MIL-STD-1753
9 November 1978

MILITARY STANDARD

FORTRAN, DOD Supplement to American National Standard X3.9-1978

FSC IPSC

DEPARTMENT OF DEFENSE
Washington DC 20301

1. This Military Standard is approved for use by all Departments and Agencies of the Department of Defense.

2. Recommended corrections, additions, or deletions should be addressed to: Hq, U. S. Air Force/ACDX, Washington, DC 20330.

Custodians:

Army - AD

Navy - OM

Air Force - 02

Preparing Activity:

Air Force - 02

1.0 SCOPE

This Supplement contains the syntax and semantics for those facilities necessary to the operating needs of the DOD Departments and Agencies. Facilities contained herein are to be provided as additions to ANSI X3.9-1978 without altering the form or function of the standard facilities. The terminology used herein corresponds to that in ANSI X3.9-1978.

2.0 ADDED FACILITIES

2.1 END DO STATEMENT

The END DO statement must only be used as the terminal statement of a DO-loop and has no other effect. The form of the END DO statement is:

```
END DO
```

If END DO is used as the terminal statement of an ANSI X3.9-1978 DO statement, the END DO statement must be labeled.

2.2 DO WHILE STATEMENT

The DO WHILE statement allows execution of a DO-loop while a logical expression is true. The form of the DO WHILE statement is:

```
DO [ label [,] ] WHILE (logical expression)
```

The logical expression is evaluated and tested at the beginning of the DO-loop.

Each DO WHILE DO-loop must be terminated by a separate END DO statement.

The label is an option to the programmer. If the label is used in the DO WHILE statement, the END DO statement that terminates the DO-loop must be labeled with the same label.

The rules for transfers into the range of the DO-loop are the same as for the current standard DO-loop.

2.3 INCLUDE STATEMENT

The INCLUDE statement provides the capability to copy source code from a file into a compiler source stream. The form of the INCLUDE statement is:

```
INCLUDE  filename
```

2.3 INCLUDE STATEMENT (cont'd)

The INCLUDE statement must be contained on one line.

The filename identifies the source code to be copied into the source stream. The filename can be processor dependent.

An INCLUDE statement will initiate copying at the beginning of the file. The file must not be empty and the first line that is not a comment line must not be a continuation line.

Implementation of the INCLUDE statement must permit the concept of nonrecursive nesting (i.e., the included copy may contain another INCLUDE statement).

2.4 IMPLICIT STATEMENT

The IMPLICIT statement is extended to provide the capability to the programmer to void all default implicit types except for the intrinsic functions. An additional form of the IMPLICIT statement is:

```
IMPLICIT NONE
```

If the IMPLICIT NONE appears in a program unit, no other IMPLICIT statements may appear in the same program unit.

2.5 READ AND WRITE PAST END-OF-FILE

The processor must provide a facility that permits reading and writing to continue past an endfile record on an unlabeled magnetic tape sequential file. Reading past an endfile record is not permitted if the READ statement does not contain an END= or an IOSTAT= specifier. The processor may require execution of a special subroutine or statement before it permits such reading or writing.

2.6 BIT FIELD MANIPULATIONS

Bit manipulation capability is provided through a standard set of external functions. This capability is compatible with similar functions included in ANSI/ISA S61.1-1976. In each of the defined functions, it is assumed that the integer arguments m and n are represented in binary form.

2.6.1 BINARY PATTERN PROCESSING

2.6.1.1 Logical Operations

Logical operations provided are the Boolean functions OR, AND, EOR and NOT. These operations are provided as integer external functions. The implicit type for OR, AND, and EOR is indicated by the use of I as the first letter of the function name. Their arguments, m and n, can be integer constants, integer variables, integer array elements, or integer expressions. After execution of the functions, the arguments remain unchanged. The operations are performed on all corresponding bits of the two operands.

2.6.1.1.1 Inclusive OR

Function reference: IOR (m, n)

The arguments, m and n, are combined according to the following truth table:

m = 0 1 0 1

n = 0 0 1 1

Function value = 0 1 1 1

2.6.1.1.2 Logical AND

Function reference: IAND (m, n)

The arguments, m and n, are combined according to the following truth table:

m = 0 1 0 1

n = 0 0 1 1

Function value = 0 0 0 1

2.6.1.1.3 Logical Complement

Function reference: NOT (m)

The argument m is logically complemented according to the following truth table:

m	0	1
Function value	1	0

2.6.1.1.4 Exclusive OR

Function reference: IEOR (m, n)

The arguments, m and n, are combined according to the following truth table:

m	0	1	0	1
n	0	0	1	1
Reference value	0	1	1	0

2.6.1.2 Shift Operations

The shift operations provided are logical and circular. The shift operations are implemented as integer functions. The arguments may be integer constants, integer variables, integer array elements, or integer expressions. The arguments m and k are as follows:

m specifies the value (binary pattern) to be shifted

k specifies the shift count

k > 0 indicates a left shift

k = 0 indicates no shift

k < 0 indicates a right shift

If the absolute value of the shift count is greater than the number of bits in a numeric storage unit, the result is undefined. The arguments are not changed by the shift operations.

2.6.1.2.1 Logical Shift

Function reference: ISHFT (m, k)

All bits representing the argument m are shifted k places. Bits shifted out from the left end or the right end, as the case may be, are lost. Zeros are shifted in from the opposite end.

