;
    name : PACKED ARRAY[1..30] OF CHAR;
  END;

```

take 2 type statements in FORTRAN. Nested declarations of types have been around since the early to middle 1960's. Why didn't the existing practice in this area prevail, rather than re-inventing TYPE?

80

P. 564

5.1 Type Declaration Statement

What prompted the double colon before the entity-decl-list in a type-declaration-stmt? A single colon is enough, double colons will lead to programmers having overdeveloped pinkies on their right hand and spots before their eyes. This is an uglification and should be changed.

Where are variants in derived types? Without a real variant in a record, I end up wasting altogether too much storage in order to express the information of a variant nature.

This entire section is confusing in organization. Does the subsectioning follow the syntax of 5.1? What makes the stuff in 5.1 any different from that in 5.2? Where is the subsection on ALLOCATABLE?

5.1.2.1 Value Attributes

A perfectly good syntax exists for specification of values, that being DATA. Why go to the trouble of defining a different syntax for this when something on the order of that below doesn't?

```
REAL, PARAMETER :: ONE DATA(1), Y DATA(0.1/3.0)
INTEGER, DATA :: NEXT, DATA(1)
INTEGER ARRAY(3), PARAMETER :: ORDER DATA(1,2,3)
```

5.1.2.1.1 Parameter Attributes

I've never liked the PARAMETER stuff. PARAMETERS are parameters to the compiler, not to the program unit being compiled. In the program unit being compiled, the PARAMETERS are constants. Why not spell PARAMETER as CONSTANT, like the rest of the world?

5.1.2.2 Accessibility Attributes

Can I specify a variable as PUBLIC with a derived PRIVATE type? If so, what is the effect?

5.1.2.4 Array Attribute

Why is the maximum dimensionality of an array limited to 7? This seems arbitrary.

5.1.2.8 Range Attribute

In the rest of the world, "range" talks about the limiting values a variable or set of variables can assume, not about the shape of the variable. Why not spell this as "SHARP"?

5.2.2 Optional Statement

From reading 12.5.2.6, the OPTIONAL statement looks like dead weight. Why not make all parameters optional, and then delete the OPTIONAL statement?

5.2.4 Save Statement

In light of recursion, valid uses for SAVE dwindle. Why not spell it "COMMON, PRIVATE" and get rid of it, since isn't that what it boils down to anyway?

5.4 Hamlet statement.

Finally!

5.5.1 Equivalence Statement

If you need to add a method of type escape to allow viewing a variable as some other type, create a type escape statement. If you need to economize on data storage, use some sort of dynamic heap (ALLOCATABLE?). If you need to drive programmers and compiler writers nuts and add untold faults to correct programs, use EQUIVALENCE.

6.1.2 Structures Components

Most common languages that have records, dereference fields of them by using a ".", not a "->". The more common approach for your examples is: SCALAR\_PARENT(SCALAR\_FIELD ARRAY\_PARENT(1:10), SCALAR\_FIELD This is truly an uglification and should be fixed.

6.2.2 Allocate Statement

It would appear that if I want to allocate a scalar derived-type I need to access the sucker as if it was an array. Truly weird and totally wrong, fix this.

73.34

73.35

73.36

73.37

73.38

73.39

73.11, cont.

73.27

73.28

73.29

73.30

73.31

73.32

73.33

50



73.40

5.2 Array Processing

ly and large, I applauded the attempt at vector-syntax in FORTRAN 9X. I should, however, remind the committee that FORTRAN stands for FORNULA TRANSMISSION. It causes my blood pressure to rise to see that:

```
DO 10 I=2,M
```

```
10 A(I) = A(I-1)*M(I)
```

does not mean the same as

```
A(2:M) = A(1:(M-1))*M
```

When translating back of the envelope code, when "hand tightening" critical sections of big codes, and when teaching the natural tendency is to think these two fragments are the same. You've introduced a wonderfully devious means of making hard to find mistakes by insisting the entire right hand side be evaluated prior to assignment. This decision should be changed so, that under assignment, the above two fragments mean the same thing.

Since array processing is spread all over the draft, I wonder how many other "features" lie hidden to rise up an bite implementors and users.

73.41

5.2.3 Deallocate Statement

I'm confused here. Do you want the compiler to generate code for deallocation of all allocated variables at any RETURN or END, or do you mean that all allocated variables become unaccessible at any RETURN or END?

73.42

5.2.5 Identify Statement

As if the static EQUIVALENCE statement wasn't bad enough, we now have a dynamic version, with a different spelling, that allows changing the name, shape, and the type of an object in one fell swoop. As far as program correctness goes, you've just taken the gun from the hands of the programmer that EQUIVALENCE gave him and replaced it with a cannon. Now the programmer can shoot even bigger holes in his code. Great... delete it.

73.43

7.1.1 Evaluation of Operations

The restriction on changing "I" in

```
A(I) = F(I)
```

```
Y = G(I) + I
```

is overly restrictive. While it is true that it may be hard to optimize thru expression like these, this isn't a legitimate reason to so severely restrict the use of a function that alters its arguments. The argument that functions should not alter their arguments doesn't wash either, since your job is to define a language, not enforce programming practices.

73.44

7.1.1.1 Evaluation of Operands

While the examples and text given in this subsection are valuable, I can't find any unambiguous statement about order of evaluation of the operands of expressions. You need something like:

The order of evaluation of operands of dyadic operators is implementation defined. Thus, in some cases the construct

```
A operation B
```

may evaluate (depending on the specific operation) only A or B, A then B, B then A, or A and B in parallel. Further, when A and B are not scalar, individual elements of A and B may be evaluated in any order.

You also require a similar statement on associativity and commutivity, since the examples aren't exhaustive.

73.45

7.2.1.2 Complex Exponentiation

Why do you allow this?

73.46

7.5 and 7.5.1 Assignment and the Assignment Statement

Great, assignment causes a variable to become defined or redefined. To what? I have to assume this means to the value of the expression that appears on the right of the "=", but I could also assume that the variable takes the value of 0, or that of the real-time clock, or anything else. You need a more precise definition of assignment than the one that appears.

Basically what I'm getting at here is what does definition of a variable mean? Once defined, do variables stay that way or do they decay until they must be redefined? Be explicit!

8.1.2.1 Form of the If Construct

Why isn't optional syntax provided to allow THEN and ELSE to appear on the next line in an IF statement (without a continuation)? In order to better preserve the nesting of if statements, I'd like to be able to write stuff like:

```
IF (EXPRESSION)
THEN
IF (OTHER_EXPRESSION)
THEN A = B
ELSE B = A
STOP
ENDIF
ELSE
IF (OTHER_EXPRESSION)
THEN C = D
ENDIF
```

! Need IF (X) THEN on 1 line

```
IF (OTHER_EXPRESSION)
THEN C = D
ENDIF
```

but it appears that I'm not allowed to. This seems a really arbitrary restriction.

8.1.3 Case Construct

Nice try, but too much noise in the syntax. First, the DEFAULT case should be restricted to be only the last one, then clean the syntax up to where the first example in 8.1.3.3 looks like:

```
INTEGER FUNCTION SIGNUM(N)
SELECT CASE N
...1: SIGNUM = -1
0 : SIGNUM = 0
1...: SIGNUM = 1
END
END
```

8.1.4 Iteration Control

I actually like DO (N) TIMES, but why not also add something like:

```
I = 1
DO UNTIL (I=500)
I = I + 1 or I = I + 1
UNTIL (I=500)
END DO
as well as something like:
DO WHILE (I>=500)
I = I - 1
END DO
```

These two additional constructs would greatly expand on the ease of implementation of many algorithms.

On a different but related topic, I can't seem to find any prohibition against changing the value of the variable of iteration while executing the DO statement. Stuff like:

```
DO I=1,10
I = I + 1 ! I is redefined by assignment
J = F(I) ! function F changes I
CALL G(I) ! Subroutine G changes I
READ(10),I ! I/O causes I to be redefined
END DO
```

As a suggested fix to this, see the Pascal standard's treatment of "threats".

11 Program Units

CONTAINS isn't necessary, since appearance of a SUBROUTINE before the outer routine's END clearly indicates a contained subroutine.

What is the effect of trying to compile and execute the following two modules?

```
MODULE A
USE B
PARAMETER (AI = BI)
```

```
MODULE B
USE A
PARAMETER (BI=AI)
```

How do I interpret what these two modules mean? How do I write a compiler which generates code for these two modules? Why didn't you try and prevent this kind of mutual supplies relationship between A and B? In the end, a statement of the form: "A module must not supply itself" must be included. This is an obvious flaw in the program units extensions, and leads me to wonder how many more are hidden away in the dpANS.

With advent of MODULE and USE, haven't you implicitly required: a non-flat external name space for use by the link-editor, and some sort of implicit interface management system that outlives invocations of the processor? Didn't you exceed the scope of the standard in doing this? Haven't you, for instance, forced vendors to supply new link-editors and interface management systems with their FORTRAN compilers? Without the specification of these, the dpANS is fatally flawed.

73.47

73.48

73.49

73.50

73.51

73.52

73.53

80

13 Intrinsic Procedures

FORTRAN 77 was bad enough with all the intrinsics that it had, but FORTRAN 8X, well I'm staggered. On top of all the new needless functions, we have stuff like CHPLN(SOURCE, MOLD - M, X-TARGET). Yuch! The entire section should be rethought and rewritten using a zero-based criteria for addition.

Most of 13.9.4 is fluff and should be deleted. ASCII collating sequence character string comparison functions must drive IBM, the Univac half of Unisys, and Control Data nuts. ADJUSTL, ADJUSTR, are excellent examples of stuff that doesn't belong in a language standard. Why is a CHR function supported, when virtually every other conversion in the language is implicit?

Everything in 13.9.6 should be "global parameters", i.e. constants, defined by the implementation, not wasteful runtime functions.

13.9.8's NEAREST, RASPCING, and SPACING sound like after the fact computational error detection. As an implementor, what should I do with EXPONENT, FRACTION, SCALS, and SETXPOSITION if my hardware doesn't support IEEE real, but something like very-long fixed point instead?

With the advent of operation under mask (WHERE), why do the functions in 13.9.10 require a mask? PRODUCT and SUM are inherently error prone without an explicit ordering of evaluation.

In the presence of EQUIVALENCE and IDENTIFY, I would venture to say that, as sure as God made little green apples, most of the routines in 13.9.11 will not behave consistently across more than 1 implementation, let alone correctly.

The functions in 13.9.12 are truly dangerous in what they allow. Has anyone actually tried using them or implementing them?

Most of 13.9.9, 13.9.13, and 13.9.14 is fluff and should be deleted.

In 13.10, what happens if my implementation doesn't support a real time clock?

73.54

Are you sure that RECURSIVE shouldn't be deleted and a change to variable declarations made that insist the non-SAVE variables declared in subroutines or functions are "automatic"? As an implementor, RECURSIVE usually doesn't tell me anything meaningful. Erring on the side of caution should have you consider replacing RECURSIVE with STATIC, thereby telling the compiler that this routine can/should receive static storage.

73.55

Other than making non-touch typist's jobs easier, what justification is put forward for ASSIGNMENT? This seems a wart and should be removed.

73.56

73.57

73.58

73.59

73.60

73.61

73.62

73.63

73.64

73.65

73.66

80

---

Public comment of Steven O. Siegfried on X133/58

Conclusion

- 1) This isn't FORTRAN, but something much larger that lacks a sense of wholeness.
- 2) Whatever it is isn't clearly defined.
- 3) This shouldn't become the next FORTRAN standard.

January 12, 1980

-- 16 --

p. 569

80

# 74

SD

DECUS Meeting  
Dallas, Texas

December 15, 1987  
88 JAN 19 11:57

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

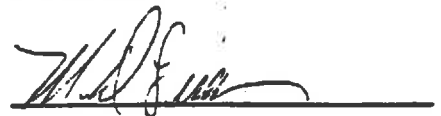
While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- 74.1 • DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient.
- 74.2 • The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice.
- 74.3 • The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy.
- 74.4 • The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive.

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,



M.D. SISSON  
Company: LTV MISSILES & ELECTRONICS GROUP  
Address: P.O. Box 650003, MSE-79  
DALLAS, TX. 75265-0003

Review Letter # 74			
Subgroup Nominations :			
1	DATA	11	
2	PROC	13	
3	DATA	13	
4	GEN	14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	

credited Standards Committee  
INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/87-11-075-X, S  
Comment #75  
Date: January 22, 1988  
Project: 67-R  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 75

Attached is Comment #75 on BSR X3.9-198x from Bob Allison of Microsoft.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

Attachment: Comment #75

cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

P. 571

Operating under the procedures of The American National Standards Institute.  
Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-2178

Tel 202-731-8888  
Fax 202-638-4972

#75

January 11, 1988

Bob Allison  
c/o Microsoft  
Box 97017  
Redmond, WA 98073-9717

Catherine Katchurik  
X3 Secretariat/CHEMA  
311 First Street NW, Suite 500  
Washington, DC 20001-2178

Attn: X3J3 Public Review Comments

1) I have been attempting to implement the CASE construct described in the draft standard and have discovered some annoying features which I hope can be corrected. The construct's syntax is misleading. Using the keyword CASE in both the selection and the case portion of the CASE construct looks ambiguous. I suggest that the keyword CASE be omitted from select-case-statement because it is unnecessary and confusing.

2) Also, allowing overlapping case value ranges in the same selector seems unusual. The intuitive way of viewing the case value range list is to consider it a set of case value ranges which happen to select the same block of code. Allowing overlapping ranges might be interesting if we were able to specify something like selection of all numbers divisible by two and all numbers divisible by three, but since the case values must be constants with their ranges specified I don't understand the benefit in allowing overlapping values.

3) It is implicitly stated that selection is allowed on character expressions with length greater than one. I think this is ill-advised. It is fairly easy to construct a case which cannot be resolved. For instance, let the expression contain a CHARACTER dummy argument concatenated to some character expression. Then the dummy argument degrades to an if-then-else condition since I can't think of a machine implementation which allows case selection on non-integer entities. We will need to store the result of this expression some place so we can compare against it for each case value. But we can't know how big the item is to reserve space for the expression result. The best that could be done would be to allocate a temporary which is as large as the largest possible character string (DEC, for instance, allows 64K for their VAX/VMS implementation) and assign the expression into it.

So, the case construct is very difficult to implement using a static model. At the very least I would suggest constraining the character expression to be a variable, since the variable's size is known and storage already allocated. Actually, I doubt multi-byte character cases are terribly useful and think constraining them to length one is reasonable.

4) I think allowing logical expressions for a case construct is frivolous.

15.1

15.2

15.3

15.4

80

75.5 3) The description of the DO construct allows some for pretty unusual DO loops. For instance:

```
named_loop: DO 10
END DO named_loop
```

which seems a bit over-zealous, but there are also cases which are more serious. For instance:

```
DO 10
DO
END DO
```

where the END DO statement terminates both loops. This example is not a serious problem, except that once END DO statements can terminate more than one loop it allows the next example:

```
DO
DO 10
END DO
```

If this case is standard-conforming, and I see no reason in the draft to indicate that it is not, then this is very ambiguous, since an END DO statement may follow, in which case the END DO statement above only terminates the inner loop, or it may not, in which case the END DO statement above terminates both loops.

This interpretation is actually reinforced by the statement that EXIT and CYCLE statements belong to a specific DO construct, whereas no mention is made of END DO statements.

If this is not the desired syntax, then I suggest that constraints be added allowing a label or a do-construct-name, but not both, and requiring each DO construct which does not specify a label have an END DO statement which belongs to that DO construct.

Review Letter # 75	
Subgroup Nominations:	
1	CIO 11
2	CIO 13
3	CIO 13
4	CIO 14
5	CIO 15
6	CIO 16
7	GEN 17
8	18
9	19
10	20

75.6

6) I'm certain this has been considered before, but since I do not know the rationale behind it I want to comment on the syntax of construct names. I think the concept of naming constructs is very good, but the particular syntax arrived upon is poor. It is significant, for instance, that everywhere a sub-program or type or module is declared in the draft standard, the END statement is elaborated, except for page 12-8, line 7 which should be "END FUNCTION SCALE", but I could not find a single use in an example of a construct name. In addition, the WHERE construct does not even allow construct names. I believe it is because they do not look natural. It is very natural to say:

```
SUBROUTINE SUB1
END SUBROUTINE SUB
```

because the keywords appear first. It is not nearly so elegant to say:

```
INVERSION_LOOP : DO 10, i-1,1000
END DO INVERSION_LOOP
```

because you cannot line up the construct names conveniently, and you cannot find the DO loop easily because it is obscured by the construct name. Indentation cannot be lined up conveniently either. In addition, they look suspiciously like symbolic labels, which they are not.

I can only recommend that the construct names be moved to after the significant keyword somehow. For instance, don't allow construct names on the old DO syntax, and allow:

```
DO :INVERSION_LOOP
END DO INVERSION_LOOP
```

or

```
SELECT :MARGIN (i)
END SELECT MARGIN
```

That way they look more like the other named items.

75.7

7) As a final observation I want to note that every time I have tried to implement a feature out of the draft I have discovered ambiguous text or features that are impossible to implement. So far I have only tried to implement features which are well known in other languages, such as case selection. I am worried about the specifications of constructs which are familiar to most of us. I am very skeptical as to the quality of the definitions of constructs which are experimental in nature.

I do not believe that a standards committee is the correct place to design new language features, especially since there is no method for testing or verification, which any competent programmer will assert is required for a successful product. I believe the quality of the draft is too poor to implement a language from and strongly suggest that in the absence of an existing implementation each specification be scrutinized in greater detail.



Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X1/87-11-075-X, S  
Comment #76  
February 22, 1988  
Project: 67-R  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 76

Attached is Comment #76 on BSR X3.9-198x from Stuart Rones of AT&T Technology Systems.  
In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.  
If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentators of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentator is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentator must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Karcholik*  
Catherine A. Karcholik  
Administrative Secretary, X3

Attachment: Comment #76  
cc: Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

\*Operating under the procedures of The American National Standards Institute  
X3 Secretariat Computer and Business Equipment Manufacturers Association  
3115 14th Street, N.W., Suite 500, Washington, DC 20001 2178

1-1 202/737 8888  
4-1 202/638 4972

ACTION REQUESTED

p. 573

# .6



2000 Spring Drive  
Menlo Park, CA 94025  
214 288 2000

88 JUN 19 AM 5:1

January 12, 1988

SUBGROUP	
1	GEN
2	DATA
3	PDoc
4	DATA
5	GEN
6	GEN

X3J3 Chair  
X3 Secretariat  
111 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8X standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- o DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient.
- o The new source manipulation capabilities (MODUL/USB) are more powerful than necessary, are too complex, and are untested in practice.
- o The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy.
- o The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing functions such as the COMMON, DIMENSION and EQUIVALENCE statements will be excessive.

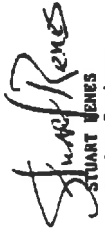


80



I urge the XIJ Committee to take action to correct these problems with the proposed FURTRAN BX Standard. I also request that XI committee to require the XIJ committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

  
STUART RIMES  
Senior Engineer

# 77

Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X1/87-11-075-X, S  
Date: Comment 877  
Project: January 22, 1988  
Ref. Doc.: 67-R  
Reply to:



January 12, 1988  
C-8:88-003  
B294  
(505) 667-5335  
(FTS) 843-5335

1	GEN	8	GEN
2	GEN	9	CIO
3	DATA	10	6EU
4	DATA	11	PRC.
5	GEN		
6	GEN		
7	DATA		

Public Comment for Dpens Fortran Revision  
X3 Secretariat  
Attn: Gwendy Phillips  
CBEMA - Suite 300  
311 First Street, N.W.  
Washington, DC 20001-2178

Gentlepersons:

I have taken a little time to pursue the Fortran 8x standard specification, and have found it not as scary as I was afraid it might have been, or as some people would have had me believe. In fact, there is nothing I have found in a casual reading of the new standard that doesn't seem to belong there. On the other hand, some extensions that were moved to Appendix F should be in the new standard.

Chief among these is the bit declaration and bit manipulation facilities. Bit operations are especially important to programmers who have to write systems programs in Fortran, and are of great use to others as well. In fact, they are of such universal importance that most 8x compilers will probably offer a nonstandard implementation for bit manipulation, as have Fortran 77 compilers. It would be vastly preferable if there were a standard for these declarations and operations.

Pointers have also become important to systems and scientific programmers alike, and, as with bits, there should be a standard implementation before multiple nonstandard versions surface.

Significant blanks would seem to be a minor issue, but a very desirable feature that may help to prevent errors. I urge that this feature be moved back into the standard as well.

1  
2  
3  
4  
5

To: XJJJ -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT 877

Attached is Comment 877 on BSR X3.9-198x (from Bob Malchic of Los Alamos National Laboratory).

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received, and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: Comment 877

cc Jeanne Adams, X3JJ Chair  
Lloyd Campbell, Technical Editor

ACTION REQUESTED

p. 575

\*Operating under the procedures of The American National Standards Institute.  
X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-2178

An Equal Opportunity Employer/Equal Opportunity for Women

Tel 202/737 8888  
Fax 202/638 4127

50

Cwundy Phillips  
C-8:88-00J

-2-

January 12, 1988

I also urge you to resist any temptations to remove any of the following new features, all of which are good additions to the language.

- a) array processing capabilities ] 6
- b) user defined data types ] 7
- c) free format ] 8
- d) the CASE statement ] 1
- e) the new form of the DO statement ] 1
- f) the alternative form of the relational operators (Why don't the logical operators have alternative forms as well?) ] 10
- g) MODULE/USE (a significant improvement over COMMON) ] 11

Thank you.

Sincerely,

*R. L. Malchia*  
Bob Malchia, Programmer/Analyst  
Computer Systems

BM:jfr

Cy: A. L. Marusak, C-3, MS B265  
CRN-4 (2), MS A150  
File

P. 576

80

# 78

80

# JONES COMPUTER CONSULTING Specialists in Real-Time FORTRAN Programming

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

1	GEN
2	GEN
3	DATA
4	PROC
5	PROC
6	GEN
7	GEN
8	GEN
9	GEN

Sirs:

As one who has significant financial ties to FORTRAN, I feel the X3J3 Committee has performed a serious disservice to the user community by not adhering to their charter to "standardize existing practice". The charter of the X3J3 Committee is not to make FORTRAN everything to everybody as this extremely large and unwieldy standard would attempt to be but rather to refine the language to enhance its accumulated worth. Although I feel that an updated FORTRAN standard is overdue, I cannot believe that the proposed FORTRAN 8x standard is what the user community wants or needs. I feel that the X3J3 Committee has, in most instances, grossly exceeded the requirements in such a way as to not fulfil them.

1  
2  
}

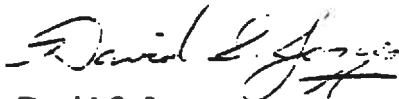
I have specific concerns that:

- The request for new, efficient data types such as BIT and POINTER is not, in my opinion, serviced by the language extensibility mechanisms proposed. ] 3
- The need for a standard INCLUDE statement was not a request for the overly complex MODULE/USE. ] 4
- The overloaded operators add a significant potential for maintenance problems that is not displaced by their utility. ] 6
- The proposed array manipulation routines add too much inefficiency to justify their existance. ] 7
- The proposed replacement of statements such as COMMON and EQUIVALENCE is unwarranted and unneeded. ] 8

In summary I feel that the overkill with which the X3J3 Committee has extended FORTRAN has created a new language that will not displace usage of modern "all purpose" languages such as Ada and PL/I but will alienate the current community resulting in an unused and quickly discarded language.

9

Sincerely,



David S. Jones  
President  
Jones Computer Consulting  
515 N. Shady Shores  
Lake Dallas, TX 75065

88 JAN 19 AM 11:37

EX

Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/87-11-075-X, S  
Comment #79  
Date: January 22, 1988  
Project: 67-X  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 79

Attached is Comment #79 on BSR X3.9-198x from Michael Clover of Los Alamos National Laboratory.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

Attachment: Comment #79

cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

\*Operating under the procedure of The American National Standards Institute.  
X3 Secretariat  
Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-2178

Tel: 202/737-4888  
Fax: 202/638-4922

ACTION REQUESTED

p. 578

Los Alamos  
Los Alamos National Laboratory  
Los Alamos, New Mexico 87545

out January 12, 1988  
transmitted to X7-88-U2  
via air F538  
airmail (505)-667-9797

JAN 19 11 51

1	GEN
2	DATA
3	DATA
4	DATA
cont.	

Public Comment for Dynamic Fortran Revision  
X3 Secretariat, Attn: Cecily Phillips  
CRFEMA  
Suite 300  
311 First Street NW  
Washington, DC 20001-2178

Dear Sirs:

I am the key leader of a large simulation code used at Los Alamos National Laboratory, which consists of over 200k lines of unique code (common blocks unexpanded). It is my impression that your proposed revised standards for Fortran 8x are sorely lacking when it comes to consideration of the types of Fortran programming that my code represents.

I propose to examine the various major proposals, offering my specific criticism, and then to conclude by examining what I perceive to be a wrong philosophical attitude on the part of the X3J3 committee towards Fortran.

With regard to array capabilities, the proposed enhancements strike me as gilding the lily. In a production code, there are indeed loops that run over the body of a matrix, and you would achieve a prime facie reduction from 5 lines of a simple do-loop to one. However, the RANGE and SET RANGE statements bring that back up to 3 lines, with savings in lines of code is not particularly significant. Furthermore, most of the existing code exists in such routines exists to handle the boundary conditions, where the 3 lines of a single do loop would be replaced by the 3 new lines. Those of us that already wait for 3 to 5 CPU minutes for the Cray compiler to compile our code can only anticipate that such suggestions of the instruction sets will only increase compile times with no concomitant advantage.

The proposed generalized floating point provisions will also accomplish nothing for us we are not likely to ever re-part our codes from machines with 64 bit words to those with 32. Further, even for people who benchmark codes, your standards won't help them much if they ask for 45 bits of mantissa, one machine could give them 40, and another 48, so the comparisons would not be any better than what would obtain by asking for double precision right now. And, if manufacturers gave exactly what was asked for, via software multiplications etc., then FORTRAN would run slower than realness in January. Why can't you just standardize IBM's REAL\*4,8,16 and INTEGER\*2,4,8,16 notation so that DOUBLE PRECISION statements are no longer ambiguous? You might consider a standard of 1 byte in 4 for the exponent and sign bits (and 2 bytes for words REAL\*8 and above).

I also find that the provisions for user-definable data types are also unnecessary for a language designed for number crunching. Fortran is not a language for fancy data base management schemes, and it does not improve its efficiency for number crunching by trying to tuck it into one.

An Equal Opportunity Employer/Operator by University of California

179

3

4

80

The "module" capability proposed for 8x is also unnecessary. There exists on the market vendor supplied utilities like Opus's HISTORIAN and CHC and CHH's UPDATE that provide the functionality to write a common block once and propagate it to all relevant subroutines. There is no need to burden the language with constructs better suited to preprocessors and loaders.

79.5

The new control structures are in themselves not objectionable. Our codes do not use computed GOTO statements except once or twice, but it is not clear that the CASE construct would make things the least bit more readable.

79.6

The suggestion that subroutine argument list should be variable in length strikes me as flying in the face of the few good recommendations that have come out of Computer Science academia recently, in that if two things are to be accomplished by the two different calling lists, then two subroutines should be provided to do those two functions. In our codes, we improve readability and maintainability in such a manner. If you want to allow bad coding practices, then just leave the ENTRY statement alone, rather than providing a new bad practice.

79.7

I find the NAMELIST addition to be good, especially since every Fortran compiler that I have ever used has had the construct in it. You should do more standardization of this sort.

79.8

Internal procedures are another laudable new feature—the need has frequently been felt for a two or three statement-long statement function. It would be nice if your standards specified that such procedures be expanded in-line, so that no compiler would be tempted to destroy optimization/vectorization by performing a jump to such a procedure.

79.9

I find the augmentation of the relational operators to be another case of unnecessary redundancy, in part because I find it easier to read code with the unambiguous "GE." while the ">" is harder to distinguish in listings from the parenthesis.

79.10

The dynamic memory constructs appear to be the worst possible way to do memory management as proposed, compilers will be inserting system calls down in the bowels of a program, totally destroying program efficiency. The CUR POINTER construct however, in conjunction with some excellent memory management libraries that have been developed at this laboratory, are orders of magnitude more efficient and free of unwanted side effects. Since SUN has adopted the POINTER construct in its compiler, you people should be standardizing it instead of inventing poor substitutes.

79.11

In conclusion, I would note two very disturbing symptoms that I wish to call to your attention: the augmentations to Fortran make it look more like ADA and C, and your revisions decouple the variables from the memory they're stored in.

79.12

79.13

With regard to the first problem, I do not feel that FORTRAN is anything to be ashamed of it does (or did) what it was designed to do, namely to allow a programmer to write scientific codes in a high-level language that generated efficient machine code on big computers. I do not wish to see Fortran bastardized so that ADA, C, or PASCAL programmers can port their source code unchanged to the FORTRAN compiler. If I wanted to write in ADA, I'd have written in ADA! Just because Computer Science claims that Fortran is old fashioned is no warrant for you to try to turn Fortran into something else. Your chief concern should be that the reason for this language's continued existence, namely efficient compile-ability, is not endangered by anything you do to standardize it.

79.12

In that spirit, it is frequently necessary in a large production code (as contrasted to the "toy" codes that computer scientists are experienced in writing), that REAL arrays be equivalent to INTEGER arrays so that certain making operations can be performed on these arrays, in an efficient manner. You may claim that it should never be necessary to do something like that, but it is not your job to grade my programs. It may also be possible that dull knives are safer than sharp ones, but that is no excuse to dull my tools. People that don't want to risk hurting themselves by writing unsafe Fortran code should write in ADA, and not prohibit me from writing safe code with "unsafe" Fortran constructs.

79.13

If you were really anxious to accomplish something, you would standardize the calling sequence among languages ("call by value" or "call by address"), so that it would be easy to link routines in one language to libraries written in other languages.

79.14

My bottom line is that the current Fortran 8x standard should be completely rejected.

79.15

I remain,

*Michael Clover*  
Michael Clover

MRC/mrc  
cc: Alex Marusak, C-3, MS D265  
1-7 File

Review Letter # 79	
Subgroup Nominations:	
1	GEN 11
2	DATA 18/12
3	DATA 13
4	DATA 14
5	PROC 15
6	CIO 16
7	PROC 17
8	CIO 18
9	PROC 19
10	GEN 20

80

Jay Elewitz  
5333 Ponder Place  
Flower Mound, Tx. 75028  
January 12, 1988

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sir:

- 30.1 I would like to make my position clear regarding the proposed changes to the FORTRAN standard. I think some changes are worth considering but I believe the committee went far beyond its charter in its recommendations. I cannot accept the proposal as written.
- 30.2 I believe that the NAMELIST input/output is an useful improvement, but I feel that the
- 30.3 MODULE/USE feature is a superfluous change to FORTRAN. I am not happy that the INCLUDE statement was not defined and I absolutely feel that the COMMON statement should not be deleted ever. I also think that the EQUIVALENCE statement should not be taken out under any circumstances.
- 30.4
- 30.5 The committee should not be inventing a new language. I believe this proposal will do more harm than good.

Sincerely,

*J. Elewitz*  
Jay Elewitz

X3 J3  
JUN 15 AM 1988

Review Letter # 80			
Subgroup Nominations :			
1	GEN	11	
2	CIO	13	
3	PROC	13	
4	GEN	14	
5	GEN	15	
6		16	
7		17	
8		18	
9		19	
10		20	

DECUS Meeting  
Dallas, Texas  
December 15, 1987

1	GEN
2	DATA
3	PROC
4	DATA
5	GEN
6	GEN
7	GEN

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community. ] 1

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. ] 2
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. ] 3
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. ] 4
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. ] 5 ] 6

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption. ] 7

Sincerely,

*Joseph R. Ziegenfuss*

Company: LTV AIRCRAFT PRODUCTS GROUP  
Address: P.O. BOX 655907 / MSF-77  
DALLAS, TX 75265-5907

15:11W 51 NVP 88.

EX



7 2

Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X1787-11-075-X, S  
Date: Comment 882  
January 22, 1988  
Project: 67-R  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 82  
Attached is Comment 882 on BSR X3.9-198x from Linda G. Yohe of Texas Exchange Carrier Association.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received, and has assigned the comment registry number indicated above.  
