BCC 500 FORTRAN REFERENCE MANUAL

Martha Crosby

Manual M-6
Issued September 30, 1975

THE HAWAII 500 PROJECT
Department of Electrical Engineering
University of Hawaii

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2884
Contract No. NAS2-8600

THE HAWAII 500 PROJECT is affiliated with the Department of Electrical Engineering at the University of Hawaii.

Research has been supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by NASA, Ames Research Center under Contract No. NAS2-8600.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency of the United States Government.

BCC 500 FORTRAN REFERENCE MANUAL

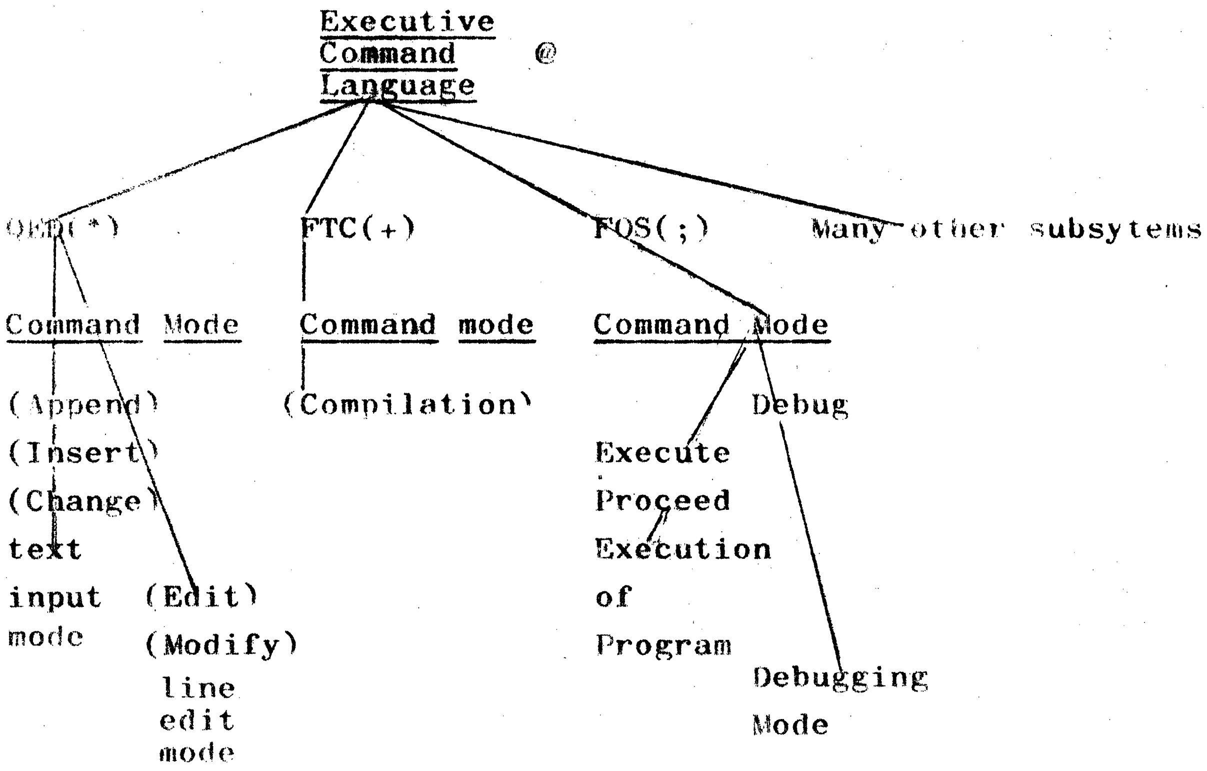
Table of Contents

1. Introduction

The BCC 500 Fortran is a subset of ANSI Fortran IV, formally called ANSI Basic Fortran. The source program is prepared by the text editor, QED. This source file is compiled by the Fortran Compiler, FTC which prepares an object file for the Fortran Operating System, FOS. Debugging facilities in FOS, allow the user to breakpoint statements, execute single statements, and examine and change Fortran variables.

This manual outlines the basic commands for each subsystem, many of which may also be obtained by typing the Help Command in each subsystem. Although the system allows 8K words of storage, much larger programs can be effectively run by using files which are common to all levels of the system, since most large programs can be broken into a sequence of smaller programs communicating through data files. FOS indicates how much storage remains after loading all subprograms. All programs referenced must be loaded whether or not they are called. The following chart show the structure of the Fortran system.

