Fortran 8x, the new Fortran standard

J K Reid

Computer Science and Systems Division
Harwell Laboratory, Oxfordshire OX11 0RA

September 1987
Fortran 8x, the new Fortran Standard

J. K. Reid

Abstract

The ANSI Fortran Standardization Committee, X3J3, has been working since 1977 to design the successor to Fortran 77 and the author has been a full member since early 1983, representing the whole of the UK Atomic Energy Authority and reporting to a small committee with one member from each principal site. A draft of the new standard is about to be released for public comment, and this paper will be presented at each site. It summarizes the recent history of the project, the facilities in the draft language and the comments that have been made on it.

Following these presentations, the Authority Fortran Committee will formulate its comments on the draft.

Computer Science and Systems Division,
Harwell Laboratory
September 1987

HL87/1443 (C13)
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Language evolution</td>
<td>2</td>
</tr>
<tr>
<td>3.  Array operations</td>
<td>2</td>
</tr>
<tr>
<td>4.  Derived data types</td>
<td>3</td>
</tr>
<tr>
<td>5.  Modules</td>
<td>3</td>
</tr>
<tr>
<td>6.  Procedures</td>
<td>4</td>
</tr>
<tr>
<td>7.  Generalized precision</td>
<td>4</td>
</tr>
<tr>
<td>8.  Miscellaneous improvements</td>
<td>5</td>
</tr>
<tr>
<td>Appendix A. Obsolescent and deprecated features</td>
<td>6</td>
</tr>
<tr>
<td>Appendix B. Summary of the June 1986 compromise</td>
<td>7</td>
</tr>
<tr>
<td>Appendix C. Features not in Fortran 8x</td>
<td>8</td>
</tr>
<tr>
<td>Appendix D. The no votes</td>
<td>8</td>
</tr>
</tbody>
</table>
1. Introduction

Fortran was the first computer language to be standardized, in 1966, and the standard was revised in 1977. These two languages are known informally as Fortran 66 and Fortran 77. During the comment period for Fortran 77, there were 1225 pages of public response, the largest outpouring of commentary in any standard review up to that time. It was felt that a third revision should be launched at once, since so many features were rejected as premature in 1977. The resulting language is known informally as Fortran 8x, where x is an unknown. It was originally hoped that x would have the value 2.

The language evolved until February 1986, when a letter ballot of the members was held to ask whether they thought that the draft was ready for release to the public for comment. The result was yes:16, no:20, and one did not vote. The rules require each no vote to be accompanied by reasons and the Committee is required to address each such reason with the aim of achieving consensus. A number of non-controversial points were raised, but the main problems were with the overall size of the language and with the mechanism for language evolution. A compromise that involves a reduction in the size of the language and slower language evolution was agreed in June 1986 (see Appendix B).

The draft was revised to reflect these changes and a fresh postal vote of X3J3 members was held in February 1987. The result was 29-7. Further revisions were made in the light of comments submitted with these ballots and the committee voted 26-9 at the May meeting in favour of releasing the draft for public comment. Because of the no votes, this does not happen automatically, but SPARC (Standards Planning and Requirements Committee) voted that the work of X3J3 accords with its charter and X3 (the parent committee of X3J3) voted 31-4 (30-5 in the final reconsideration ballot) in favour of release for public comment.

The public comment period will last four months and will probably start in October 1987. Official copies of the draft Standard will be available from Global Engineering Documents, Inc. by calling (714) 540-9870 at a price that will probably be about $50. Copies of the document forwarded to X3 are available for loan within the Authority and C.E.G.B. Apply to G.H. France (C.E.G.B., Park St), P. Grayson (N.N.), P. Kirby (Culham), C.K. MacKinnon (Risley), J.K. Reid (Harwell), or M.J. Roth (Winfrith), who are members of the Authority Committee. For an informal description of the language, see Fortran 8x Explained by Mike Metcalf and John Reid, to be published in November by OUP (£10.95 in paper-back form).

From now on, any change to the draft will require a 2/3 majority of those X3J3 members voting and an absolute majority of all members. This rule will mean that changes really do need a consensus of the Committee. Where opinion is divided (for example on significant blanks) we are likely to keep what we now have.

