8-5-S

FLOATING-POINT SYSTEM PROGRAMMING MANUAL

PDP-8

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

PDP-8 FLOATING-POINT SYSTEM PROGRAMMING MANUAL

September 1965

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Reprinted January 1967
Reprinted June 1967
Reprinted November 1967

PREFACE

The PDP-8 comes to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources, past programming effort on the PDP-5 computer, and present and continuing programming effort on the PDP-8. Thus the PDP-8 programming system takes advantage of the many man-years of program development and field testing by PDP-5 users.

Although in many cases PDP-8 programs originated as PDP-5 programs, all utility and functional program documentation is issued in a new, recursive format introduced with the PDP-8.

Programs written by users of either the PDP-5 or the PDP-8 and submitted to the users' library (DECUS - Digital Equipment Corporation Users' Society) are immediately available to PDP-8 users.

Consequently, users of either computer can take immediate advantage of the continuing program developments for the other.

CONTENTS

CONTENTS (continued)

CHAPTER 1

INTRODUCTION

The PDP-8 Floating-Point System enables the programmer to concentrate on the logic of his computation rather than on decimal points. The system maintains a constant number of significant digits throughout the computation, thereby enhancing the accuracy of the result.

Floating-point notation is particularly useful for computations involving numerous multiplications and divisions where magnitudes are likely to vary widely and where only crude predictions can be made as to the amount of variation involved. The main advantage of the PDP-8 Floating-Point System is derived from the ability to store very large or very small numbers by storing only the significant digits together with the exponent for that number. In an integral or fractional fixed-point machine, the programmer must either include all of the 0's between the significant digits and the decimal point (and in so doing loose considerable accuracy), or account for the point in each step of his programming.

The PDP-8 Floating-Point System is constructed as a self-contained package which includes its own input, arithmetic, and output routines. It allows the programmer to use floating-point arithmetic without having to construct his own arithmetic subroutines. For example, the polynomial:

$$Y = AX^3 + BX^2 + CX + D$$

may be programmed as follows (the variables X and Y, and the parameters A, B, C, and D are stored in registers of the same name):

```
......
JMS   I  7      /ENTER FLOATING PACKAGE
FGET  A         /LOAD PSEUDO AC
FMPY  X         /MULTIPLY
FADD  B         /ADD
FMPY  X         /MULTIPLY
FADD  C         /ADD
FMPY  X         /MULTIPLY
FADD  D         /ADD
FPUT  Y         /STORE
FEXT            /EXIT FLOATING PACKAGE
......
```

CHAPTER 2

FLOATING-POINT REPRESENTATION

A floating-point number consists of a mantissa and an exponent. For example, the decimal number 12 may be represented as:

$1200.0 \cdot 10^{-2}$

$120.00 \cdot 10^{-1}$

$12.000 \cdot 10^{0}$

$1.2000 \cdot 10^{1}$

$.12000 \cdot 10^{2}$

....

etc.

where the exponents are the numbers −2, −1, 0, etc.

Since the PDP-8 is a binary machine, floating-point numbers are stored as floating-point binary internally. For example, the binary number 110 (6 decimal) may be represented as:

....	
$11000.0 \cdot 2^{-2}$	$(24 \cdot 1/4 = 6)$
$1100.00 \cdot 2^{-1}$	$(12 \cdot 1/2 = 6)$
$110.000 \cdot 2^{0}$	$(6 \cdot 1 = 6)$
$11.0000 \cdot 2^{1}$	$(3 \cdot 2 = 6)$
$1.10000 \cdot 2^{2}$	$(3/2 \cdot 4 = 6)$
$.110000 \cdot 2^{3}$	$(3/4 \cdot 8 = 6)$
....	
etc.	

Notice that the binary exponent is always a signed integer. The PDP-8 Floating-Point System uses the convention:

$$1/2 \leq |\text{MANTISSA}| < 1.$$

When this is true, the word is said to be normalized.

The value of the number is then:

$$\text{MANTISSA} \cdot 2^{\text{EXPONENT}}$$

where the MANTISSA is a signed quantity. The result of this convention is that more significant bits are retained. For example, the number .10 (decimal) is equal to:

|0.00 011 001 100 | 110 011 001 100 | 110 011 001 100 110 ....