Structure of Fortran System



II. Fortran Compiler

A. Syntax

Expressions and Definitions

Integer Constant- Integer without decimal point using digits 0,1,...9 with optional preceding + or - sign.

Integer Variable-Series of alphanumeric characters (except special characters) the first of which is I,J,K,L,M, or N. The series may be any length for readability but the first six characters comprise the name.

Integer String-Integer variable name defined by nH=n characters where n is from 1 to 3.

Real Constant- Any number written with a decimal point using decimal digits and optional sign. An integer exponent may follow a floating point constant, it may also have a preceeding sign. Examples are:

.0097
9.7E-3
5.0E6

Real Variable-Series of alphanumeric characters (except special characters), the first of which is alphabetic and not I,J,K,L,M, or N. The first six characters define the name.

Real String- Real variable name defined by nH=n characters where n ranges from 1 to 6.

Subscripted Variables- An integer or real variable followed by 1, 2, or 3 integers greater than 0, that are separated by commas. Examples are:

```
Beta(5*J-2)
Max(I,K+2,L)
```

Arithmetic Statements

The form of arithmetic statements is "a"="b" where a may be a subscripted variable and b is an expression.

Expression- Sequence of constants, subscripted or non-subscripted variable(s) and operation symbols which indicate a quantity or a series of calculations.

Operation Symbols- +,-,*,/,** indicating addition,subtraction, multiplication,division, and exponentation.

Rules for constructing expressions

1. The simplest expression consists of a combination of constants and variables. If the quantities are integer tbe expression is in the integer mode, real quantities are in the real mode, string quantities may be used in either mode depending on their names.
2. Exponentation of a quantity does not affect its mode, but an integer may not be given a real exponent.
3. Quantities may be preceded by a +,- or connected by any of the operators to form expressions provided:
   No two operators appear consecutively.
   Quantities connected are of the same mode.
   No operators are assumed to be present.
4. Parenthesis do not affect the mode of the expression but may be used to specify order of precedence which is normally executed left to right in the following order:
   ** exponentation
   */ multiplication and division
   +- addition and subtraction

Control and Specification statements

The following is a list of the Fortran statements, the general form of the statement, the purpose of the statement, and an example.

ACCEPT n, List where n is the statement number of statement, and list is a list of the quantities to be transmitted.

Purpose: read information from teletype as specified by format statement n. Numbers may be right justified to the format by adding a comma after each number as it is typed in.

Example: Accept 9, I,J,K which may be entered as 3,4,5, regardless of the contents of Format statement 9.

ASSIGN i to n where i is a statement number an n is a real or fixed variable which appears in an assigned GO TO statement or as a format number.

Purpose: causes a subsequent GO TO n, $(n_1,n_2,\dots n_m)$ to transfer control to statement i where i is included in the series above, or to transmit data with different formats during execution.

Example:
```
  A(2)=.....
  A(3)=.....
  ASSIGN 3 TO A(1)
3 FORMAT(2F10.5)
  TYPE A(1),A(2),A(3) where A(1) acts as a label
```

CALL name $(a_1,a_2,\dots a_n)$ where name is the name of a subroutine subprogram, and each a_i is an argument.

Purpose: used to call subroutine subprograms; the call transfers control to the subprogram and presents it with the parenthesized arguments.

Example: CALL QDRTIC(P*9.732,Q/4.536,R-S**2.,X1,X2)

CLOSE (i) where i is file number given in OPEN command for reading or writing symbolic (QED) files.

OPEN (i,INPUT,SYMBOLIC,"name:9SYM") where i is file number
OUTPUT and name is file name.

COMMON $a_1,a_2,...a_n$ where each a_i is the name of a variable or non subscripted array name.

Purpose: Causes each a_i to be assigned a location in common storage allocated by position.

Example COMMON X,ANGLE,MATA,MATB

```
        SUBROUTINE SPHERE
        COMMON A,B,C,D
```

CONTINUE

Purpose: used as last statement in range of a DO when the DO would otherwise end with a transfer of control. Also used as a no operation for program readability.

DIMENSION $v_1,v_2,...v_n$ where each v_i is the name of an array subscripted with 1,2 or 3 unsigned integer constants. Each subscript indicates the size of one dimension of the array.

Purpose: provides information necessary to allocate storage for arrays. Storage is assigned columnwise.

Example: DIMENSION A(10), B(5,5,5), J(12,3)

DO n $i=m_1,m_2,m_3$ where n is a statement number, i is a non subscripted integer variable, and m_1,m_2, m_3 are either integer constants or non subscripted variables. If m_3 is not stated it's value is assumed to be 1.