This paper summarizes the language under seven main headings. The no votes are summarized in Appendix D, together with a brief summary of the position of the majority that voted yes.
2. Language evolution

A program that conforms to Fortran 77 will conform to Fortran 8x. Thus a Fortran 8x processor is a Fortran 77 processor with extensions. A small number of features that have replacements in Fortran 77 are labelled ‘obsolete’ and may become ‘obsolete’ in Fortran 9y, that is will not be present in Fortran 9y. A larger list of features are labelled ‘deprecated’ and may become obsolete in Fortran 9y and obsolete in Fortran 10z. The features involved are listed in Appendix A.

Since there has historically been an 11-year revision cycle, this allows over 20 years for programs to be modified to remove deprecated features or to fall into disuse.

3. Array operations

Arrays may be ‘automatic’ (automatically allocated on entry to a procedure and deallocated on return) or ‘allocatable’ (allocated and deallocated by explicit statements). Dummy arrays may be ‘assumed-shape’ (take their shapes from the corresponding actual arguments). Arrays may be of size zero.

Arrays may be used in whole-array expressions such as

\[ B + C \times \sin(D) \]

The operations are performed element-by-element, that is the sine function is applied to each element of \( D \), multiplied by the corresponding element of \( C \), and added to the corresponding element of \( B \). The arrays must have exactly the same shape, but scalars may be intermixed freely. Temporary array storage may be needed for intermediate results. Array expressions may be used as actual arguments. They may be used in whole array assignments provided there is exact shape agreement.

Rectangular subarrays, called sections, may be used as arrays. Examples are \( A(:,7) \) which is the 7-th column of \( A \) and \( A(2:10:2,7) \) which consists of components 2, 4, 6, 8, 10 of the 7-th column of \( A \).

Functions may be array-valued. Scalar functions may be called ‘elementally’ in the way \( \sin \) was called in the above example. There are many new inquiry intrinsics that return the array properties of their arguments and many new array-valued intrinsics, for example \( \text{MATMUL} \) for matrix multiplication, \( \text{MAXVAL} \) for the largest element, and \( \text{SUM} \) for summation.

The executable statement \text{IDENTIFY} provides access through an alias name to a section that may be skew. For example,

\[ \text{IDENTIFY} \ (\text{DIAG}(I)=A(I,I), I=1,N) \]

permits the diagonal of \( A \) to be accessed as a rank-one array called \( \text{DIAG} \).

Arrays of rank one may be constructed as lists of scalars and other arrays of rank one, possibly repeated. An example of any array constant of size 10 is \([1.0, 2.7, 4[1.0, 2.0]]\). In the integer case, triplets such as \( 1:10:2 \) may be included and are interpreted as for \( \text{DO} \) indices. There is a \text{RESHAPE} intrinsic function to allow arrays of other shapes to be constructed.
WHERE statements allow array assignment statements to be masked. For example

\[
\text{WHERE (A \cdot \cdot \cdot O) B=LOG(A)}
\]

causes the evaluation and assignment of logarithms only for elements that are positive. There is also a block form with an optional ELSEWHERE block.

Arrays may be ‘ranged’, in which case whole array statements refer to a subarray that is set and reset by execution of SET RANGE statements.

4. Derived data types

Fortran 8x permits the construction of derived data types whose values have components that are of intrinsic type or of another derived type. Functions may be used to define operations on such types of data and subroutines may be used to define assignments between them. The operators may be intrinsic (e.g. +, *,.,EQ.), in which case the existing priorities are used for the new operators, or nonintrinsic (e.g. .MERGE.), in which case the priority is maximum for unary operators and minimum for binary operators.

An example of a type useful for sparse matrices is

\[
\text{TYPE NONZERO}
\]
\[
\text{REAL VALUE}
\]
\[
\text{INTEGER ROW, COLUMN}
\]
\[
\text{END TYPE}
\]

which allows a matrix nonzero and its row and column numbers to be treated together. A sparse matrix with up to 100 nonzeros may be held in an array declared thus

\[
\text{TYPE (NONZERO) :: A (100)}
\]

The symbol % is used for components of data of derived type. For example, \(A (10) \% \text{VALUE}\) would be the value of the nonzero stored in \(A (10)\). A value may be constructed as a list of component values. For example, we might set \(A (10)\) thus

\[
A (10) = \text{NONZERO (6.0, 10, 7)}
\]

Derived data types may have integer parameters. For example,

\[
\text{TYPE MATRIX (M, N)}
\]
\[
\text{REAL R (M, N)}
\]
\[
\text{END TYPE}
\]

might be used for \(m \times n\) matrices. Indeed, if matrix arithmetic is wanted, some such type would be needed.