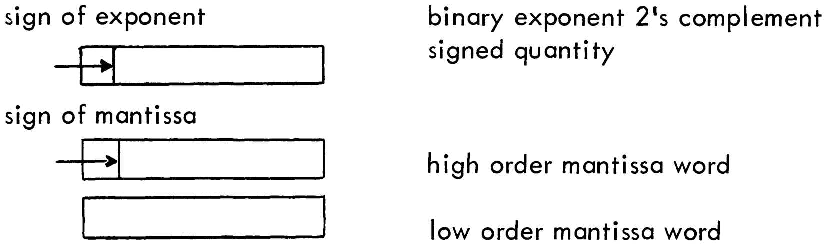
in binary.

When this word is stored in two 12-bit words (24 bits), the leading 0's are nonsignificant and only 20 bits of significance are maintained. If the number is rewritten as:

2^{-3}.| 0.11 001 100 110 | 011 001 100 110 | 011 001 100 110 ....

and the mantissa stored in two 12-bit words, 23 bits of significance are maintained.

The exponent is stored in a third word making a total of three words for storage.



The number .1 (decimal) would be stored as:

111 111 111 101 | 011001100110 | 011001100110

or written in octal:

7775 3146 3146

The address of this number would be considered 2146 if it were stored as follows:

2146/7775
2147/3146
2150/3146

The number −.1 would be:

7775
4631 (double precision 2's complement of mantissa)
4632

Since the mantissa is greater (in magnitude) than 1/2 and less than 1, its binary point is considered to be between bit 0 and bit 1 of the high order mantissa word.

Further examples:

12 (decimal)

$1100. = (2^4)\ (0.110000000000000000000000)$

would be stored as (written in octal):

0004

3000

0000

Octal	Decimal		Floating Binary	
-1	-1	0001	6000	0000
-10	-8	0004	6000	0000
.4	.5	0000	2000	0000
3	3	0002	3000	0000
14	12	0004	3000	0000
.2	.25	7777	2000	0000

Convert the decimal number 1.6 to PDP-8 floating-binary form for storage.

An easy method to convert a decimal fraction to an octal fraction is illustrated below:

```
   .6
  x 8
4) .8
  x 8
6) .4
  x 8
3) .2
  x 8
1) .6
    8
4) .8
    8
6) .4
    8
3) .2
    8
1) .6
```

$1.6_{10} = 1.46314631_8$

$= 1.100110011001100110011001$

$= (2^1)\ (0.11001100110011001100110)$

= 0001

3146

3146

ARITHMETIC

Since floating-point numbers are stored in a three-register format, the floating-point system uses a "psuedo" floating accumulator (FAC) which consists of three registers in the floating-point package: 44, 45, and 46. Register 44 contains the exponent; 45 and 46 contain the high and low order parts of the mantissa, respectively.

BASIC FLOATING-POINT COMMANDS

The basic floating-point commands include the following:

load floating accumulator
store floating accumulator
add to floating accumulator
subtract from floating accumulator
multiply by floating accumulator
divide into floating accumulator
normalize floating accumulator

All arithmetic operations are called through an interpreter. The command codes have a format that is almost identical to the format of the PDP-8 memory reference instruction, namely:

OP	I	Z	ADDRESS

where the op code is from 000 to 111 or 0_8 - 7_8.

INTERPRETER

The interpreter contains, at all times, the address of the memory location containing the next pseudo instruction to be executed. This is initially stored when the program enters the interpreter using a JMS instruction. When the interpreter encounters an instruction with an op code of 0 and with bits 8-11 of the pseudo instruction equal to 0, it exits to the next memory location.

FLOATING-POINT INSTRUCTIONS

The floating-point instructions are:

Op Code	Mnemonic	Effect
1	FADD	Floating Addition Add the contents of the effective address to the floating accumulator.

2 FSUB Floating Subtract

Subtract the contents of the effect address from the floating accumulator.

3 FMPY Floating Multiply

Multiply the floating accumulator by the contents of the effective address.