Purpose: command to execute repeatedly the statements which follow up to and including the statement with statement number n. The first time the statements are executed with $i=m_1$. For each succeeding execution i is increased by m_3. Control passes to the statement following n when i exceeds m_2.

Example: DO 25 I=1,15

```
         DO 20 J=1,I
```

END

Purpose: indicates end of source program or subprogram

EQUIVALENCE (a,b,c),(d,e,f)..... where a...f are variables which may have a single subscript.
Purpose: causes all variables specified by each parenthetical expression to be assigned the same location in storage.
Example: EQUIVALENCE (TOP,SIDE(3)), (BOT(14),H)

FORMAT $(s_1,s_2,...s_n)$ where s_i is a format specification.
Purpose:describe type of conversion and format of data to be used in the transmission of an input/output list. Connections may be established during execution as described in the ASSIGN statement. Formats are data interpreted by FOS, therefore it is possible to input an appropriate string of characters into an array from any file such as the teletype at runtime. This feature allows programs to be tested with minimal formats and expanded to any desired level, also part of the output may be deleted with FO.O, IO, or EO.O specified. The format string is referred to by the name of the array which stores it in memory. nA3 should be used for an integer array and nA6 for a real array. In both cases, the number of words n must be greater or equal that required to hold the string but may not exceed the size specified by dimension statements. Termination of output does not produce a carriage return, enabling many different statements to produce one physical line of output. Literal values are delimited by $----$.
Example:

```
  TYPE 1
1 FORMAT($SUMS OF SQUARES = $)
  TYPE 2,SUMSQ
2 FORMAT(I3/)
```

This produces the integer conversion of SUMSQ, one line of output and the explicit carriage return / to line feed.

FUNCTION name $(a_1,a_2,.....a_n)$ where name is the function name subject to mode convention and a_i are arguments.

Purpose: the statement is used at the beginning of a function type subprogram to define its name and arguments.

Example: FUNCTION ROOT(B,A,C)

GO TO n where n is a statement number

Purpose: transfers control to statement n

GO TO $n,(n_1,n_2,...n_m)$ where n is a non-subscripted integer variable appearing in a previously executed ASSIGN statement and n_i is also a statement number that may have been assigned to n by a previously executed ASSIGN statement.

Purpose: transfers control to the statement with statement number equal to that value of n which was last given by an ASSIGN statement.

Example: GO TO K,(100,200,300) where k is 100, 200 or 300.

GO TO $(n_1,n_2,.....n_m)$, i where $n_1,n_2,...n_m$ are statement numbers and i is a non subscripted integer variable.

Purpose: transfers control to the ith value on the list.

Example: GO TO (10,20,30,40),J where J is 1,2,3, or 4.

IF(a) n_1,n_2,n_3 where a is an expression and n_1,n_2, and n_3 statement numbers .

Purpose: causes transfer of control to statemnt n_1,n_2,n_3 depending on whether a is less than, equal to, or greater than zero. Basic fortran does not support logical IF's.

Example: IF((X+Y)-10.) 5,15,25

IF (SENSE LIGHT i) n_1, n_2 where n_1 and n_2 are statement numbers.

Purpose: causes transfer of control to statement n_1 or n_2 if the sense light i is on or off respectively. There are 24 sense lights that may be tested.

Example: IF (SENSE LIGHT 3) 30,40

IF (SENSE SWITCH i) n_1, n_2 where i is the number of a sense switch (1 through 4) and n_1 and n_2 are statement numbers.

Purpose: transfers control to statement n_1 or n_2 if sense switch i is up or down. Sense switches are set in FOS with the i;S for SET and i;R for RESET commands.

Example: IF(SENSE SWITCH 2) 10,20

PAUSE n where n is a number typed if non zero.

Purpose: Stops execution of program temporarily and types "PAUSE n" on the teletype. The user may type ;P to continue the program or debug at that time.

Example: PAUSE 1

READ n, list where n is the statement number of a format and list is the quantities to be transmitted.

Purpose: Allows any QED file to be accessed. Specific symbolic files may be assigned and reassigned during a run. If a file is not assigned default is to the teletype.

Example: READ 1, DATA

RETURN

Purpose: returns control to main program which called it.

SENSE LIGHT i where i, a number between 1 and 24, is turned on. If i is zero, sense lights are turned off.

Purpose: permits sense lights to be turned on or off so that they may later be tested to cause a program to branch.