Derived data provides the language with a powerful form of extendibility. It means that ordinary in-line operator notation will be available for matrices, extended precision arithmetic, interval arithmetic, etc.

5. Modules

Modules are collections of data, type definitions, and procedure definitions. For example
MODULE JKR
  TYPE MATRIX (M, N)
  REAL R (M, N)
END TYPE
  TYPE (MATRIX (10, 10)) :: A, B, C
CONTAINS
  FUNCTION ADD (A, B) OPERATOR (+)
    TYPE (MATRIX (*, *)) A
    TYPE (MATRIX (M(A), N(A))) B, ADD
    ADD%R = A%R + B%R
  END FUNCTION
END MODULE

is a module containing the definition of the type MATRIX, three matrices (A, B, and C), and
the definitions of matrix operations. Entities may be declared PRIVATE, in which case they
are available within the module, but not outside it.

A simple USE statement such as

USE JKR

provides accesses to all the public entities in the module. To allow for possible name clashes,
there is a renaming facility. Access may be limited to a list of entities.

Modules provide a safe replacement for COMMON. Note that the definitions are given only
once. It is more general than COMMON in that type and procedure definitions are included.

6. Procedures

A procedure may be called recursively provided its leading statement includes the qualifier
RECURSIVE. It may be internal, at one level only, to an external procedure or to a procedure
in a module. Keyword calls, as in the i/o statements of Fortran 77, are available. The dummy
argument names serve as keywords. Arguments may be omitted provided they are declared
as OPTIONAL. The intrinsic function PRESENT may be used to inquire whether an optional
argument is present. Dummy arguments may be declared to be IN, OUT, or INOUT. Several
procedures may have the same name (overloading) provided they may be distinguished by
the types or ranks of their arguments.

Interface blocks that contain statements just like the leading statements of a procedure may
be used to specify the interface to an external or dummy procedure. For example, this permits
keyword calls to be made to a procedure written in assembly language. If a procedure is
intrinsic, internal, is accessed from a module, or has an interface block, it is said to have an
‘explicit’ interface. An explicit interface is required to call an array-valued function, to call a
procedure with assumed-shape dummy arguments, to call a procedure as an operator, or to
make a keyword call. In all these cases, the interface information is needed so that the system
can set up the linkage before run-time.

7. Generalized precision

REAL and COMPLEX declarations may be parameterized to indicate the desired precision
and exponent range. For example

REAL (10, 99)
takes the shortest machine representation that gives the equivalent of at least 10 significant
decimals and a range of at least $10^{499}$.

There are many inquiry and manipulation intrinsic functions that return information on the
representation or manipulate parts of data (e.g. extract the fractional part).

Constants may be specified with the help of a letter other than E, D, or H that is associated
with a particular precision and exponent range pair, as in the example

\[
\begin{align*}
\text{REAL (7, 50) PI} \\
\text{EXPONENT LETTER (7, 50) L} \\
\text{PI} = 3.141592654L0
\end{align*}
\]

In a binary expression such as A+B, the precision of the result is the greater of the
precisions of A and B and the exponent range is the greater of the exponent ranges. For a
general expression, the precision and exponent range may be deduced from its expansion into
a sequence of binary operations and the rule for each binary operation.

In a procedure call, the declared precisions and exponent ranges of the dummy and actual
arguments must match. A dummy argument may be declared as REAL(*, *) or
COMPLEX(*, *) and such an argument automatically matches the corresponding actual
argument provided all such actual arguments in a given call have the same precision and the
same exponent range. In effect, one version of the source therefore serves for all possible
precisions.

8. Miscellaneous improvements

(a) Conformance. The definition of conformance remains as in Fortran 77. A program
conforms to the standard if it ‘uses only those forms and relationships described in the
standard and has an interpretation according to it’. A processor conforms if it ‘executes
standard-conforming programs in a manner that fulfills the interpretations prescribed in
the standard’. Thus a processor is permitted to have extensions. However, it is now
required to be able to detect the use of any extension to the syntax, or any deprecated or
obsolete syntax. Although the draft does not use these words, the intention is that
such checks will be made at compile time rather than at run time.

(b) Input-output. The only major new feature is NAMELIST, but there are a large number of
minor enhancements.

