LA-UR-7?-111?

TITLE: A "CORE + MODULES" APPROACH TO FORTRAN STANDARDIZATION

AUTHOR(S): WALT(ER) (SCOTT) BRAINERD

SUBMITTED TO: ACM 79
[illegible]

University of California

LOS ALAMOS SCIENTIFIC LABORATORY
Post Office Box 1663 Los Alamos, New Mexico 87545
An Affirmative Action/Equal Opportunity Employer

UNITED STATES
DEPARTMENT OF ENERGY
CONTRACT W-7405-ENG. 36

A "CORE + MODULES" APPROACH TO FORTRAN STANDARDIZATION

by

Walt Brainerd
Los Alamos Scientific Laboratory

In January 1978, ANSI X3J3 voted to adopt a framework consisting of a "core" and "modules" for developing the next revision of the ANSI Fortran standard. Of course, this is a decision that could be reversed if the approach appears to be unsuitable or technically unsound after some experimentation. However, the approval of this procedure is an indication that the committee wants to invest considerable effort in an attempt to make this approach work.

There are at least three reasons for adopting the "core + modules" approach:

(1) to provide a mechanism to interface with collateral standards and implementations in major applications areas,

(2) to provide a mechanism for having optional functional areas described within the standard,

(3) to specify a smaller, more elegant language than Fortran 77 without decreasing the status of Fortran 77 as a standard language.

Each of these reasons is now discussed in more detail.

One of the major concerns of X3J3 is the development of collateral standards in areas such as data base management, real-time process control, and graphics. X3J3 does not have the resources to do the technical development of standards in all of these areas; in some cases X3J3 may not even be involved directly in the approval of such a standard. It is important, therefore, that X3J3 provide a scheme whereby collateral standards in these applications areas can be regarded as modules in the language that are "attached" to the core of the language in a standard way. The only mechanism considered so far for interfacing with these modules is through the CALL statement, extended to allow the arguments to be identified by key words. This topic is covered in more detail in another paper and is an area that can use more good ideas, because it is a very difficult and important problem.

A second kind of extension might be called a "language feature module." This sort of module would include a collection of related language features that might not be appropriate to include in the core, but which should have its form specified so that all extensions to the core in this area will be the same. Examples of candidates for such modules are array processing, a bit data type, and specification of numerical precision. Fortran 77 should be considered to be such a module. It may be quite inappropriate to add some of these language features to Fortran 77. For example, it would be rather messy to add a bit data type or REAL*11 (indicating at least 11 digits of precision) on top of the Fortran 77 equivalencing mechanism. For these reasons it is important to design a core that is sufficiently trim that new language features can be added in a natural way.

Because Fortran 77 will be one of the modules, the core need not be constrained to contain all archaic features of Fortran. One of the design objectives is to eliminate those features (for example, the arithmetic IF statement) that are no longer necessary due to the addition of better equivalent features or those features (for example, storage association) that actually stand in the way of adding features recognized as contributing to high-quality programming practices.

To provide just one example illustrating how the storage-association concept impedes the addition of useful features, consider the possibility of a conditional array assignment.

```
REAL A(90), B(90), C(90), D(90)
A(*) = 0
B(*) = 0
WHERE (A(*) .LT. 2) DO
      C(*) = B(*) + 1
      END WHERE
```

If no equivalencing is allowed, the assignment may be implemented as

```
  DO 9 I = 1, 90
9 IF (A(I). LT. 2) C(I) = B(I) + 1
```

However, if the program may contain the statement

```
EQUIVALENCE (C(1), B(2))
```

the loop above will set C(I) = I for I = 1 to 90 instead of setting each element to 1. The implementation will be more complex on most machines.

In August 1978, X3J3 approved a proposal to create a first cut at a core language by starting with Fortran 77 and making a number of changes. There are two kinds of changes: features added to the core and features remaining in Fortran 77 but not included in the core.

When reading the list of changes given below, it is important to keep in mind that Fortran 77 will be one of the modules, so any compiler that contains the Fortran 77 module will be able to process programs containing any of the Fortran 77 features. The following two paragraphs are excerpted from the X3J3 proposal to indicate some of the objectives of this approach.

"The general philosophy governing this core design is that the core should be comprehensive, containing virtually all of the generally useful features of Fortran and that it should form a practical, general-purpose programming language. Modules would be used largely for special-purpose language features that entail high-implementation costs or are used primarily in special-purpose application areas. The number of such modules should remain small in order to minimize problems of program portability. Three examples might be (1) a module providing comprehensive array processing facilities, (2) one providing data base management facilities, and (3) one providing features of Fortran 77, and possibly certain other isolated special-purpose features, not contained in the core.

Another goal is to produce a more elegant language by moving redundant features and including features which lend themselves to modern programming practices."

The following changes are not final, but are given to provide a flavor of the final result.

To Be Added	To Be Moved to Fortran 77 Module
Free-form source	Column 6 continuation
Large character set	C for comment
Longer names	
Simple data structures	EQUIVALENCE
Some array processing	COMMON and BLOCK DATA
Global data definition	Passing an array element or substring to a dummy array
Bit data type	Association of ENTRY names
A length (digits) for REAL	DOUBLE PRECISION
Enhanced looping	Arithmetic IF
	Computed GO TO
	Alternate RETURN
	ASSIGN and assigned GO TO
	Statement functions
	ERR = and END = specifiers
	H, X, and D edit descriptors
	Specific names for intrinsics

The net effect of these changes is

- Subroutine linkage facilities are enhanced to improve the interface with applications modules written in Fortran;
- archaic control structures are replaced with modern ones;
- the concept of storage association is removed; and
- fixed-form source is replaced with free-form source.