Example: SENSE LIGHT 5

STOP

Purpose: causes object program to halt and allow for debugging or return to system supervisor.

SUBROUTINE name(a_1,a_2...a_n) where name is the symbolic name of each subprogram, and each a_i is an argument.

Purpose: first statement of SUBROUTINE-type subprogram and defines it to be such,as well as defining its name and arguments.

Example: SUBROUTINE QDRTIC(B,A,C,ROOT1,ROOT2)

TYPE n, list where n is the statement number of a format and list is a list of quantities to be transmitted.

Purpose: causes quantities to be typed on the teletype in accordance with FORMAT n. Many type statements can produce the same physical line of output if a "/" is not encountered in the FORMAT statement.

Example: TYPE 10, A,B,C

Procedures

Fortran procedures consist of Functions and Subroutines. In order to use them they must be defined and called. Functions may be defined in the following four ways:

Arithmetic Statement Functions: These functions are defined by a single arithmetic statement in the source program.

Built in Functions: pre-defined and exist in the program similar to macro's at the assembly level, that is they are incorporated into the object program each time it is refered to by the source program.

Library functions: pre-defined and exist in program library.

Function subprograms: usually user subprograms, that may consist of more than one statement and are common to all subprograms.

Each type of function must observe the following conventions:

May use other functions in its definition.

May have as many variable as desired passed as arguments.

Must have names formed in accordance with rules for naming functions.

Calling functions must follow these rules:

Name indicates the mode of the single value that is result.

Arguments must correspond in number, order, and mode with arguments which appear in the program definition.

Subroutines differ from the more specialized functions in two ways:

They may not be referenced by their appearance in an arithmetic expression but must be used with a CALL.

They may return more than one value which may be passed either with arguments or through COMMON.

B. Library

A number of functions are available from the library file, #2:FLIBE, when called by a loaded program. A compiled subprogram may have the same name as a library function. When two or more subprograms of the same name are read by FOS, the first one is loaded and the rest are ignored.

The library presently contains the following functions:

ALOG computes the natural logarithm of a real argument.

Memory: 138 words

Accuracy: relative error less than $6*10^{-11}$

EXP computes exponential base e of real argument.

Memory: 144 words

Accuracy: relative error less than $6*10^{-11}*2^{\max(0,(\log_2 x+1))}$

SQRT computes square root of real argument.

Memory: 83 words

Accuracy: relative error less than 10^{-11}

ATAN given two arguments, y and x, the routine computes the arctangent of y/x giving the result in radians in the proper quadrant. If one argument is given x is assumed to be 1.

Memory: 256 words

Accuracy: relative error less than 10^{-11}

ABS,IABS real or integer absolute value, argument may be of either mode.

Memory: 13 words

FLOAT converts integer argument to real

Memory: 4 words

IFIX,INT,AINT integer or floating value of real argument truncated to integer. Positive and negative arguments are both truncated toward zero.

Memory: 8 words

ISIGN,SIGN integer or real result of the algebraic sign of the second argument, assigned to the value of the first argument.

Memory: 20-21 words

AMOD requires two real arguments, returns the remainder when the first is divided by the second. That is

AMOD(A,B)=A-FLOAT(FIX(A/B))*B

Memory: 13 words

MOD requires two integer arguments. Returns the remainder when the first is divided by the second. For integers

MOD(I,J)=I-(I/J)*J

Memory: 9 words

MAX,AMAX finds integer or real maximum of any number of arguments of either mode.

MIN,AMIN finds integer or real minimum of any number of arguments of either mode.

Memory: 60 words, includes all four entries.

DIM requires two real arguments, returns the difference if the first one is greater than the second, otherwise returns zero.
DIM(A,B)=AMAX(A-B, 0.0)
DIM(A,0.0)=AMAX(A,0.0) and -DIM(0.0,A)=AMIN(0.0,A)
The DIM function is much shorter if the result is needed.
Memory: 10 words

IDIM requires two integer arguments. Returns the difference if the first is greater than the second, otherwise returns zero.
IDIM(I,J)=MAX(I-J,0)
Memory: 10 words

LOCF returns the absolute address of an argument of either mode.
Memory: 4 words

IF given two real arguments, P and Q, this function returns zero if they are equal within the four low order mantissa bits, otherwise it returns an integer with the sign of P-Q. Given one real argument P, the function returns zero if its magnitude is less than 10^{-10}, otherwise it returns an integer with the sign of P. This function is useful in conjunction with the if statement to provide a means of testing equality of decimal numbers in binary.
Memory: 25 words.