2.6.1.2.2 Circular Shift

Function reference: ISHFTC (m, k, ic)

The rightmost ic bits of the argument m are shifted circularly k places; i.e., the bits shifted out of one end are shifted into the opposite end. No bits are lost. The unshifted bits of the result are the same as the unshifted bits of the argument m. The absolute value of the argument k must be less than or equal to ic. The argument ic must be greater than or equal to one and less than or equal to the number of bits in a numeric storage unit.

2.6.2 BIT SUBFIELDS

Bit subfields are referenced by specifying a bit position and a length. Bit positions within a numeric storage unit are numbered from right to left and the rightmost bit position is numbered 0. Bit fields may not extend from one numeric storage unit into another numeric storage unit, and the length of a field must be greater than zero.

2.6.2.1 Bit Extraction

Function reference: IBITS (m, i, len)

where m, i, len are integer expressions

This function extracts a subfield of len bits from m starting with bit position i and extending left for len bits. The result field is right justified and the remaining bits are set to zero. The value of i+len must be less than or equal to the number of bits in a numeric storage unit.

2.6.2.2 Bit Move Subroutine

CALL MVBITS (m, i, len, n, j)

This subroutine moves len bits from positions i through i+len-1 of argument m to positions j through j+len-1 of argument n. The portion of argument n not affected by the movement of bits remains unchanged. All arguments are integer expressions except n must be a variable or array element. Arguments m and n are permitted to be the same numeric storage unit. The values of i+len and j+len must be less than or equal to the number of bits in a numeric storage unit.

2.6.3 BIT PROCESSING

Individual bits of a numeric storage unit can be tested and changed with the following routines for bit processing. The functions have two arguments n and i which are integer expressions.

n specifies the binary pattern

i specifies the bit position (rightmost bit is bit 0)

If i is negative or greater than the number of bits in a numeric storage unit, the result of the function is undefined.

2.6.3.1 Bit Testing

Function reference: BTEST (n, i)

This function is a logical function. The ith bit of argument n is tested. If it is 1, the value of the function is .TRUE.; if it is 0, the value of the function is .FALSE.

2.6.3.2 Set Bit

Function reference: IBSET (n, i)

The result of the IBSET function is equal to the value of n with the ith bit set to a 1.

2.6.3.3 Clear Bit

Function reference: IBCLR (n, i)

The result of the IBCLR function is equal to the value of n with the ith bit set to a 0.

2.6.4 BIT CONSTANTS

The following two forms of bit constants are permitted in DATA statements:

$O'd_i \ldots d_n'$

$Z'h_i \ldots h_n'$

where d_i are octal digits and h_i are hexadecimal digits with A - F representing the decimal equivalent of 10 - 15. These constants are right-justified and may be associated only with integer entities. These constants may appear only in DATA statements.

(NOTE: For reasons of efficiency, it is desirable that the bit manipulation capabilities also be available as in-line code.)

☆ U.S. GOVERNMENT PRINTING OFFICE: 1978 — 603-022/5405

INSTRUCTIONS: In a continuing effort to make our standardization documents better, the DoD provides this form for use in submitting comments and suggestions for improvements. All users of military standardization documents are invited to provide suggestions. This form may be detached, folded along the lines indicated, taped along the loose edge *(DO NOT STAPLE)*, and mailed. In block 5, be as specific as possible about particular problem areas such as wording which required interpretation, was too rigid, restrictive, loose, ambiguous, or was incompatible, and give proposed wording changes which would alleviate the problems. Enter in block 6 any remarks not related to a specific paragraph of the document. If block 7 is filled out, an acknowledgement will be mailed to you within 30 days to let you know that your comments were received and are being considered.

NOTE: This form may not be used to request copies of documents, nor to request waivers, deviations, or clarification of specification requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document*(s)* or to amend contractual requirements.

(Fold along this line)

(Fold along this line)

DEPARTMENT OF THE AIR FORCE

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 73236 WASHINGTON D. C.

POSTAGE WILL BE PAID BY THE DEPARTMENT OF THE AIR FORCE

Hq, U.S Air Force/ACDX
Washington, DC 20330

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

(See Instructions – Reverse Side)

1. DOCUMENT NUMBER	2. DOCUMENT TITLE

3a. NAME OF SUBMITTING ORGANIZATION	4. TYPE OF ORGANIZATION *(Mark one)*
	☐ VENDOR
b. ADDRESS *(Street, City, State, ZIP Code)*	☐ USER
	☐ MANUFACTURER
	☐ OTHER *(Specify)*: _____

5. PROBLEM AREAS

a. Paragraph Number and Wording:

b. Recommended Wording:

c. Reason/Rationale for Recommendation:

6. REMARKS

7a. NAME OF SUBMITTER *(Last, First, MI)* – Optional	b. WORK TELEPHONE NUMBER *(Include Area Code)* – Optional
c. MAILING ADDRESS *(Street, City, State, ZIP Code)* – Optional	8. DATE SUBMISSION *(YYMMDD)*

(TO DETACH THIS FORM, CUT ALONG THIS LINE.)

DD FORM 82 MAR 1426 PREVIOUS EDITION IS OBSOLETE.