4 FDIV Floating Divide

Divide the floating accumulator by the contents of the effecitve address.

5 FGET Floating Get

Load the floating accumulator with the contents of the effective address.

6 FPUT Floating Put

Store the contents of the floating accumulator at the locations specified by the effective address. The contents of the floating accumulator are unchanged.

7 FNOR Floating Normalize

Normalize the contents of the floating accumulator.

0 is decoded as follows:

Bits 8-11 = 0000 Floating Exit

Return control to following instruction

= 0001 Floating Square

Square the contents of the floating accumulator

= 0010 Floating Square Root

Take the root of the absolute value of the floating accumulator

= 0011-1111 Expandable commands

This may be summarized:

$$\left.\begin{array}{l}\left.\begin{array}{lll}\text{FADD} & \text{Y; 1000;} & \text{C(FAC)} + \text{C(Y)} \rightarrow \text{C(FAC)}\\ \text{FSUB} & \text{Y; 2000;} & \text{C(FAC)} - \text{C(Y)} \rightarrow \text{C(FAC)}\\ \text{FMPY} & \text{Y; 3000;} & \text{C(FAC)} \times \text{C(Y)} \rightarrow \text{C(FAC)}\\ \text{FDIV} & \text{Y; 4000;} & \text{C(FAC)} \div \text{C(Y)} \rightarrow \text{C(FAC)}\end{array}\right\}\begin{array}{l}\text{Result is}\\ \text{normalized}\end{array}\\ \begin{array}{lll}\text{FGET} & \text{Y; 5000;} & \text{C(Y)} \rightarrow \text{C(FAC)}\end{array}\end{array}\right\}\begin{array}{l}\text{C(Y)}\\ \text{unchanged}\end{array}$$

FPUT Y; 6000; C(FAC) → C(Y)

FNOR ; 7000; C(FAC) normalized → C(FAC)

FEXT ; 0000; exit from interpreter to instruction following this command

SQUARE ; 0001; $C(FAC)^2 \rightarrow C(FAC)$

SQROOT; 0002; $C(FAC)^{1/2} \rightarrow C(FAC)$

The assembler recognizes all of these mnemonics except for SQUARE and SQROOT which may be defined as:

```
SQUARE=0001
SQROOT=0002
```

The floating-point interpreter is entered with an effective JMS 5600.

For example:

```
SQROOT=0002
*7
 5600
*200
         JMS I 7      200/      4407
         FGET A       201/      5206
         SQROOT       202/      0002
         FPUT I B     203/      6611
         FEXT         204/      0000
         HLT          205/      7402
A,       0003         206/      0003
         2000         207/      2000
         0000         210/      0000
B,       300          211/      0300
$
```

When this program is started at 0200, it will halt at location 205. The state of the machine will be:

```
44/      0002
45/      2000      floating accumulator contains
46/      0000           2.0, i.e., (4.0)^1/2

206/     0003
207/     2000      register A contains 4.0
210/     0000

300/     0002
301/     2000      answer stored here
302/     0000
```

CHAPTER 3

FLOATING-POINT INPUT/OUTPUT

FLOATING-POINT INPUT

The basic floating-point package contains an input routine to read characters from the 33 ASR keyboard. Numbers are read in in decimal and converted to the internal floating-point binary format. Input format is floating decimal. The number 726.7 may be read in in any of the following forms:

```
726.7
.7267E3
.7267E+03
+7267E-1
etc.
```

Input is terminated when a character is typed that is not a part of this format. The conversion of "12.0." would be terminated on the second "." and the binary number:

```
000000000100        (octal 0004)
011000000000        (octal 3000)
000000000000        (octal 0000)
```

would be in the floating-point accumulator upon completion of the conversion.

Flags

There are several flags associated with the input routine that are useful to the programmer.

0056 This register is a switch that has the following meaning:

If C(56) = 0, do not type a line-feed after a carriage-return was read.
If C(56) = 7777, type a line-feed when a carriage-return is inputted.
This switch is initially set to 7777.