EXIT same effect as STOP statement, except that *EXIT* is typed. FOS returns to the command mode.
Memory: 10 words.

POWER,FORM,TIME,BRS,EOF,ISIZE, and IPOSIT also exist in the library file and are for the most part built in functions.

C. Symbol Table Size

Symbol table storage is dynamically allocated by the compiler. None of the tables have fixed length; each may be lengthened, shortened, or relocated as items are added or removed. No table can be exceeded until all memory is used. Included in the symbol table storage is the working storage for statement translation. This area is expanded during the analysis of each statement and contracted as the program is written out. Since it's size fluctuates rapidly in proportion to statement complexity, it is difficult to predict the available symbol table storage, but may be approximated at 150 words. Table storage is bound in the following way:

N+2S+6A+2F+I+2G+4L+2C+3D+3D+M+W less than TABLESIZE

where:

A= number of array variables
C= common identifiers
D= do loops
E= equivalence identifiers
F= real constants
G= global subprograms
I= integer constants
L= local subprograms (Arithmetic Statement functions)
M= format statements
S= number of scalar variables
W= working storage

D. Commands

To invoke the Fortran Compiler give the executive command @FTC which responds with it's name, version and +

+H.lists all the commands available in the subsystem.

The commands which must be confirmed by a "." are:

+Input from (FILE-NAME). Source file should be 9SYM
+Output from (FILE-NAME). Compiled object program 9BIN
+List to (FILE-NAME). 9SYM If listing is wanted on terminal "*T" should be specified as file name.
+Debug. must be invoked prior to compile if runtime debugger is going to be used.
+Map. gives map of program variables
+NoMap. listing normally produces map of program variable storage. This is omitted by invoking NoMap after list.
+NoList.
+NoDebug.
+Compile.
+Finished.
+" Ignore this line.

Syntax check with nocode generation is provided by not invoking the output command.

New files are created by enclosing the file name in double quotes.

Typing control-K at any time returns to the "+" command processor.

III. Fortran Operating System

A. Loader

FOS includes such operations as floating point arithmetic, format scanning, and program debugging. Fortran programs compiled by FTC are loaded and executed with FOS by giving the following command:

@FOS carriage return The system responds with

LOAD MAIN PROGRAM
FROM FILE (FILE-NAME).

If subprograms are called they must be read following the routine which calls it, if this order is violated, the names of the missing routines will be typed and the file should be read again. If library routines are not included in the user files, they should be loaded when the system responds

LOAD SUBPROGRAMS
FROM FILE   for any subprograms type FILE-NAME.
for the library type #2:FLIBE.

When all the programs and subprograms the system will respond with

LOADING COMPLETE, the time, and the unused storage.

Transfers to the executive are permitted during the loading process provided FOS isn't waiting for the user to open another file. The following situations may arise while loading:

FILE NOT BINARY	Files not 9BIN or not
ILLEGAL FILE	generated by FTC
PROGRAM TOO BIG	Exceeds 8K currently available for programs and subprograms

At this point the program may be executed, if the program was compiled without the debugging option, only the following commands may be used:

+;G	Go to the first statement of the main program
+;P	Proceed after pause or error
+;D	Disregard previous error or pause hereafter
+;N	Reinstate all disregarded items
+;F	Exit Fortran Operating System
+(n);S	Set senseswitch n
+(n);R	Reset sense switch n
+"	Ignore this line
Control /K	Return to "+" command processor
+;H	Prints all commands of subsystem

B. Runtime Debugging

If the program was compiled with the DEBUG option, the debugger commands may be used. These include:

+(address);G	Go to addressed statement
+;P	Proceed afer error pause or breakpoint
+(address)!(n)	Set breakpoint at addressed line $0<n<4$
+!(n)	Clear breakpoint n
+!0	Clear all breakpoints
+(address);C	Replace address with continue statement
+.=	Print address of current line
+(address)=	Print closest relative adress
+(name)/	Print variable name in intrinsic mode
+(name)[	Print variable name in octal
+(name)"	Print variable name in ASCII
+(name)←/	Intrinsic mode input
+(name)←[	Octal mode input
+(name)←"	ASCII mode input

Addresses for the debugger may have one or two parts. The two part address specifies a program unit followed by a relative label address. Once a two part address has been given the debugger remembers the program unit. Thereafter a one part address specifying only the relative label will operate within the most recently specified program unit. Initially the main program is assumed specified. Examples:

+USER,100 Subprogram USER, label 100

+USER,100-5 Subprogram USER, 5 statements prior to label 100

+$,40 Main program, label 40

+100+10!1 Main program, breakpoint 1 at 10 statements beyond label 100

C. Runtime Diagnostics

AGTO An assigned GOTO statement has been encountered but no variable has been assigned.