The X3.9-198x comment period closes on February 23, 1988.  
If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
Catherine A. Kachurik  
Administrative Secretary, X3

Attachment: Comment 882  
cc: Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

\*Operating under the procedures of The American National Standards Institute  
X3 Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 900, Washington, DC 20001 2178  
Tel: 202/737 6000  
Fax: 202/638 4927

DECUS Meeting  
Dallas, Texas  
December 15, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 900  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTTRAN 8x standard is not in the best interest of the DEC FORTTRAN community.

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient.
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice.
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy.
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive.

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

*Linda G. Yohe*

Company: Texas Exchange Carrier Assoc.  
Address: P.O. Box 58907  
One Bell Plaza, 31st Floor  
Dallas, TX 75250

(over)

88 x1

Review Letter # 82	
Subgroup Nominations:	
1	DATA 11
2	PROC 13
3	DATA 13
4	GEN 14
5	GEN 15
6	GEN 16
7	GEN 17
8	18
9	19
10	20

88

ACTION REQUESTED

P. 582

US CHAPTER, DECUS, COMMENTS ON THE DRAFT FORTRAN STANDARD (FORTRAN 8X)

The Languages & Tools SIG, US Chapter DECUS, has the responsibility for all US Chapter, DECUS, activities involving the Fortran computer language. This SIG has been following the progress the ANSI X3J3 committee for three years. On two occasions, members of X3J3 have made formal presentations to our membership at our Symposium meetings. A member of the L&T SIG Steering Committee responsible for Fortran has also attend an X3J3 meeting. At a meeting on December 11, 1987, the DECUS membership attending our most recent Symposium approved the following statement regarding the proposed Fortran draft standard.

The proposed draft standard for Fortran, is actually a proposal for a new language that happens to be called Fortran. This language does not preserve two important characteristics of Fortran. These characteristics are the compact size of the language and the ability of the code generation sections of Fortran compilers to generate efficient machine code. By any reasonable means of measurement, Fortran 8X is a huge language. Based on experience with the Fortran 77 standard, many computers currently in use, such the PDP-11, would not have sufficient memory to hold a compiler for Fortran 8X. Discussions with compiler writers who are members of DECUS cause us to believe that the code generation sections of Fortran compilers would have to be completely rewritten in order to support the features of Fortran 8X. This would eliminate the computational efficiency of many programs and reduce their reliability while the errors that would inevitably be introduced into the new or revised Fortran compilers are found.

§2.5

We also believe that, in order to use the new features contained in the draft standard, much of the existing Fortran code would have to be revised. This does not protect the huge investment in existing programs.

§2.6

US Chapter, DECUS, recommends that the proposed draft standard for Fortran be withdrawn and that X3J3 be required to produce a new draft that standardizes language features that have already been implemented in at least one commercially available Fortran compiler.

§2.7

US Chapter, DECUS, also would like to note that by our count, at least 7 voting members of the X3J3 committee work for the same employer, i.e., the US Department of Energy. These members all represent National Laboratories controlled by DOE or are employees of companies that manage these laboratories under contract to DOE. We also understand that these voting members are also members of a DOE sponsored language group. We believe that this is in violation of the rules governing ANSI technical committees and hope that ANSI will move quickly to correct this situation.

NOT TO BE CONSIDERED BY THE DPNNS

#83

DECUS Meeting  
Dallas, Texas  
December 15, 1987

- 1 GEN
- 2 DATA
- 3 PROC
- 4 DATA
- 5 GEN
- 6 GEN
- 7 GEN.

SD

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community. ] 1

Specifically, I have major concerns in the following areas:

- DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient. ] 2
- The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice. ] 3
- The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy. ] 4
- The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. / The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive. ] 5 ] 6

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption. ] 7

Sincerely,

Allen W. Wadle  
 Company: ALLEN W. WADLE (M/S EM 27)  
 LTV Missile & Electronics Group  
 Address: P.O. Box 650003  
 Dallas, TX 75265-0003

EZ: 11V SI NVP 88.  
EX

# 84

80

DECUS Meeting  
Dallas, Texas  
December 15, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

X3  
86  
JAN -7  
M

Dear Sirs:


While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- 84.1 • DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient.
- 84.2 • The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice.
- 84.3 • The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy.
- 84.4 • The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive.

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

  
 Company: Austin Bell (TECA)  
 Address: DAE Bell Plant  
Waller TX 75265

Review Letter # 84			
Subgroup Nominations :			
1	DATA	11	
2	PROC	13	
3	DATA	13	
4	GEN	14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	

Revised Standards Committee  
INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3J3-11-075-X, S  
Comment #85  
Date: January 21, 1988  
Project: 67-R  
Ref. Doc:  
Reply to:

To: X3J3 -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 85

Attached is Comment #85 on BSR X3.9-198x from Scott D. Matthews of Idaho National Engineering Laboratory (INEL).

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentor of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

Attachment: Comment #85

cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

P. 586

\*in accordance with the procedures of The American National Standards Institute.  
\*Incl. Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 600, Washington, DC 20001-2178

Tel. 202/717 8888  
Fax. 202/638-4822



December 29, 1987

- 1 GEN
- 2 GEN
- 3 GEN
- 4 PROC
- 5 PROC
- 6 CIO
- 7 CIO
- 8 DATA
- 9 DATA
- 10 DATA
- 11 GEN

ATTN: Gwendy Phillips  
Public Comment for Dpans  
Fortran Revision, X3 Secretariat  
Computer and Business Equipment  
Manufacturer's Association  
Suite 300, 311 First Street, M.W.  
Washington, DC 20001-2178

PROPOSED FORTRAN 8X STANDARD - SOH-07-87

Dear Ms. Phillips:

Like many people in the software industry, I have anxiously awaited the first public draft of the proposed Fortran 8X Standard. Overall I was not disappointed.

The strength of Fortran has been that it is easily addresses the majority of engineering and scientific problems in a straightforward manner. This version continues that tradition. The array processing capabilities will especially be a boon.

Modern practices betray the standard as too large. The processes exist because people hesitate to mark any features as obsolescent. With modern programming practices and software engineering, programmers can better accomplish their software implementation without "Alternate RETURNS", Arithmetic IFS, "ASSIGNED GO IDS", "COMMON", or "EQUIVALENCE". In fact, Fortran 8X addresses all the above features more succinctly. Why is it necessary that all the excess baggage be hauled from one version to the next? Fortran will simply drown into oblivion if the antiquated baggage keeps being hauled around.

The "module" unit, "internal procedures", "CASE", a very versatile "DO", and the "user-defined data types" in Fortran 8X will give Fortran further credence as a language that working software engineers can use.

Although the bit-data type and pointers were neglected, I feel it is more imperative to adopt this standard and then implement those features in Fortran 9X. We need a Fortran standard now! With the advent of ADA and C and their proponents maneuvering for domination in the scientific programming arena, Fortran will not survive into the 21st Century if the Fortran Standard is not quickly adopted. So, please quickly approve the standard so programmers can again apply the features of Fortran to developmental efforts instead of having to use ADA or C as a "Modern Programming" language!

EGS Idaho Inc. P.O. Box 1625 Idaho Falls, ID 83416

SD

#85

Ms. Gwendy Phillips  
December 29, 1987  
SDM-07-87  
Page 2

Thank you for your efforts on this important task.

Sincerely,

*Scott D. Matthews*  
Scott D. Matthews  
Sr. Scientist

sh

cc: Board of Standards Review

80

# 86

**Credited Standards Committee  
INFORMATION PROCESSING SYSTEMS\***

Doc. No.: X3/B7-11-075-X, S  
Date: Comment 886  
Project: January 22, 1988  
Ref. Doc.: 67-R  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT 886

Attached is Comment 886 on BSR X3.9-198x from David Burrows, Ph.D. of The Pennsylvania State University, Department of Astronomy.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kichurik*  
Catherine A. Kichurik  
Administrative Secret

Attachment: Comment 886  
cc: Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

p. 588

**THE PENNSYLVANIA STATE UNIVERSITY**  
315 DAVEY LABORATORY  
UNIVERSITY PARK, PENNSYLVANIA 16802

College of Science  
Department of Astronomy

December 23, 1987

Ann Code 010  
63-9118

Public Comment for Dpans  
Fortran Revision  
X3 Secretariat

Attn: Gwendy Phillips  
Computer and Business Equipment  
Manufacturers Association  
Suite 300  
111 First Street, N.W.  
Washington, DC 20001-2178

Dear Ms. Phillips:

I am writing to comment on the proposed Fortran 8x standard. As I have not yet received the copy of the standard that I requested a month ago, I will have to base my comments on the reviews of it that are available to me (primarily the review in the NMFEC monthly bulletin for November). Unfortunately, I will not have another chance to comment before the expiration of the public comment period. Due to a scheduling conflict in January and February, I hope that the information I am basing my remarks on is accurate.

It has been said that a camel is a horse designed by a committee. The Fortran 8x revision looks to me like the latest camel of computer languages. It would appear that the committee has taken a Fortran base, mixed in liberal doses of C, Pascal, and/or Modula 2, stirred, and concocted a new language which they mistaken refer to as a revision of Fortran. Fortran 8x is not a revision of Fortran: it is an entirely new language. It is incompatible with the basic structure of Fortran programming, and apparently requires a compiler that recognizes two distinct languages in order to maintain a semblance of compatibility with Fortran 77 code. I think that this is an unacceptable situation. I'm sure that computer science types would love to do away with statement labels and column-oriented statement structure, but they are inherent in Fortran. Programmers who spend as much time programming in Fortran as I do have absolutely no trouble dealing with them. Fortran should not be forced to resemble Pascal or C, as the proposed standard seems to attempt to do. Concocting schemes of statement continuation, etc., that ignore column placement

86.2  
86.1  
86.2

Review Letter # 86		
Subgroup Nominations:		
1	GEN	11 PROC
2	GEN	12 GEN
3	GEN	13 PROC
4	DATA	14 GEN
5	DATA	15
6	GEN	16
7	DATA	17
8	PROC	18
9	CIO	19
10	CIO	20

\*Operating under the procedures of The American National Standards Institute.  
1 Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 600, Washington, DC 20001-2178

86

86.2 may be a nice idea, but it is fundamentally incompatible with all previous versions of Fortran. Fortran revisions should be as compatible with previous versions as possible. Enormous amounts of tested code are at stake if this revision is implemented in its present form.

86.3 This is not a small problem. It requires a compiler that recognizes two incompatible coding techniques, which cannot be mixed in a single program unit. Since much new code is heavily based on previous code, this will make for a horrible mess in code development. Either the dual structure will be perpetuated indefinitely, or else millions of lines of old code will become useless.

86.4 My remaining complaints are less fundamental, and there are some features of the proposed standard that look quite good. A major omission, however, is the lack of any intrinsic functions to do bit manipulation on integers. They would be an extremely valuable addition.

86.5 The idea of providing a technique to allow portable floating point specs is good, but it seems to me that the proposed technique is unnecessarily clumsy. Why not use the IEEE standard, or define the minimum precision of single and double precision that the compiler must provide?

86.6 The array operations look very useful.

86.7 I can't remember ever needing to use user defined data types in all my years of Fortran programming. I don't see any need for them.

86.8 I also don't see any need for modules in Fortran. Fortran subroutines and functions already hide their internal variables from the outside world, and vice versa.

86.9 I also don't see any need for CASE statements that aren't already served by the IF-THEN-ELSE block.

86.10 The same is true with the changes in the DO loop. They are just an attempt to C-ify the language.

86.11 Internal subroutines would be a useful addition. It's not clear from the information I have whether they would be able to access variables in the routine that contains them as global variables. In some versions of Fortran I've used in the past this was possible.

86.12 I've heard a rumor to the effect that Sn does away with common blocks and EQUIVALENCE statements. I sincerely hope that this rumor is false! That would be a major problem with most of the large programs I work with.

86.13 A comment in the review I've read refers to "a demise in independent compilation" as being a point of controversy. If the proposed compilation of Fortran code is no longer possible under the proposed standard, this is a disaster indeed! Independent compilation is one of Fortran's strengths.

86.14 In summary, I would like to plead for a return to the philosophy of Fortran 77. Fortran 77 was a very significant improvement over Fortran IV, but was highly compatible with the previous versions. The proposed standard for Fortran 8x is highly incompatible with previous versions, to the point of being a distinct language. Please do not perpetuate this on the community of Fortran programmers! Let those who wish to program in C or Pascal do so, rather than attempt to convert Fortran into something it was never intended to be.

Sincerely,

*David Burrows*  
David Burrows, PhD



#81

SD

DECUS Meeting  
Dallas, Texas  
December 15, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs:

While I believe that an updated FORTRAN standard is overdue, I must agree with Digital Equipment Corporation (DEC) that the proposed FORTRAN 8x standard is not in the best interest of the DEC FORTRAN community.

Specifically, I have major concerns in the following areas:

- 87.1 • DEC users have expressed the need for improved data type support. The proposed standard attempts to satisfy this need by language extensibility mechanisms rather than new intrinsic types. The implementations resulting from this method will be too inefficient.
- 87.2 • The new source manipulation capabilities (MODULE/USE) are more powerful than necessary, are too complex, and are untested in practice.
- 87.3 • The new features added to enhance portability of numerical software are untested in practice and are not clearly effective in obtaining the desired portability because they do not account for such things as round-off error and accuracy.
- 87.4 • The features chosen for possible obsolescence in the future are not justifiable based on potential benefits or costs. The cost of replacing statements such as the COMMON, DIMENSION, and EQUIVALENCE statements will be excessive.

I urge the X3J3 Committee to take action to correct these problems with the proposed FORTRAN 8x Standard. I also request that X3 committee to require the X3J3 committee to correct these and other problems found during the public review prior to re-submitting this proposed standard for adoption.

Sincerely,

*Sharon Clark*

Company: TECA  
Address: One Bell Plaza  
Room 3120  
Dallas, TX 75202

Review Letter # 87			
Subgroup Nominations :			
1	DATA	11	
2	PROC	13	
3	DATA	13	
4	GEN	14	
5		15	
6		16	
7		17	
8		18	
9		19	
10		20	

Accredited Standards Committee  
X3, INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3/B7 11-075 X, S  
 Comment #88  
 Date: January 22, 1988  
 Project: 67-R  
 Ref. Doc.:  
 Reply to:

To: X3J3 -- FOR ACTION

Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 88

Attached is Comment #88 on BSR X3.9-198x from Robert R. Hench of General Electric Information Services, Information Processing Technology.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an estimated date of action. When a final response is issued you must inform the commentors of their need to notify the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

*Catherine A. Kachurik*  
 Catherine A. Kachurik  
 Administrative Secretary, X3

Attachment: Comment #88  
 cc: Jeanne Adams, X3J3 Chair  
 Lloyd Campbell, Technical Editor

\*Operating under the purview of The American National Standards Institute.  
 X3 Secretariat  
 Computer and Business Equipment Manufacturers Association  
 311 Fost Street, N.W., Suite 500, Washington, DC 20001-2178

ACTION REQUESTED

P. 591

7138



General Electric Information Services, Company USA

January 7, 1988

Public Comment for dpsans Fortran Revision

Secretariat  
 TN: Gwendy Phillips  
 WPA  
 11 First Street, N.W.  
 Arlington, D.C. 20001-2178

1	GEN	9	DATA
2	GEN	10	DATA
3	GEN	11	DATA
4	CIO	12	PROC
5	PROC	13	DATA
6	GEN	14	DATA
7	CIO	15	GEN
8	CIO	16	GEN

Information Services would like to offer the following comments regarding dpsans SB.104 - Fortran 8x. For nearly 20 years, we have been a major supplier of computer software services to the worldwide business community and have a substantial interest and investment in Fortran. We were the first to provide a commercially available revision of Fortran 77 and have participated on X3J3.