(c) Source form. Names may contain up to 31 characters, including _, which is regarded as
alphanumeric. The Fortran character set is extended to include

\[ ! " \& ; < > ? [ ] _ \]

There is a free source form, incompatible with the Fortran 77 form, that attaches no
particular significance to columns 1 to 6 or 72 onwards, allows end-of-line comments
(using ! as a separator), allows several statements on a line (using ; as a separator), and
uses a terminating & to indicate continuation to the next line. Lines may have length up
to 132 characters and statements up to 2640 characters.

(d) Operator spelling. The operators .LT., .GT., .LE., .GE., .EQ., and .NE. have the
alternative spellings <, >, <=, >=, ==, and <>.
(e) Control structures. There is a CASE construct, exemplified by the following

```
SELECT CASE (N)
   CASE (:0) ! N ≤ 0
   :
   CASE (1) ! N=1
   :
   CASE (5:7) ! N=5, 6 or 7
   :
   DEFAULT ! Any other value
   :
END SELECT
```

There is a form of the DO loop that does not use labels, exemplified by the following

```
OUTER: DO ! Unlimited DO, named OUTER
   :
   DO (N TIMES) ! N executions
   :
   DO I=1,N ! I=1,2,...,N
   :
      IF(...)EXIT OUTER! Possibly exit
      ! outer loop
   END DO
   IF(...) CYCLE ! Skip to end of loop
   :
END DO
END DO OUTER
```

(f) Intrinsic subroutines. There are intrinsic subroutines to provide the date and time as up to 9 integers, to provide information on the real-time clock, and to provide random numbers.

(g) Intrinsic functions. There are a large number of new intrinsic functions and all the Fortran 77 intrinsics are elemental, that is they may be called for array arguments and are then applied to every element.

(h) Data statement. There is a new form for the DATA statement that associates data values more obviously with data objects, for example

```
DATA ( I=1, S=7.9, (A(K), K=1,7,2)=[4[1.2]])
```

Appendix A. Obsolescent and deprecated features

The following features are obsolescent:–

(1) Arithmetic IF
(2) Noninteger DO index
(3) DO termination other than on a CONTINUE or END DO statement
(4) Branching to END IF from outside its block
(5) Shared DO termination
(6) Alternate return
(7) PAUSE
(8) ASSIGN and assigned GO TO
(9) Assigned FORMAT specifiers

The following features are deprecated because they involve storage association:—

(1) Assumed-size dummy arrays
(2) Passing an array element to an array
(3) BLOCK DATA
(4) COMMON
(5) EQUIVALENCE
(6) ENTRY

The following features are deprecated because they involve redundant functionality:—

(7) Old source form
(8) Specific names for intrinsics
(9) Statement function
(10) Computed GO TO
(11) The old form of the DATA statement and allowing DATA statements among executable statements
(12) DIMENSION
(13) DOUBLE PRECISION
(14) CHARACTER*len

Appendix B. Summary of the June 1986 compromise

(1) Move the following features to an appendix:—

(a) Treating arrays of arrays as arrays
(b) Condition handling
(c) Intrinsic functions DIAGONAL, RANK, REPLICATE, PROJECT, FIRSTLOC, LASTLOC
(d) Nested internal procedures
(e) Passing internal procedures as arguments
(f) FOR ALL
(g) Vector-valued subscripts
(h) Variant structures
(i) BIT type

(2) Simplify internal procedures, modules, and USE statement
(3) Add bit transfer capability
(4) Remove significant blanks and insignificant underscores
(5) Replace name-directed i/o with NAMELIST
(6) Simplify IDENTIFY to have

(a) only one host
(b) no many-one mappings
(c) canonical form for map

(7) Split deprecated features into two
Appendix C. Features not in Fortran 8x

Fortran 8x does not include a pointer facility, although this is wanted by a significant number of users of the language and there appears to be a concensus among Committee members in favour. Unfortunately, the Committee is not unanimous over the exact form it should take, but is likely to perform some parallel processing by working on it during the public comment period.

Fortran 8x does not include a BIT type. It used to have something extremely like LOGICAL. A case has been made for parameterizing LOGICAL so that BIT would be spelt as LOGICAL(1), say.