ARGM An argument of the wrong mode has been transmitted to a subprogram. The incorrect mode is used.

ARGN The wrong number of arguments has been transmitted to a subprogram. If too many were transmitted, the extra ones are ignored. If too few were transmitted the extra positions are filled with garbage.

CGTO The value of a computed GOTO lies outside the range specified. Control transfers to the first statement of the given list.

EF1A FOS is unable to output one or more variables as the FORMAT statement lacks a needed E,F,I, or A. The variable is not transmitted.

EXP The argument of an expontial function is greater than 176 octal. The answer is set to the maximum real value.

FCHR FOS has detected an illegal format character. The character is ignored and a scan for the next specification is begun. Character has same effect as a comma.

FORM,FORP The I/O statement variable references something other than a format statement.

FORI The I/O statement which references a FORMAT has never been assigned.

ICHR FOS has received an illegal input character. The character is ignored and a scan is begun for the next input field.

IFSL The value of an IF SENSE LIGHT statement is not between 1 and 24. The sense light is assumed off.

IFSS The value of an IF SENSE SWITCH is other than between 1 and 4. The sense switch is assumed off.

INUM An input number to FOS is outside of range. The value is set to zero.

LABL Program specifies a transfer to an undefined label The program cannot be continued, but the debugger may be used.

LOG The argument of a logarithm function is negative or zero, the result of the function is set to zero.

N**F The program has tried to raise a negative number to a non-integral real power. The form /N/**F is computed instead.

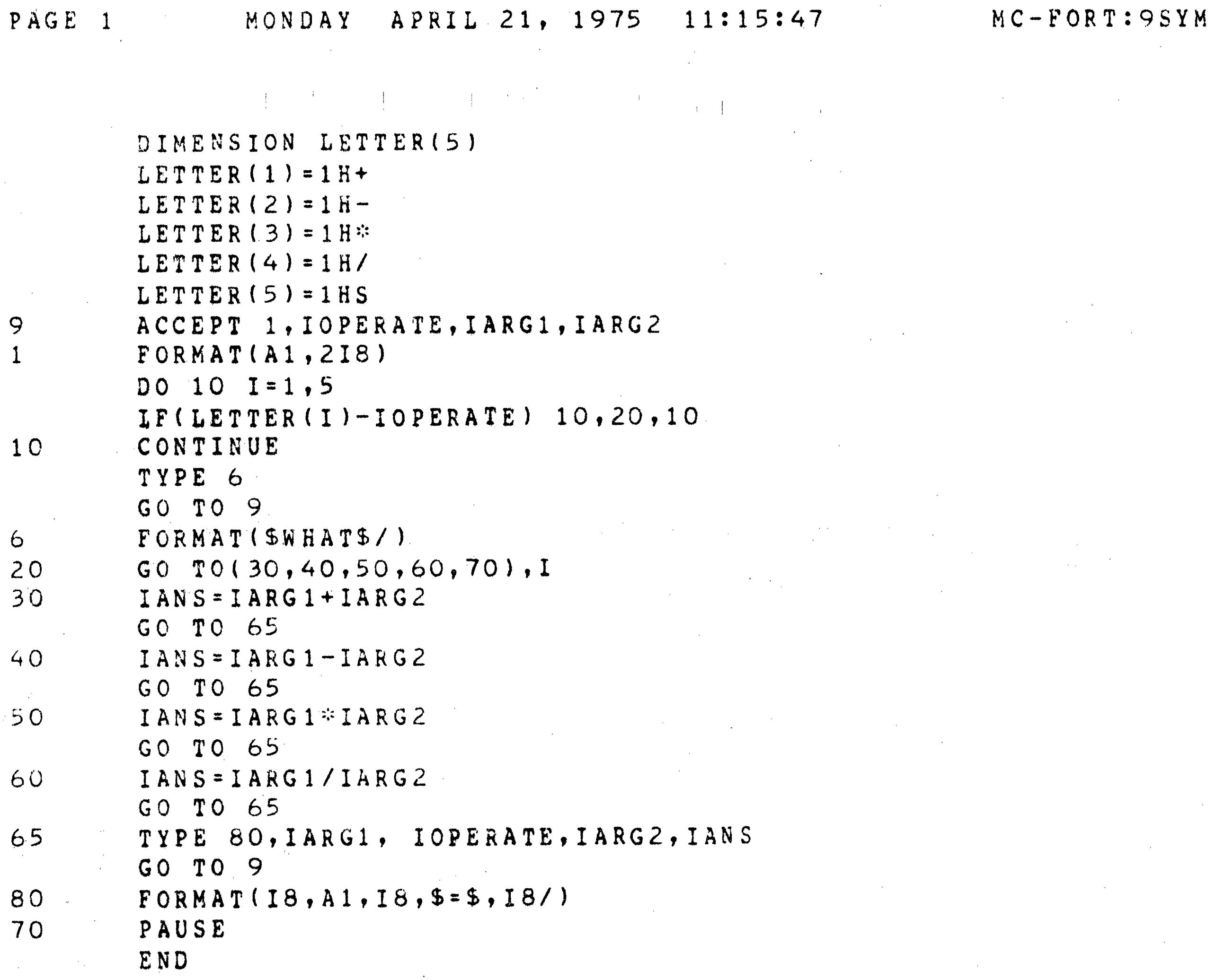
OEXP Output exponent exceeds range. The number is transmitted with 0 exponent.