The inclusion of free form input, structured and derived data types, array-processing features, the CASE statement and recursion is good - these are features which Fortran should have; GFS does not think that sufficient consideration has been given to standardizing existing practice and usage as exemplified by what is currently implemented in Fortran processors and applications. We note that the X3J3 representatives from Data General, DEC, IBM, UNISYS, Borland, and Sirius International made similar comments when voting not to forward the proposed draft for public comment.

Missing common extensions which should be a part of any F77 successor standard include both DO WHILE and DO UNTIL, execution control constructs, variable length character strings, a bit data type, some sort of printer and a macro pre-processor. Should the committee decide to consider inclusion of these features in the next Fortran standard, we offer the following suggestions:

While a sort of variable length character string may be implemented via the MORDIE concept, it has been demonstrated that such an implementation is incomplete. Variable length character strings and a bit data type should be consistently adopted, one with the other. Bit strings and character strings should share similar syntax and semantics throughout the language.

88



GENERAL ELECTRIC  
INFORMATION SERVICES

Gwendy Phillips  
January 7, 1988  
Page Two

14

- 2. A pointer data type should be implemented in the spirit of Fortran. By this we mean that Fortran is a high level language and any adopted pointer type should be more along the lines of those in, say, Pascal rather than "C".

While we are aware that all of F77 is imbedded within the language proposed by SB-104, we do not like that certain of F77's features which are inherent to Fortran are being categorized as deprecated and are candidates for removal from the language in the future.

15

Specifically, the memory binding capabilities of COMMON, EQUIVALENCE, and BLOCK DATA subprograms are significant to both the character and use of Fortran and should not be candidates for removal from future standardizations of Fortran.

16

On the subject of the document itself, we feel that it could be better organized. Section 14, "Scope, Association, and Definition," should either be distributed throughout the document wherever and as often as is required or there should be a liberal use of forward references to the section where necessary. Similarly, Appendix C, "Section Notes," should be made a part of the standard and either included as footnotes throughout the document or referenced where appropriate in the body of the standard.

GE Information Services is grateful for this opportunity to comment on SB-104 and appreciates any consideration which may be given to the ideas presented here.

Sincerely,

*Robert R. Hench*

Robert R. Hench  
Vice President and General Manager  
Information Processing Technology

cc: D. Deutsch  
C. Dickman

/lmb

80

**CG&E** The Energy Service Company

The Cincinnati Gas & Electric Company  
P.O. Box 960 - Cincinnati, Ohio 45201-0960

X3 077

~~#89~~

# 89 (80)

BE JAN 12 AM 120

January 6, 1988

1 GEN  
2 GEN

Public Comment for dpANS Fortran Revision  
X3 Secretariat  
Attn: Gwendy Phillips  
Computer & Business Equip Manufacturers Association  
Suite 500  
311 First Street, NW  
Washington, DC 20001-2178

Dear Ms. Gwendy Phillips:

Having reviewed several summaries of the proposed FORTRAN standards revision, we have only one objection to this proposal. Your recommendation to classify the features COMMON, EQUIVALENCE, and ENTRY as obsolescent appears to be an underserved judgement against three widely used features. In the case of all other features identified for reclassification, an improved method of obtaining the same results has been proposed; but in the case of these features, no alternative has been proposed. Our experience also shows that these features are very heavily used in both the locally written programs and the programs received from outside sources.

2

Until an alternative is found for these features, we do not think their status should be changed.

Sincerely,

Charles A. Light

CAL:kb

cc: Public Comment for dpANS Fortran Revision  
Board of Standards Review  
American National Standards Institute  
1430 Broadway  
New York, NY 10018

Sylvia Sund  
SLAC, Bin 96  
P.O. Box 4349  
Stanford, CA 94305

Credited Stand... Committee  
INFORMATION PROCESSING SYSTEMS\*

Doc. No. X3J3-11-075-X, 5  
Comment #90  
Date: January 21, 1988  
Project: 67-8  
Ref. Doc.:  
Reply to:

To: X3J3 -- FOR ACTION  
Subject: BSR X3.9-198x, Programming Language FORTRAN, TRANSMITTAL OF PUBLIC REVIEW COMMENT # 90  
Attached is Comment #90 on BSR X3.9-198x from Jeanine McDermott of ElectroSPACE Systems, Inc.

In order to provide administrative control, the Secretariat is maintaining a register of all such comments received and has assigned the comment registry number indicated above.

The X3.9-198x comment period closes on February 23, 1988.

If the Technical Committee action is to accept in whole or in part a proposal contained in the comment, then the changes should be sent to the X3 Administrative Secretary together with any TC comments supporting the change. If the TC action is to reject in whole or in part proposals contained in the comment, the response should provide the rationale for the rejection.

The comment should be discussed at the next TC meeting, and if not definitively responded to at once, an interim acknowledgment should be sent along with an explanation of the action. When a final response is issued you must indicate the date of that final response to the Secretariat of their satisfaction or dissatisfaction with the committee's response. The commentor is required to send the Secretariat a written statement indicating acceptance or rejection of the TC response within fifteen working days. The commentor must be made aware that failure to respond within fifteen working days that the comment stands will indicate to the Secretariat that the comment is to be withdrawn.

Attachment: Comment #90  
cc Jeanne Adams, X3J3 Chair  
Lloyd Campbell, Technical Editor

\*Working under the procedures of The American National Standards Institute.  
Secretariat: Computer and Business Equipment Manufacturers Association  
311 First Street, N.W., Suite 500, Washington, DC 20001-2178

Tel: 202/731-8888  
Fax: 202/638-4922

p. 594



ElectroSPACE Systems, Inc.  
Box 831359  
Richardson, Texas 75080-1359

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178



Dear Sir,

I am against the suggested standard for FORTRAN 8x.  
There are several areas that I do not like:

1. The new statements do not seem to mix very well with the current FORTRAN 77 language standard. One example is the USE statement which introduces dependent compilation into the language.

2. There are too many FORTRAN statements, some which will hardly ever be used; 81 statements is a bit much.

3. Some of the useful statements, such as INCLUDE, COMMON, EQUIVALENCE, ASSIGNED GOTO, Computational GOTO and arithmetic IF are being considered for removal in the future versions. These are very useful and necessary statements to have in the FORTRAN language.

Please reconsider the implications of the proposed FORTRAN 8x draft. We do not need a new FORTRAN language so drastically different from the current standard.

Sincerely,  
*Jeanine McDermott*  
Jeanine McDermott  
Programmer/Analyst

JM/sk

P.O. Box 831359 • Richardson Texas 75083-1358 • (214) 678-2000 • TWX 910-687-4768

50  
Handwritten marks



Jeanine McDermott  
265 Daniel Drive  
Plano, Tx 75074

XJJJ Chair  
XJ Secretariat  
211 First Street NW  
Suite 500  
Washington, DC 20001-2178

Dear Sirs

I don't care to the suggested standard of coherence or  
I am disappointed that the DO WHILE statement was not  
included. I am unhappy that the IF/ELSE statement was  
not included. I feel that the DO/WHILE and DO/WHILE  
statements must not be re-vised ever.

Comment 90

7	CIO
8	PROC
9	GEN

Sincerely,

Jeanine McDermott

XJ  
14 JUN 78

# 91

50

THE UNIVERSITY OF TEXAS AT DALLAS



November 18, 1987

X3J3 Chair  
X3 Secretariat  
311 First Street NW  
Suite 500  
Washington, DC 20001-2178

1	GEN
2	PROC
3	DATA
4	DATA
5	DATA
6	GEN

Dear Sir:

1 I feel that the WHERE statement is an important extension. I feel that the  
 2 MODULE/USE feature is an unneeded modification to the language. I am  
 shocked that the BIT data type was not defined. I am also disappointed that  
 the varying length CHARACTER data type was not included. I am surprised  
 3 that the VMS RECORD structures was missing. I think that the COMMON  
 statement should not be taken out of ever. I also believe that the  
 EQUIVALENCE statement should not be deleted under any circumstances.  
 4  
 I believe that the DIMENSION statement must not be deleted from any  
 5 FORTRAN of the future.

Sincerely,

*Joe Nation*

Joe Nation

Subgroup Nominations for Public Review Letter #73

87

1. GEN	26. DATA	51. PROC
2. DATA	27. DATA	52. PROC
3. DATA	28. ED	53. PROC
4. DATA	29. DATA	54. PROC
5. GEN	30. DATA	55. PROC
6. ED	31. DATA	56. PROC
7. ED	32. DATA	57. PROC
8. ED	33. DATA	58. PROC
9. ED	34. DATA	59. PROC
10. ED	35. DATA	60. PROC
11. DATA	36. CIO	61. PROC
12. DATA	37. DATA	62. PROC
13. GEN	38. DATA	63. PROC
14. GEN	39. DATA	64. PROC
15. GEN	40. DATA	65. PROC
16. GEN	41. DATA	66. PROC
17. GEN	42. DATA	
18. ED	43. GEN	
19. ED	44. GEN	
20. ED	45. GEN	
21. GEN	46. GEN	
22. GEN	47. CIO	
23. GEN	48. CIO	
24. DATA	49. CIO	
25. DATA	50. CIO	





american national standards committees:  
 X3—Computers & Information Processing  
 X4—Office Machines & Supplies

COMMITTEE CORRESPONDENCE

Doc. No.

Date July 18, 1977

Project 67, 68

81

secretariat: BEMA, 1828 L St., N.W., Washington, D.C. 20036 (202) 466-2288

FRANK ENGEL, JR.  
 179 Lewis Road  
 Belmont, Mass. 02178

The ANSI Technical Committee X3J3 appreciates your interest in the standardization and development of the FORTRAN programming language. The comments you and 288 others contributed give evidence of careful study and thoughtful attention in reviewing and commenting upon the draft proposed FORTRAN standard, BSR X3.9, that was published in March 1976 SIGPLAN Notices. The public response was overwhelmingly favorable, and urged X3J3 to complete the final approved ANS X3.9 FORTRAN as soon as possible.

A detailed report of the X3J3 processing of the comment letters and amending the BSR X3.9 document was presented to X3, and may be obtained from the Secretariat, CBEMA. Upon receiving a comment letter, the X3J3 Secretary gave it a sequence number, identified each separate item discussed in the letter, and assigned responsibility for each item to one of seven X3J3 working groups. The marginal notation C7.3-4 identifies the third item of comment letter 7, for which group 4 had cognizance. The annotated letters, comprising 1225 pages were distributed to X3J3 members as working document X3J3/81. The working group prepared a response for each item assigned to it, and when a change to BSR X3.9 was deemed necessary, the group prepared and submitted a proposal for consideration by the full committee, together with the textual modifications recommended by the group to effect the change. Each comment letter and its responses were reviewed independently by at least three X3J3 members, their critiques were reviewed at the May 1977 X3J3 meeting, and the approved set of responses as appropriately modified by X3J3 is contained in document X3J3/91. The voluminous documents X3J3/81 and /91 are on file with the X3 Secretariat.

The 2397 comment items were distributed among several categories as follows: Addition of new features, 40% of the items, of which X3J3 accepted 18%; Modification to existing features, 16% (46% accepted); Deletion of features, 6% (21% accepted); Textual clarity, style, etc., 27% (56% accepted); Clarification and interpretation, 6%; and Misunderstanding, 2%. The many suggestions for improvement of the document were carefully considered, and a significant portion were accepted. X3J3 wished to avoid any drastic changes that would delay the timely completion of the new standard; that would seriously impact on the implementations in progress; or that were not immediately crucial to a sound and viable standard. Changes that corrected errors and misstatements or improved clarity were considered mandatory. X3J3 recognized the desirability of modifying or removing syntactic forms that might inhibit future developments as suggested by the public comments or that would improve program portability. Over 500 motions to change the document were considered, with 83% being approved, 14% rejected, and 3% tabled without a deciding vote. Items that were rejected as inappropriate for this revision will be candidates for inclusion in the next revision for which the draft document is projected for public review in 1980.

Enclosed are a summary of the X3J3 actions in response to all of the public comments, a copy of your annotated letter from X3J3/81, and the responses to your comments as they appear in X3J3/91. X3J3 generalized some of the 1166 item responses so that they could be used to respond to similar comment items appearing in different letters. Thus you may find that a response may reference certain details about things you did not mention. Should you believe that your comments have not been answered satisfactorily, you should notify the Chairman, Board of Standards Review, American National Standards Institute, 1430 Broadway, New York, NY 10018, within fifteen working days from the date of this letter, giving reasons for your continued objection. In order further to expedite consideration, X3J3 would appreciate receiving a copy sent directly to Mr. Campbell.

The amended BSR X3.9 FORTRAN document, X3J3/90, that resulted from the consideration of the public comment is an improved FORTRAN language and document. Thank you for your suggestions.

B598 *Frank Engel, Jr.*

Summary of X3J3 Actions in Response to Public Comment

The following is a summary of X3J3 actions in response to public comment on the draft proposed revised FORTRAN standard X3J3/76 published for public review in March 1976.

A number of public comments resulted in X3J3 actions to change the draft; others resulted in X3J3 actions affirming the features of the language substantially as described in the draft. These two groups of items are listed separately in this summary.

It should be noted that the draft proposed standard describes two levels of the FORTRAN language. All changes made to the full language in response to public comment were carefully reviewed for their applicability to the subset language, and where necessary appropriate action was taken by X3J3 to change the subset as well. Items in this list marked (\*) involved changes to the subset language as well as to the full language.

Part 1. Actions resulting in change to the March 1976 Draft

- \*1. Further restrictions on collating sequence. The letters A - Z and the digits 0 - 9 must not overlap in the collating sequence. This restriction was not included in the Draft.
- \*2. Blank lines. A completely blank line among the lines of a program unit is considered to be a comment line. Formerly it was the initial line of a statement.
- \*3. Statement ordering. A further restriction was adopted, to the effect that symbolic names of constants, or variables appearing in dimension bound expressions, must not be explicitly typed farther down in a program unit.
- 4. Form of a constant. The rules were relaxed to permit an integer constant (as well as a real constant) as either part of a complex constant.
- 5. Data type of a symbolic (PARAMETER) name. Type rules for PARAMETER statements were changed, so that the type of the symbolic name of a constant depends upon the form of the name rather than upon the form of the constant.
- 6. Character length for a symbolic (PARAMETER) name. A symbolic name for a constant of character type may be declared in a CHARACTER statement with a length specification consisting of an asterisk. The actual length for such a symbolic name is determined from the length of the character constant in the PARAMETER statement. This way of specifying length was not provided in the Draft.
- 7. Restriction to integer type, for subscripts etc. In a number of places where the Draft permitted integer, real, or double precision expressions, only integer expressions are now permitted. These are: subscript expressions, substring expressions, label selectors for computed GO TO, external unit identifiers, record number specifiers, record length specifiers, and alternate return specifiers.

\*8. Intrinsic function names. The Draft provided that certain type statements could change an intrinsic function name to external. This has been changed so that only an EXTERNAL statement can remove a function name from the intrinsic category in this way.

9. PARAMETER syntax. The following changes were made to the Draft:

An expression, which may include symbolic names of previously defined constants, may be included in the definition of a symbolic constant in a PARAMETER statement.

The entire list of definitions must be enclosed in parentheses.

10. Exponentiation. An integer, real, or complex expression may be raised to a real or complex power. Complex exponentiation was prohibited in the Draft. Rules have been included to determine the principal value of the result.

Exponentiation with integer powers is now permitted in an arithmetic constant expression.

11. Logical operators. Two logical operators were added to those described in the Draft. These are .EQV. and .NEQV., and both of these operators have lower precedence than any of the previous logical operators. If L1 and L2 are logical expressions, then L1 .EQV. L2 is true when L1 and L2 are both true or both false, and L1 .NEQV. L2 is true when one is true and the other is false.

\*12. SAVE statement. The following changes were made to the Draft:

An integer variable specified in a SAVE statement, that is defined with a statement level value, no longer becomes undefined upon return from a procedure.

When a labelled common block is specified in a SAVE statement in one subprogram of an executable program, the block must be specified in a SAVE statement in every subprogram in which it appears.

13. DATA statements. The rules for correspondence between the list of names and the list of constants were relaxed, so that an entity of complex type (as well as integer, real, or double precision) in either list may correspond to an entity of any of these types in the other list.

\*14. Block IF. The following statements were added to the language:

```
IF (e) THEN
ELSE IF (e) THEN
ELSE
END IF
```

For each IF-THEN statement there must be a corresponding END IF statement. Between the IF-THEN and the corresponding END IF there may appear any number of ELSE IF-THEN statements, followed by at most one ELSE. Groups of statements delimited by IF-THEN and END IF must be properly nested, with respect to other such groups of statements and with respect to DO loops

\*15. Restrictions on transfer of control. The following rules were adopted:

Transfer of control into the range of a DO loop is prohibited.

Transfer into a group of statements delimited by IF-THEN, ELSE IF-THEN, ELSE, or END IF, from outside the immediate group, is prohibited. However, transfer to an END IF statement is not prohibited.

16. Input and output statements. The following changes were made:

An IOSTAT specifier may be included in any input or output statement (including auxiliary and file positioning input or output statements) to permit determination of the cause of an error or end-of-file condition without requiring a transfer of control. (The ERR = and END = specifiers are retained.)

\* An asterisk may be used as a unit specifier in a WRITE statement, or in a READ statement that contains a control information list. The asterisk identifies the same processor-determined external unit (which must be preconnected for sequential access) that is used for a PRINT statement or for a READ statement without a control information list, respectively.

\* An empty list is no longer prohibited.

The "array block" list item form was deleted. (An unqualified array name is still permitted.)

\*17. Input and output errors. The following changes were made:

There are no longer any "mandatory" errors that must be treated as errors by all standard conforming processors. Instead, the list of error conditions is processor dependent.

When an error or end condition occurs during an input or output operation, any implied-DO variables in the input or output list become undefined.

18. File properties. The following changes were made:

The Stream access method is no longer provided.

\* A file may have more than one allowed access method (Sequential or Direct) and more than one allowed form (Formatted or Unformatted). However, for any connection at most one access method and one form are established.

\*19. Connection properties. Properties formerly associated with a file are now properties of a connection between a file and a unit. This includes the following:

An access method (Sequential or Direct) is established for the connection.

A form (Formatted or Unformatted) is established for a connection to a file that exists. (Note that a Sequential connection to a file that exists now has a form property.)

A record length is established for a connection whose access method is Direct.

A blank significance property (ZERO or NULL) is established for a connection whose form is Formatted. If the connection results from execution of an OPEN statement that does not explicitly specify a blank significance property, the default is BLANK = NULL.

20. Auxiliary input and output statements. The following changes were made:

New specifiers SEQUENTIAL, DIRECT, FORMATTED, and UNFORMATTED were added to the INQUIRE statement to provide information concerning the set of allowed access methods and the set of allowed forms for a file.

- \* The MAXREC specifier is no longer included in the OPEN and INQUIRE statements.

A BLANK specifier is now included in the INQUIRE statement, to permit determination of whether the currently established blank significance property is ZERO or NULL.

An inquiry specifier must not be referenced by any other inquiry specifier in the same INQUIRE statement (to avoid side effects).

A file that is opened as a scratch file must not be closed with KEEP status.

21. Records. The following changes were made:

Records written by a suitable explicit format may be read by list-directed formatting.

- \* The length of an unformatted record is measured in processor-dependent units. If the list of an unformatted WRITE does not fill the record on a file connected for direct access, the remainder of the record is undefined.
- \* In a formatted record, a value corresponding to an input list item of logical type may contain an optional period before the T or F (so that the logical constant forms .TRUE. and .FALSE. may be used as input values). For list-directed input, the input form corresponding to a list item of complex type may have blanks before or after the real or imaginary part.

22. Format specification. The following changes were made:

- \* Negative zero is prohibited in the exponent field for E and D output.

The form  $Gw.dEe$  was added, which has the effect of  $Fw.d$  or  $Ew.dEe$  depending upon the magnitude of the datum. The form  $Ew.dDe$  was removed, but  $Ew.dEe$  was retained.

- \* The form  $Ew.dEe$  was added to the subset language.

The forms  $+nX$  and  $-nX$  were changed to  $TRc$  and  $TLc$  respectively.  $TRc$  has exactly the same effect as  $nX$ , if  $c$  and  $n$  have the same values. Neither affects the output record length.

23. Intrinsic functions. The following changes were made:

Intrinsic functions ICHAR and CHAR were added, to provide conversion of a character to an integer and vice versa. The pattern matching function INDEX was also added.

- \* ICHAR was added to the subset language.
- \* REAL is now permitted as a specific name with an integer argument. (This should encourage use of REAL and INT instead of FLOAT and IFIX in the subset language). REAL with a complex argument must now be considered a generic name. DBLE is retained as a generic name only; DFLOAT is eliminated entirely.
- \* The type conversion functions (INT, IFIX, IDINT, FLOAT, SNGL, REAL, CHAR, ICHAR) are no longer permitted as actual arguments.
- \* The intrinsic function SIGN is no longer undefined when its second argument is zero; its value is now defined as the absolute value of the first argument.
- \* The generic name of the arctangent function with two arguments was change to ATAN2.
- \* The definitions of ATAN2 and CLOG were corrected.

24. Subprograms involving character data type. The following changes were made:

Character function names may now be used as actual arguments.

A statement function of type character, having constant length, is now permitted.

Character strings of "adjustable" length (specified by a non-constant length expression) are now prohibited as dummy arguments. However, the "passed" length (determined by the length of the actual argument) is retained.

The length specification for a statement function dummy argument of character type must be an integer constant expression.

25. Other subprogram features. The following changes were made:

- \* An asterisk is permitted as the upper dimension bound for the last dimension of a dummy array. The "assumed size" of such a dummy array is determined from the size of the actual argument array.

If a function subprogram contains ENTRY statements, a reference to one of the function procedures defined by the subprogram must not cause definition of an associated function procedure name (as a variable) of a different type.

- \* A subroutine with no arguments may be defined with empty parentheses following the subprogram name. The same applies to an ENTRY statement in a subroutine. The parentheses in a FUNCTION statement are mandatory even if there is no argument list.

26. Scope of a name. The scope of the name of an implied-DO variable in a DATA statement was not specified in the Draft. The scope of such a name is now specified to be the implied-DO list.

Part 2. Actions resulting in no change to the March 1976 Draft

During the processing of public comments, proposals were rejected by the committee which would have made the following changes to language features described in the Draft published for public review in March 1976.

1. Arrangement of statement lines.

Permit more than one statement to be written in the sequence of an initial line and its continuation lines, with a semicolon between statements and an optional semicolon at the end.

Permit a # character on any initial line or continuation line except an initial line that contains an END statement. Characters on the line to the right of the # would be treated as a comment.

2. Longer symbolic names.

Permit variable names up to 8 characters in length.

Permit variable names, array names, and symbolic constant names up to 16 characters in length.

3. Alternative forms for logical constants.

Permit .T. and .F. as alternative forms for logical constants.

4. Additional data types.

Add Double Precision Complex data type, or Bit data type, to FORTRAN, along with the necessary intrinsic functions, etc.

5. Different substring syntax.

Change the notation for a substring of an array element, by placing the substring specifier inside the parentheses that delimit the subscript, with the substring specifier preceding the subscript expressions.

Change the substring specifiers to first character and substring length, instead of first character and last character.

Replace the parentheses that delimit the substring specification, with angular brackets < and >.

6. Integer DO variables.

Require that a DO variable be of integer type, and that the DO loop parameters be integer expressions.

## 7. Expressions and Assignment.

Provide .NG. and .NL. relational operators, as alternatives to .LE. and .GE. respectively.

Provide .XOR. logical operator, with the same precedence as .OR., instead of .NEQV.

Permit "full array references" (of the same form as an array element reference, but with asterisks in place of subscript expressions) on the left side, and in the expression on the right side, in an assignment statement. All dimensions would be required to match throughout an assignment statement containing such references. The right side would be prohibited from containing scalars or other entities partially associated with the array on the left. The interpretation would be equivalent to a sequence of arithmetic assignment statements, one for each of the elements of the arrays so referenced.

## 8. Control statements.

Delete the assigned GO TO statement (but retain statement label assignment for use with FORMAT statements).

Delete the keyword THEN from the block IF and ELSE IF statements.

Add CASE, WHILE, or UNTIL statements.

Change the minimum DO loop iteration count from zero to one.

Delete the alternate return feature in subroutine calls.

Provide an internal procedure mechanism (within a program unit).

## 9. Input and output.

Prohibit execution of an OPEN statement specifying a unit that is already connected.

Permit a RECL specifier, to establish a maximum record length, in an OPEN statement for a sequential connection.

Provide the NAMELIST input and output features.

Provide the R edit descriptor.

## 10. Additional intrinsic functions.

Add the intrinsic functions EPSLN, INTXP, and SETXP, to provide information concerning precision parameters of the host computer.

Add the VERIFY function with two character string arguments, which is true if each character of the first string is also present in the second.



Add "lexical comparison" functions of logical type, one of which is true if the first argument precedes the second in the ASCII collating sequence, and the other one of which is true if the first argument follows the second in the ASCII collating sequence.

Add complex trigonometric intrinsic functions CTAN, CSINH, CCOSH, CTANH.

Add a one-argument sign function SIGNUM, whose value is -1, 0, or +1 according as the argument is negative, zero, or positive.

Part 3. List of Major Differences from ANS X3.9-1966 FORTRAN

The following is a list of major differences between the draft proposed FORTRAN standard (Document X3J3/90) and the previous standard, ANS X3.9-1966.

An extremely important consideration in the preparation of the draft proposed standard was the minimization of conflicts with the previous standard. The differences listed here represent (with only two exceptions, noted below) extensions to, rather than conflicts with, the previous standard.

It should be noted that the draft proposed standard describes two levels of the FORTRAN language, referred to as FORTRAN (the full language) and subset FORTRAN. Differences noted in this list refer to the full language.

1. "Structured" branching statements. The following statements have been added to the language:

```
IF (e) THEN
ELSE IF (e) THEN
ELSE
END IF
```

For each IF-THEN statement there must be a corresponding END IF statement. Between the IF-THEN and the corresponding END IF there may appear any number of ELSE IF-THEN statements, and at most one ELSE (which must not precede any of the ELSE IF-THEN statements). Groups of statements delimited by IF-THEN and END IF must be properly nested, both with respect to other such groups and with respect to DO loops. Transfer of control into such groups is prohibited.

2. Character data type. A new data type, consisting of character strings of fixed declared length, has been added to the language. Included are character constants, character variables, and arrays of character data. Operations on character data include concatenation and designation of substrings. Intrinsic functions for conversion between single characters and small integers, for pattern matching, and for determining the length of a string are included.

The Hollerith data type of ANS X3.9-1966 has been deleted. Because this introduces a conflict with the previous standard, it is anticipated that some processors will wish to retain Hollerith data as an extension to the new standard; accordingly Appendix C (Section 21) of the draft proposed standard contains recommendations for the form such an extension should take.

3. DO loop changes. A DO statement specifying a terminal parameter whose value is less than that of the initial parameter is no longer prohibited. If the incrementation value is positive, such a statement specifies a loop to be executed "zero times." Negative increments are also permitted. The DO variable remains defined at completion. Transfer of control into a DO loop is prohibited (in conflict with the previous standard).
4. List-directed input and output. A form of input and output is provided, which does not require an explicit format specification. The form of the external representation is determined by the input and output list items.
5. Expressions. An arithmetic expression may include subexpressions of more than one type. (If an operator has two operands of different types, the operand whose type differs from that of the result is converted before the operator is applied.) A subscript expression may be any integer expression. A DO parameter may be any expression of integer, real, or double precision type.
6. Compile-time constants. A PARAMETER statement has been provided, which declares the value corresponding to the symbolic name of a constant. Such a name may be used in an expression, in a DATA statement, or in following PARAMETER statements.
7. Implicit type declaration. An IMPLICIT statement may be used to declare implicit types for variables and array names beginning with certain letters.
8. Generic intrinsic functions. Many intrinsic (predefined) functions produce a value whose type depends upon the type of the function arguments.
9. Subprogram reference. Subroutines and functions may contain ENTRY statements, and subroutines may have alternate returns.
10. Array bounds. An array declaration may include both upper and lower subscript bounds; if only one of these is specified, the default lower bound is one. Arrays may have up to seven dimensions. The upper dimension bound for the last dimension of a dummy argument array may be an asterisk, designating that the size of the array is to be determined from the actual argument.
11. Computed GO TO default. If the control expression of a computer GO TO is out of range, execution continues with the statement following the computed GO TO.
12. Input and output statements. The following features have been included:
  - An output list may contain constants and expressions.
  - An input or output statement may contain a character string to be used as the format specification.

End and error condition control for input and output are provided.

Tab edit descriptors have been added for format specification.

Direct access input and output have been provided.

A character array may be used as an internal file.

OPEN, CLOSE, and INQUIRE statements are included.

13. SAVE statement. Values of entities in a subprogram may be preserved during the time when the subprogram is no longer being referenced, if the names of the entities are specified in a SAVE statement.

14. FORTRAN character set. The apostrophe and the colon are added to the FORTRAN character set. The collating sequence is only partly specified.

15. Comment lines. An asterisk or a C in column 1 designates a comment line.

To: X3J3

107(\*)ALM-2

From: Alex Marusak

Date: February 8-12, 1988 (107th meeting)

Subject: Compiler-Verifiable Aids to Naming in Fortran 8x

#### DISCUSSION

Since the beginning of Fortran array references have been confused with function references. That is, absent other information, the name 'EXT' in

$$X(I,J,K) = EXT(X,I,Z)$$

may refer to a function or to an array.

With the advent of whole-array notation in Fortran 8x, array names may be confused with functions names as before, and in addition may be confused with scalar names. That is,

$$X = Y(I,J,K) + Z - EXT(I,J,K)$$

may be, absent additional information, either an array or a scalar operation, with or without function references.

Names of variables of user-defined type (structures), together with the use of user-defined operators, may exacerbate the problem.

Some languages, recognizing the problem, have distinguished functions and arrays by using the pair '()' for one and the pair '[' ]' for the other.

Several X3J3 members, in a related matter, have suggested problems with determining whether in the array expression

$$X = X + Y*Z/C$$

the ranges are 'effective' or 'declared'. Although this concern over range misses the point (it is less important whether the range is 'effective' or 'declared' than whether there is a range at all), it is symptomatic of the concern over notation.

The following proposals, if approved, will allow users to distinguish certain classes of names with one character (each).

In the best traditions of Fortran, use of these distinguishing characters is completely optional.

PROPOSAL

107(\*)ALM-2

In X3J3/S8.104, page 3-5, add:

Section 3.3.3 Use of Certain Names:

A name whose associated entity is a function procedure may be used in all cases interchangeably as 'name' or as '\$name'. The leading currency symbol is not part of the name and does not count towards the maximum of 31 characters.

A name whose associated entity has the array characteristic may be used in all cases interchangeably as 'name' or as '\_name'. The leading underscore is not part of the name and does not count towards the maximum of 31 characters.

A name whose associated entity is of derived type may be used in all cases interchangeably as 'name' or as '?name'. The leading question mark is not part of the name and does not count towards the maximum of 31 characters.

Names whose associated entity is more than one of 'procedure', 'array', or 'derived type' may have correspondingly more than one of a leading currency symbol, a leading underscore, and a leading question mark, in any order.

Examples:

```
REAL, ARRAY(9,9,9) :: X, Y      ! may be written _X, _Y
TYPE (USERS) :: BLIVIT        ! may be written ?BLIVIT
INTERFACE
FUNCTION EXT (I,J,K)          ! may be written $EXT
TYPE (USERS), ARRAY (10) :: EXT ! or $?_EXT, ?EXT, _EXT, $?EXT,
                                ! etc.
END INTERFACE
. . .
X = _Y + $EXT (I,J,K)
. . .
```

END OF PROPOSAL

107(\*)ALM-2

TO: X3J3  
FROM: Jerry Wagener  
SUBJECT: X3J3/S14.107 (WG5 Resolutions, Liverpool, August 3-7, 1987)

Resolution L1 - Submission of S8 to SC22  
individual vote: 28-7-0 country vote: 7-2-0

That WG5 confirms the action of its convenor in forwarding S8.104 to SC22 for processing as a Draft Proposed Standard.

X3J3 Response. X3J3 concurs with this resolution, and notes that it requires no action on X3J3's part.

---

Resolution L2 - French Translation  
individual vote: 34-0-1 country vote: 9-0-0

That WG5 appreciates the offer of the French member body (AFNOR) to be responsible for the French translation of the final standard.

X3J3 Response. X3J3 concurs with this resolution, and notes that it requires no action on X3J3's part.

---

Resolution L3 - X3J3 Schedules  
individual vote: 31-3-1 country vote: 8-1-0

That WG5 explicitly states that all recommendations to X3J3 and requests for additions, deletions, study, etc., of Fortran 8x features contained in the Liverpool resolutions should be processed by X3J3 in a way which does not delay the public comment process.

X3J3 Response. X3J3 appreciates WG5's strong sentiment that no WG5 resolution unduly delay the processing of Fortran 8x.

---

Resolution L4 - Status of Halifax Resolutions  
individual vote: 31-0-4 country vote: 9-0-0

That WG5 thanks X3J3 for its work in response to the Halifax resolutions, acknowledges receipt of N227 (status of Halifax Resolutions) and notes that no further work is requested of X3J3 on those resolutions.

X3J3 Response. X3J3 notes and appreciates this resolution.

---

83a

Resolution L5 - Resolution Life Cycle  
individual vote: 35-0-0      country vote: 9-0-0

That WG5 now establishes a standing operating procedure whereby reporting of actions and responses to resolutions shall not be carried forward beyond the next regularly scheduled WG5 meeting, and that unresolved resolutions from the previous meeting shall be withdrawn unless they are the subject of explicitly reaffirmed or amended resolutions.

X3J3 Response. X3J3 notes that this resolution requires no X3J3 response. However, X3J3 strongly supports this resolution, and commends WG5 on this action.

---

Resolution L6 - Content of Resolution Response Document  
individual vote: 35-0-0      country vote: 9-0-0

That WG5 requests its convenor to ensure that responses to WG5 resolutions shall be recorded in a document which includes the following for each resolution:

1. The full text of the resolution, including the WG5 voting figures.
2. The response, including the voting figures and explanation where relevant.

X3J3 Response. X3J3 concurs with this resolution, and trusts that this response document adequately reflects the requested format.

---

Resolution L7 - Temporary Nature of an Extension Features Appendix  
(cf. Halifax 3)      individual vote: 23-9-3      country vote: 8-1-0

That WG5 reaffirms its Halifax resolution 3 (1986), namely that WG recommends to X3J3 that the final published document not contain an appendix of suggested extension features.

X3J3 Response. X3J3 intends to remove Appendix F from the final document. A Fortran Journal of Development (see 106-JCA-20, approved as amended at meeting #106 by a vote of 18-12) will be maintained as X3J3 standing document X3J3/S17. The initial contents of this document will be Appendix F. S17 will be evolved by X3J3 and made available to users and implementers of Fortran 8x; it will also be the ~~principal~~<sup>a source</sup> source of incremental features during the development of Fortran 9x.

---

832

Resolution L8 - Pointers (cf. Halifax 11)  
individual vote: 22-3-10      country vote: 8-0-1

That WGS reaffirms the intent of Halifax resolution 11 (1986), namely that WGS feels that a major feature that is lacking in the current S8 is that of a pointer facility.

X3J3 Response. Assigned to the Data Concepts subgroup.

This topic continues to receive X3J3 attention, and a tutorial on a recursive data structures form of pointers was given at X3J3 meeting #105 (Liverpool, August 1987). Following the tutorial, a straw vote of 18-11-5 favored this approach to pointers, and consequently a proposal was scheduled for discussion at X3J3 meeting #106 (Ft. Lauderdale, November 1987). At meeting #106 a formal roll call vote on pointers, requiring a two-thirds majority, failed 18-10-2. This vote was taken in the context of discussions on appropriate times to pursue major technical issues, and consequently it was decided not to pursue pointers ~~during the public review period~~, *until after receiving the public review comments.*

---

Resolution L9 - Pointers and IDENTIFY  
individual vote: 13-9-13      country vote: 3-2-4

That WGS recommends that if pointers are adopted by X3J3 they should either be integrated into the IDENTIFY facility, or else the latter facility should be deleted.

X3J3 Response. Assigned to the Data Concepts subgroup.

The recursive data structures proposal mentioned in the preceding resolution response uses the IDENTIFY statement for pointer assignment.

---

Resolution L10 - Deprecated Features (cf. Halifax 13)  
individual vote: 33-0-5      country vote: 7-0-2

That WGS withdraw its Halifax resolution 13 (1986).

X3J3 Response. X3J3 notes and appreciates this resolution.

---

Resolution L11 - Decremental Features  
individual vote: 28-4-3      country vote: 8-0-1

That WGS recognizes the motivation for splitting decremental features in Fortran 8x into two different classes.

However, WGS requests X3J3 to consider the possibility of identifying both obsolescent and deprecated features in the text of the standard, consistent with the requirement imposed on processors for detecting both classes of decremental features.



X3J3 Response. Assigned to the General Concepts subgroup.

---

Resolution L12 - Significant Blanks (cf. Halifax 14)  
individual vote: 26-8-1      country vote: 9-0-0

The WG5 reaffirms the intent of Halifax resolution 14 (1986), namely that WG5 believes that significant blanks are logically associated with free source form and that the appropriate time to introduce this feature is at the same time as the free source form, even if no syntax in Fortran 8x is dependent on its presence. The presence of significant blanks in Fortran 8x will give greater flexibility for the future development of the language and will simplify development of software tools.

WG5 notes the response prepared by X3J3 to Halifax resolution 14 but requests X3J3 to reconsider this matter as part of its processing of the comments received during the public review period.

X3J3 Response. Assigned to the General Concepts subgroup.  
At X3J3 meeting #105 (Liverpool, August, 1987), by a straw vote of 16-7-4, it was decided to defer action on significant blanks until after receiving the public review comments.

---

Resolution L13 - Name-directed I/O (cf. Halifax 22)  
individual vote: 23-6-6      country vote: 5-0-4

That WG5 withdraw its Halifax resolution 22 (1986).

X3J3 Response. X3J3 notes and appreciates this resolution.

---

Resolution L14 - Language and Style  
individual vote: 35-0-0      country vote: 9-0-0

That WG5 appreciate that X3J3 has made significant improvements in the readability of S8, and particularly wishes to thank Lloyd Campbell and Walt Brainerd for their efforts.

WG5 suggests to X3J3 that the index be improved and that more examples be added.

X3J3 Response. Assigned to the Editorial subgroup.

---

83a

Resolution L15 - Section Notes

individual vote: 27-4-4      country vote: 9-0-0

That WG5 requests X3J3 to introduce some reference mechanism between the text of the standard and the section notes.

X3J3 Response. Assigned to the Editorial subgroup.

---

Resolution L16 - Revision Indication

individual vote: 25-5-5      country vote: 9-0-0

That WG5 suggests to X3J3 that each succeeding internal draft of Fortran 8x have some indication of changes and deletions with respect to the previous draft.

X3J3 Response. Assigned to the Editorial subgroup.

---

Resolution L17 - Program Size and Complexity

individual vote: 20-9-6      country vote: 7-1-1

That WG5 requests that X3J3 investigate the possibility that a standard conforming processor should be capable of detecting and reporting violation of its processor dependent limits on program size and complexity.

X3J3 Response. Assigned to the General Concepts subgroup.

---

Resolution L18 - Usage of Interfaces

individual vote: 33-0-2      country vote: 9-0-0

That WG5 requests that X3J3 add more examples to clarify definition and usage of the concept of interfaces.

X3J3 Response. Assigned to the Procedures and Program Units subgroup. The section describing procedure interface blocks is being rewritten to make it clearer, and more examples illustrating the use of interface blocks will be added.

---

Resolution L19 - Multiple Character Sets

individual vote: 24-1-10      country vote: 7-0-2

That WG5 recommends that X3J3 in cooperation with the Japanese member body add a facility to the Fortran language to manipulate as data within a single program unit more than one character set, with very different numbers of characters in each set, so as to allow for the use within Fortran of natural

languages such as Chinese or Kanji. Further WG5 recommends that such facility accommodate mixtures of characters from different character sets in input and output.

X3J3 Response. Assigned to Jim Matheny.

Tutorials on this topic were presented and discussed at X3J3 meetings #104 (Seattle, May 1987) and #105 (Liverpool, August 1987). A proposal to extend the CHARACTER data type with a KIND= type parameter was scheduled for discussion at X3J3 meeting #106 (Ft. Lauderdale, November 1987), but a proposal for an NCHARACTER (National CHARACTER) data type was introduced instead. A formal role call vote on large character sets, requiring a two-thirds majority, failed 14-13-4. This vote was taken in the context of discussions on appropriate times to pursue major technical issues, and consequently it was decided not to pursue large character sets ~~during the public review period.~~ *until after receiving the public review comments.*

---

Resolution L20 - Referral to SC22 of Processing Ideographic Languages  
individual vote: 34-0-0      country vote: 9-0-0

That WG5 requests its convenor to report to SC22 the concerns of WG5 that projects in the program of work of SC22 and related committees allow for the processing of ideographic languages such as Chinese and Kanji in a consistent and efficient manner.

X3J3 Response. X3J3 concurs with this resolution, and notes that it requires no action on X3J3's part.

---

Resolution L21 - Use of National Characters  
individual vote: 31-0-4      country vote: 9-0-0

That WG5 expresses to X3J3 its concern about the negative effect on the production of standard-conforming processors if characters in the national use positions in ISO 646, such as square brackets, are required in the Fortran 8x character set.

X3J3 Response. Assigned to the General Concepts subgroup.  
*(see text at bottom of next page)*

---

Resolution L22 - Bit Data Type  
individual vote: 22-3-10      country vote: 7-0-2

That WG5 believes that there is a significant unsatisfied demand for a bit data type facility in Fortran and that the need for such a facility will tend to increase during the lifetime of Fortran 8x. It, therefore, recommends that X3J3 review its earlier decision to remove BIT from 8x.

X3J3 Response. Brian Smith and Jerry Wagener assigned to investigate providing bit functionality with the logical data type, and prepare a report for the Data Concepts subgroup. A tutorial on parameterizing the logical data type to provide the Appendix F bit functionality was presented and discussed at X3J3 meeting #105 (Liverpool, August 1987). This issue was scheduled for further discussion at X3J3 meeting #106 (Ft. Lauderdale, November 1987); that discussion was limited to the transmittal of the Smith/Wagener report (X3J3-106-59) to the Data Concepts subgroup. A formal roll call vote on bit data type, requiring a two-thirds majority, failed 16-13-2. This vote was taken in the context of discussions on appropriate times to pursue major technical issues, and consequently it was decided not to pursue bit data type ~~during the public review period.~~ *until after receiving the public review comments.*

---

Resolution L23 - Passed-On Precision

individual vote: 25-2-8      country vote: 8-0-1

That WG5 draw the attention of X3J3 to the concerns of the German member body (DIN) about passed-on precision contained in paper N245.

X3J3 Response. Assigned to Brian Smith.

---

Resolution L24 - Range and Set Range

individual vote: 12-10-12      country vote: 4-3-2

That WG5 recommends to X3J3 that the RANGE and SET RANGE facilities be deleted from the language.

X3J3 Response. Assigned to the Data Concepts subgroup.

---

end 107.JLW-2

L21 Response

*At meeting #106 (Ft. Lauderdale, November, 1987) X3J3 discussed documents 106-54, 106-98, and 106-119. A motion to remove the use of square brackets in Fortran 8x (document 106-119), requiring a two-thirds majority, failed 16-10. The chair ruled (ruling upheld by a vote of 23-1) that another proposal on this issue be considered at meeting #107.*

84

TO: X3J3  
FROM: Paul L. Sinclair  
SUBJECT: SCRATCH Files  
DATE: November 18, 1987

In the following, references are to Draft S8, Version 104. References to the document are in the form: p/l or p/l-l where p is a page number (for example, 4-5), and l is a line number. Material copied from the draft is in quotes. Items in apostrophes are suggested rewordings.

Background:

It is not clear which of paragraphs 9-6/10-12, 9-6/16-18, and 9-6/19-25 has priority. Specifically, what happens under the following conditions:

1. The unit is connected (or preconnected) to an existing file just prior to the execution of the OPEN statement.
2. A FILE= specifier is not included in the OPEN statement.
3. The STATUS= specifier has a value of SCRATCH.

Does the file remain connected to the unit as stated in 9-6/10-12 and therefore if the connected file is not a SCRATCH file, the OPEN fails? Or, does an implicit close occur because the file to be connected is not the same as the file to which the unit is connected as stated in 9-6/16-18? <sup>YES</sup> <sup>YES</sup>

When SCRATCH is specified, does this always mean the file to be connected is not the same as the file to which the unit is connected? If so, how does one change BLANK=, DELIM=, PAD= for a SCRATCH file? If STATUS= specifier is omitted, then the default value of the STATUS= specifier (UNKNOWN) is not the same as SCRATCH which according 9-6/19-25 is not allowed.

One possible conclusion may be that to open a SCRATCH file on a unit that is connected (or preconnected) to an existing file, an explicit CLOSE must occur to the unit prior to the OPEN. That is, there is no way to cause an implicit close when trying to open a SCRATCH file since SCRATCH can't have FILE= (and therefore the connection stays the same) but then you can't have a STATUS which is different (9-6/19-21). This doesn't seem very nice. When you have a FILE= the connection can be implicitly closed, but when trying to OPEN a SCRATCH file, there is no way to implicitly close the connection.

Another possible conclusion may be that to DELIM=, PAD=, and PAD specifiers cannot be changed for a SCRATCH file.

Pertinent References:

1. 9-6/10-12

"If a unit is connected to a file that exists, execution of an OPEN statement for that unit is permitted. If the FILE= specifier is not

P. 618  
PLS

included in such an OPEN statement, the file remains connected to the unit."

84

2. 9-6/13-15

"If the file to be connected to the unit does not exist but is the same as the file to which the unit is preconnected, the properties specified by the OPEN statement become a part of the connection."

3. 9-6/16-18

"If the file to be connected to the unit is not the same as the file to which the unit is connected, the effect is as if a CLOSE statement without a STATUS= specifier had been executed for the unit immediately prior to the execution of the OPEN statement."

4. 9-6/19-25

"If the file to be connected to the unit is already connected to the unit, only the BLANK=, DELIM=, PAD=, ERR=, and IOSTAT= specifiers may have a value different from the one currently in effect. Execution of such an OPEN statement causes any new value of the BLANK=, DELIM=, or PAD= specifiers to be in effect, but does not cause any change in any of the unspecified specifiers and the position of the file is unaffected. The ERR= and IOSTAT= specifiers from any previously executed OPEN statement have no effect on any currently OPEN statement."

5. 9-6/26-27

"If the file is already connected to the unit, execution of an OPEN statement on that file and a different unit is not permitted."

6. 9-18/18-22. After a successful OPEN, the file exists.

7. 9-15/5-6. After a successful WRITE or PRINT, the file exists.

8. 9-7/26. referring to STATUS=, "If this specifier is omitted, the default value is UNKNOWN."

Proposal:

the second sentence in

to be.

is the same as the file to which the unit is connected

1. replace 9-6/10-12 with

~~"If a unit is connected to a file that exists, execution of an OPEN statement for that unit is permitted. If the FILE= specifier is not included in such an OPEN statement, and the STATUS= specifier does not have a value of SCRATCH, the file remains connected to the unit. If the STATUS= specifier does have a value of SCRATCH in such an OPEN statement and the status of the file connected to the unit is SCRATCH, the file remains connected to the unit and the file to be connected is the same as the file already connected. If the STATUS= specifier does have a value of SCRATCH in such an OPEN statement and the status of the file connected to the unit is not SCRATCH, the file to be connected is not the same as the file already connected."~~

84

2. 9-6/19-21. Replace sentence "If the file ... currently in effect." with

'If the file to be connected to the unit is the same as the file to which the unit is connected, only the BLANK=, DELIM=, PAD=, ERR=, and IOSTAT= specifiers may have a value different from the one currently in effect.'

P. 620

To: X3J3  
From: John Reid  
Date: 10 Feb 88  
Subject: S8 Edits

107-JKR-4

Proposal Make the following changes to S8:

1. Page 13-41, lines 25 to 38. Replace by:

Result Value. The value of the result is zero if no character of STRING is in SET. or if the length of STRING or SET is zero. otherwise:

Case (i): If BACK is absent or is present with the value .FALSE., the value of the result is the position of the leftmost character of STRING that is in SET.

Case (ii): If BACK is present with the value .TRUE., the value of the result is the position of the rightmost character of STRING that is in SET.

Examples. SCAN ('FORTRAN', 'BCD') has the value 0.

Case (i): SCAN ('FORTRAN', 'TR') has the value 3.

Case (ii): SCAN ('FORTRAN', 'TR', BACK=.TRUE.) has the value 5.

2. Page 14-5, line 15. Change 'entity' to 'associated nonalias object'.

3. Page B-6, line 20+. Add the paragraph:

There are, of course, other uses of the computed GO TO statement that do not translate directly



into a CASE construct (for instance, one cannot 'drop through' from one case to the next). It is believed, however, that straightforward use of the CASE construct, the IF construct, the unconditional GO TO statement, and the internal procedure will result in a program that is easier to read and maintain.

4. Page 5-19. Move lines 43-44 to line 31+ and make them a constraint.
5. Page 7-7, line 42. Delete last 'or' and add  
 (6) A subobject of a constant whose each subscript, section subscript, substring starting point, and substring ending point is a constant expression, or
6. Page 7-8, line 1. Change '6' to '7'.
7. Page 7-11, line 48+. Add  $(X+Y)+Z$        $X+(Y+Z)$
8. Page 7-15, line 9. 'rel-op' should be in italic.
- ~~9. Page 11-3, line 6. Change '[only list]' to 'only-list'.~~
10. Page 12-8, line 1. Roman font for '0.0', '1', '2', '1', '2'.
11. Page 12-9, line 2+. Add 'A function subprogram is a subprogram whose first statement is a FUNCTION statement!.
12. Page 12-10, line 12+. Add 'A subroutine subprogram is a subprogram whose first statement is a SUBROUTINE statement!.
13. Page 12-13, line 47 and page 12-14, lines 1, 4, and 16. Change 'expr' to 'scalar-expr'.

To: X3J3  
From: Ed. Comm.  
Subject: Edits for S8.104

86<sup>d</sup>

107(x) LWC-5  
Feb. 10, 1988  
Page 1 of 2

862

1. Delete all constraints in Section 2:

- P. 2-1/4, after "1.5." add "Note that some of the syntax rules in this section are subject to constraints which are given only at the appropriate places in later sections."
- P. 2-1/26-29, delete constraint. (was copied to P. 12-9/18+ & 12-10/26+)
- P. 2-1/34-36, delete constraint. (is at P. 11-2/29-31)
- P. 2-1/40-42, delete constraint. (is at P. 11-5/45-47)
- P. 2-3/1, delete constraint. (expanded version is at P. 5-11/3-9)
- P. 2-3/2-4, delete constraint. (see P. 5-10/30-31 & 5-10/39-40)
- P. 2-3/38-39, change "executable construct" to "executable-construct" and move constraint to P. 12-11/17+. (delete 2-3/38-39)
- P. 2-3/40, delete constraint. (see P. 12-11/34)

2. P. 5-1/4, change "Collectively these properties, including the type," to "Collectively, these properties (including the type)".

3. P. 7-10/23, change "a subscript" to "the array element reference" and in line 24, change "the subscript" to "its subscripts".

4. P. 1-3/17, add sentence after "needed.": "Some rules in Sections 2 and 3 are more fully described in later sections; in such cases, the section number s is the number of the later section where the rule is repeated."

5. P. 10-13/31, change "a name" to "an object name or a subobject designator".

~~6. Delete unnumbered pages between Appendices D and E. (delete BPF cross reference list)~~

7. P. 9-2/8-10, change "processor-determined" to "processor-dependent". Also in item 1 of 107(x) PLS-2 (34).  
Also at P. 9-2/14, 9-5/21, 9-7/15, 9-15/10, & 9-15/12.

8. P. 9-10/42, delete constraint. (covered by line 41 constraint)

9. p. 4-3/32 and 36, change "processor-defined" to "processor-dependent". (See 5-3/16, 19 & 42)
10. p. 5-8/41, delete "array".
11. p. 10-15/28-33, move lines 28-33 to follow line 41.  
(See p. 10-12/2-17)
12. p. 10-12/4, change "characters" to "value separators".  
(See p. 10-15/35)
13. p. 10-12/6, change "datum" to "character constant".  
(See p. 10-15/37)
14. p. 10-15/40-41, replace lines 40-41 with p. 10-12/9-11.  
(clarify namelist rules and make same as list-directed)
15. p. 10-15/18-24, replace "when the ... next record." with  
with p. 10-11 lines 45-48. (Simplify namelist text  
and make same as list-directed text.)
16. p. 13-20, lines 18-19, 21-22, and 24, replace "has value"  
with "has the value". Also at 13-11/31 & 39,  
13-19/7, 13-28/14, 13-29/7 (twice).
17. In 107(20) RAH-1 (36) paragraph 3.3.1.3.2,  
replace "an" at beginning of third line with  
"must not be followed by commentary. An"  
and delete the last sentence "The "&" ...  
commentary." (to clarify which & must  
not be followed by commentary)
18. p. 4-3/15, add "exponent" before "range".
19. p. 12-7/1, change "or" to "for".
20. p. 5-13/19, change "loop" to "construct".
21. Appendices A, C, D, E, and G need a period after  
the appendix letter in appendix title.

3: X3J3  
 From: John Reid  
 Subject: Edits  
 Date: Feb 11, 1988

Proposal: Make the following changes to S8:

1. Page 5-3. On line 42, delete 'real' and on line 44 after 'agree', add 'The default value is the same as the default value for the same precision selector in a REAL type specifier.'
2. Page 7-9, lines 24 and 25-26. Change 'specification sequence' to 'specification-part', twice.
3. Page 7-10, line 25. Change 'range' to 'expressions'.
4. Page 8-1, line 15. Change 'empty.' to 'empty; execution of an empty block has no effect'.
5. Page 8-2, line 24. Add 'Execution of an END IF statement has no effect'.
6. Page 12-9, lines 22-23. Change 'supplied' to 'present' and 'agree with ... on' to 'be identical to the function-name specified in'.
7. Page 12-10, lines 30-31. Change 'agree with ... on' to 'be identical to the subroutine-name specified in'.
8. Page 14-8, line 33. Add comma after 'defined'.
9. Page A-5, line 3. Delete '(SD-3)'.
10. Page 13-1, line 39. Change 'TRIM function' to 'transformational function TRIM'.

11. Page B-3, lines 3-6. Replace by

'(3) As deprecated features fall into disuse, it is recommended that future Fortran standards committees move these features from the deprecated list to the obsolescent list.' (106-62, item 28)

12. S16.107, modified by <sup>approved changes</sup> ~~the proposals~~ in 107-47a (JKR-3a) and 107-79a (LWC-4a), records the editorial <sup>changes</sup> ~~corrections~~ to S8.10<sup>4</sup> made before the start of this meeting.

13. Page 6-8, lines 33-33½, as modified by S16.107. Replace by

R628 parent-array-element is object-name [(mapping-subscript-list)]  
■ [% component-name [(mapping-subscript-list)]]...

Constraint: ~~Each~~ <sup>A</sup> mapping-subscript-list must follow the name of ~~each~~ object or component that is array-valued. A mapping-subscript-list must not follow the name of an object or component that is scalar-valued.

(107-13; GP-2)

14. Page 7-7, line 13+. Add 'The term unspecifiable precision is used because such precision cannot be specified for a named entity. An example is the expression 2.0 \* E where E has been declared by the statement  
REAL (6, 50), PARAMETER :: E = 2.7182818  
because 2.0 is default Real and E is specified precision.'

15. Page 9-21, line 22. After 'connection' add 'or if the file is connected for direct access'. [CIO group: please check]

To: X3J3

From: Larry Rolison (on behalf of DATA)

Subject: PUBLIC vs. PRIVATE fix-up with type definition

Background: Public comment item 73.31 asked the question (DATA thinks): If a TYPE statement of a derived-type definition contains a PRIVATE access spec, what does it mean when a variable declared to be of this type has the PUBLIC attribute? We believe the question uncovered a hole in 58. Our intent is to prevent this conflict.

Proposal:

p. 5-2, line 15: add the constraint

Constraint: If the derived-type-stmt in the derived-type definition of type-name contains a PRIVATE access-spec, the entity being declared must not have the PUBLIC attribute.

Discussion:

- \* Note that the PUBLIC attribute may be conferred in a number of ways.
- \* Since the default accessibility within a module is PUBLIC, if a user has a derived-type definition in such a module where the TYPE statement has PRIVATE, the user will have to add the PRIVATE attribute to each declaration referencing the type.
- \* The case where the type definition contains a PRIVATE statement but a variable with the PUBLIC attribute is declared referencing the type already exists (and is OK - it is the basis of "object-oriented" programming).