Fortran 8x does not include facilities for multiprocessing. It is felt that we do not yet know the best way to do this, so it is too early to standardize. Note, however, that in Fortran 77 the result of a function must not depend on the order in which its arguments are evaluated. Therefore functional programming may be performed in Fortran 77 and could be implemented with parallel execution of argument evaluations. A simple example is to break a summation into two parts thus

\[
\text{SUM1 ( SUM2 (A, N/2), SUM3 (A(N/2+1), N-N/2) )}
\]

The Japanese are making a strong case for adding a feature that can accommodate Kanji characters, which occupy two bytes of storage. The request did not come until May this year, so there is nothing in the draft, but the Committee is looking serious at possibilities. Of course, there are other countries that need such facilities.

Appendix D. The no votes

The nine no votes at the May meeting were:

Adamczyk (Adv. Comp. Tech.) wants the draft to be limited to existing practice and to contain more explanatory text to assist public comment.

Allison (Harris) has serious misgivings and is concerned that the 2/3 rule will make it too difficult to make changes during the public comment period.

Bowe and Rolison (Unisys) found virtually no new feature that will enhance either compiler or run-time performance and were concerned that the committee may have created a disincentive for the average user to move to an 8x implementation.

Harris (DEC) wants

(i) derived types to be removed and new intrinsic types to be added for bit strings, general pointers, and varying-length character strings;

(ii) to disallow modules to contain procedures or access other modules; to add an INCLUDE facility;

(iii) to remove generalized precision; to require minimal accuracy standards for operations and intrinsic functions; and

(iv) to reduce the obsolescent features list to noninteger DO indices, ASSIGN, assigned GO TO, and assigned FORMAT specifiers; to reduce the deprecated list to one item, the old source form.
Marusak (Los Alamos) wants to add BIT type and is concerned that the 2/3 rule may make this impossible.

Moss (SLAC) feels that the draft is not yet in a sufficiently understandable and polished form and is concerned about the 2/3 rule.

Philips (Boeing) is concerned

(i) about the expense of converting programs to avoid the obsolescent and deprecated features;
(ii) the difficulties that casual users may have with a more complex language having redundant features,
(iii) that major vendors have predicted poor performance for compilation and execution of compiled code;
(iv) that portability may impacted during the transition period of partial implementations; and
(v) that the language may be too complex for implementation on PCs.

Wearing (NCC, Manchester) feels that the draft is not sufficiently well written.

Weaver (IBM) wants Fortran 8x to be reorganized into two languages. One would be called Fortran and would consist of Fortran 77 plus data structures, array language as an option, bit string data type, dynamic allocation, environmental inquiry functions, precision specification, keyword and optional arguments on CALL statements, IMPLICIT NONE, and the new form of the DO. The rest would be processed as a technical report to allow implementation experience, testing and evaluation.

The four no X3 votes were:

Data General wants Fortran 8x to be based on Fortran 77, incorporate popular vendor extensions (such as Hollerith; NAMELIST; INCLUDE; bit, byte, and logical data types; word and bit manipulation routines from MIL-STD 1753), avoid obsolescent and deprecated features, and remain a small and efficient language.

DEC believes that most of the proposed new features are of little interest to its users and that several important improvements have been omitted. It wishes to remove the array features; replace the generalized precision facilities with accuracy requirements; remove derived data types and add intrinsic data types for pointers, bits, varying-length character strings, and multi-byte characters; remove the deprecated features. It also complains about procedural issues within X3J3, essentially saying that is is improper to go to public comment against the wishes of some of the members.

IBM thinks that the draft does not preserve the essential characteristics of Fortran as a relatively small language that permits efficient implementations, and that it does not protect the users investment in existing programs. It wants the changes suggested by Weaver with his no vote, plus a multi-byte character type.

UNISYS believes that the new features will increase compilation cost, program size, and object code execution time; require re-education of traditional Fortran users; and that dynamic allocation of storage will force new constraints on operating systems.
The essence of the argument is that a substantial majority of the Committee wish to see a major revision of the language, which they see befitting a revision after more than ten years. They also wish to see a mechanism for evolving the language, while recognizing that the investment in existing software written in Fortran is so huge that deletions must be made slowly. The minority view is that the revision should be much more conservative and few or no features should ever be removed from the language. The Committee has made enormous efforts to reach consensus, and it has become clear that this is impossible. Removal of any more of the extensions would lead to the creation of more no votes that it would remove. The extensions have all been included in response to some request that the Committee has been convinced is genuine and important, so removing many would mean that it was not responding to requests from the public. It is very important that Fortran users look carefully at the draft during the public comment period and tell the Committee what they think.