SIZE The size of storage has been exceeded, continuing program will destroy common storage required by subroutine calls.

SNLT The value of the SENSE LIGHT is not in the range of 1 to 24. The statement is ignored.

SQRT A negative argument was passed to the square root subroutine. The absolute value is used.

O**N The program tried to raise 0 to a non positive power. If it was to the 0 power a 1 or 1. is returned. If it was to negative power, the maximum possible real or integer value is returned.



```
 PAGE 1        MONDAY  APRIL 21, 1975  11:15:47          MC-FORT:9SYM


          DIMENSION LETTER(5)
          LETTER(1)=1H+
          LETTER(2)=1H-
          LETTER(3)=1H*
          LETTER(4)=1H/
          LETTER(5)=1HS
 9        ACCEPT 1,IOPERATE,IARG1,IARG2
 1        FORMAT(A1,2I8)
          DO 10 I=1,5
          IF(LETTER(I)-IOPERATE) 10,20,10
 10       CONTINUE
          TYPE 6
          GO TO 9
 6        FORMAT($WHAT$/)
 20       GO TO(30,40,50,60,70),I
 30       IANS=IARG1+IARG2
          GO TO 65
 40       IANS=IARG1-IARG2
          GO TO 65
 50       IANS=IARG1*IARG2
          GO TO 65
 60       IANS=IARG1/IARG2
          GO TO 65
 65       TYPE 80,IARG1, IOPERATE,IARG2,IANS
          GO TO 9
 80       FORMAT(I8,A1,I8,$=$,I8/)
 70       PAUSE
          END
```

This example shows assignment and comparison of non-numeric data, the arithmetic if statement, and input-output to teletype using TYPE and ACCEPT commands.

```
@ FTC

VERSION 12-03-7Ø ("H." FOR HELP) TODAY IS Ø4/21/75 125Ø:39
+INPUT FROM MC-FORT.
+OUTPUT TO "OBJECT".
+COMPILE.


COMPILING MAIN PROGRAM


COMPILE TIME Ø:Ø:5

+FINISHED. TOTAL COMPUTE TIME Ø:Ø:7


@ FOS

VERSION 12-Ø3-7Ø (";H" FOR HELP) TODAY IS Ø4/21/75 1252:Ø5
LOAD MAIN PROGRAM
FROM FILE OBJECT.
LOADING TIME Ø:Ø:1
8Ø63 WORDS OF STORAGE UNUSED
+;G


+1234,5678,    1234+    5678=    6912
-9876,8765,    9876-    8765=    1111
*2468,1111,    2468*    1111= 2741948
/9999,9999,    9999/    9999=       1

S

PAUSE
+;F
TOTAL COMPUTE TIME Ø:Ø:3

@
```

```
      DIMENSION IGRADE(50),KEY(50),SSN(2,50)
1     FORMAT ( 2A6,I3/)
10    I=1
      SDEV=0.
      XMEAN=0.
      TYPE 2
2     FORMAT($ TYPE ID AND GRADES$/)
20    ACCEPT 1, (SSN(K,I),K=1,2),IGRADE(I)
      IF(IGRADE(I)-100)  30,30,40
30    GRADE=IGRADE(I)
      XMEAN=XMEAN+GRADE
      SDEV=SDEV+GRADE**2
      KEY(I)=I
      I=I+1
      GO TO 20
40    I=I-1
      CALL SORTI(IGRADE,KEY,I)
      DO 45 M=1,I
      L=KEY(M)
45    TYPE1,(SSN(K,L),K=1,2),IGRADE(M)
      LHIGH=IGRADE(I)
      LOW=IGRADE(1)
      FN=I
      M=(FN+1.)/2.
      MEDIAN=IGRADE(M)
      SDEV=SQRT((SDEV-XMEAN**2/FN)/(FN-1.))
      XMEAN=XMEAN/FN
      TYPE 3,I,XMEAN
      TYPE 6,SDEV,MEDIAN
      TYPE 7,LHIGH,LOW
3     FORMAT($ FOR$,I3,$ DATA POINTS, THE MEAN IS $,I3/)
6     FORMAT($ THE STANDARD DEVIATION IS$,F5.2,$ THE MEDIAN IS$,I3/)
7     FORMAT($ RANGE IS FROM$,I3,$ TO$,I3/)
50    PAUSE 1
      GO TO 10
      END
      SUBROUTINE SORTI (L,KEY,NO)
      DIMENSION L(1),KEY(1)
      MO=NO
10    IF(MO-16) 80,20,20
20    MO=2*(MO/8)+1
30    KO=NO-MO
      JO=1
40    I=JO
      IPMO=I+MO
50    IF(L(IPMO)-L(I)) 60,60,70
60    LEMP=L(I)
      L(I)=L(IPMO)
      L(IPMO)=LEMP
      KEMP=KEY(I)
      KEY(I)=KEY(IPMO)
      KEY(IPMO)=KEMP
      IPMO=I
      I=I-MO
```

```
        IF(I-1) 70,50,50
70      JO=JO+1
        IF(JO-KO) 40,40,10
80      IF(MO-1) 100,100,90
90      MO=2*(MO/4)+1
        GO TO 30
100     RETURN
        END
```

This example shows calling a subroutine with variable dimensions, using non-numeric data in an array, and conversion from real to fixed point output in the format statement.

The data could be read from a symbolic file by inserting the following changes; file "DATA" is written in QED.

```
        .....
        OPEN(3,INPUT,SYMBOLIC,"DATA:9SYM")
        .....
        .....
20      READ(3,1) (SSN(K,I),K=1,2),IGRADE(I)
        .....
        .....
        CLOSE(3)
        .....
```

```
@ FTC

VERSION 12-Ø3-7Ø ("H." FOR HELP) TODAY IS Ø4/22/75 1329:56
+INPUT FROM TEST.
+OUTPUT TO "TOBJ".
+DEBUG.
+COMPILE.


COMPILING MAIN PROGRAM


COMPILING SUBROUTINE SORTI


COMPILE TIME Ø:Ø:9

+FINISHED. TOTAL COMPUTE TIME Ø:Ø:1Ø


@ FOS

VERSION 12-Ø3-7Ø (";H" FOR HELP) TODAY IS Ø4/22/75 1331:1Ø
LOAD MAIN PROGRAM
FROM FILE TOBJ.
MISSING
 SQRT
 24ØSYS
LOAD SUBPROGRAMS
FROM FILE #2:FLIBE.
LOADING TIME Ø:Ø:4
692Ø WORDS OF STORAGE UNUSED
+;G
 TYPE ID AND GRADES
 523-48-8131 9Ø
 312-44-1Ø3Ø 7Ø
 Ø26-36-5475 82
 576-46-4387 78
 575-38-2978 92
            999
 312-44-1Ø3Ø 7Ø
 576-46-4387 78
 Ø26-36-5475 82
 523-48-8131 9Ø
 575-38-2978 92
 FOR  5 DATA POINTS, THE MEAN IS  82
 THE STANDARD DEVIATION IS 8.99 THE MEDIAN IS 82
 RANGE IS FROM 92 TO 7Ø

PAUSE 1
+KEY(1)/ 2
+KEY(2)/ 4
+KEY(3)/ 3
+KEY(4)/ 1
+KEY(5)/ 5
+;F
TOTAL COMPUTE TIME Ø:Ø:8

@
```

P. REFERENCES

1. FORTRAN II Reference Manual, Document 30.50.50
Feb. 8, 1966, C. Stephen Carr, University of California, Berkeley.

2. Batch FORTRAN Reference Series, Tymshare, Revision 4, October 1968.