0057 This register contains the character that terminated the input conversion.

0060 This register contains 0000 if no input conversion was made; i.e., a space or other terminator was initially typed.

The input routine is entered with an effective JMS 7400. It returns control to the instruction following the calling JMS upon receipt of a terminator. The floating accumulator contains the input number in normalized floating-binary. Register 0057 contains the terminating character in ASCII, and C(0060) indicates whether or not there was a valid input.

For example:

```
*5
 7400
*7
 5600
*200
 JMS I 5         /INPUT ROUTINE
 JMS I 7         /CALL FLOATING POINT
 FPUT A
 FEXT
 JMS I 5
 JMS I 7
 FPUT B
 FEXT
 HLT
A,  0
    0
B,  0
    0
    0
$
```

When this program is started at 0200 and the following is typed:

```
X2.0Y
```

The program will halt at location 0210, and A and B will contain

```
A,  0
    0
    0          and C(57) = 0331, the second TERMINATOR
B,  0002
    2000
    0000
```

The first input was considered a 0 because a terminator "X" was inputted.

This program could be written to ignore the non-numeric information as follows:

```
*5
    7400
*7
    5600
*200
    JMS I 5          /CALL INPUT ROUTINE
    TAD 60           /ANY VALID INPUT?
    SNA  CLA
    JMP .-3          /NO - IGNORE
    JMS I 7          /YES
    FPUT A           /STORE IT
    FEXT
    JMS I 5          /GET NEXT
    TAD 60
    SNA CLA          /VALID?
    JMP .-3          /NO - IGNORE
    JMS I 7          /YES
    FPUT B           /STORE IT
    FEXT
    HLT      /HALT
A,  0
    0
    0
B,  0
    0
    0
$
```

Register 57 may be used for integrating control characters into the input.

Rubout

There is one special input character, the rubout. If it is struck before an input delimiter, the input routine is restarted and previous numbers deleted.

For example, assume the input routine has been entered.

```
276 Rubout 1T
```

The input routine would exit with 1 (decimal) in the floating accumulator or:

```
44/       0001
          2000
          0000
```

FLOATING-POINT OUTPUT

The basic floating-point package contains an output routine. Upon entry the contents of the floating accumulator are converted to floating-point decimal and typed out on the ASR 33 in the following format:

```
        ±0.XXXXXXXE±XX
```

For example, if the floating accumulator contained:

```
        44/     0002
                2000
                0000
```

and the output routine were entered,

```
        +0.2000000E+01
```

would be typed.

Entry

The output routine is entered by an effective JMS 7200. After outputting the contents of the floating accumulator, it returns control to the instruction following the calling JMS instruction. The contents of the floating accumulator are lost. The floating-output routine has a flag or program switch in location 0055 on page 0.

If C(0055) does not equal 0, type a carriage-return/line-feed following each output. This location initially contains 7777.

Program Example

```
/SOLVE THE QUADRATIC EQUATION
/AX2+BX+C=0 FOR VALUES
/OF A, B, C.        TYPE OUT ROOTS
SQROOT=0002
SQUARE=0001
*5      7400        /POINTER TO INPUT ROUTINE
        7200        /POINTER TO OUTPUT ROUTINE
        5600        /POINTER TO INTERPRETER
*200
            KCC
            TLS
BEGIN,      JMS I 5     /INPUT A
            JMS I 7     /ENTER INTERPRETER
        FMPY TWO        /2.0
        FPUT A          /STORE 2.0·A
        FEXT
        JMS I 5         /INPUT B
        JMS I 7         /ENTER INTERPRETER
        FPUT B          /STORE B
        FEXT
        JMS I 5         /INPUT C
        JMS I 7         /ENTER INTERPRETER
        FMPY TWO        /MULTIPLY BY 2.
        FMPY A          /MULTIPLY BY 2A
        FPUT TEMP       /4AC
        FGET B
        SQUARE          /B SQUARED
        FSUB TEMP       /B SQUARED -4AC
        SQROOT          /TAKE SQUARE ROOT
        FPUT TEMP       /STORE IT
        FGET B
        FMPY MIN1       /-1·B = — B
        FPUT B          /STORE
        FSUB TEMP       /-B—SQUARE ROOT
        FDIV A
        FEXT            /ANSWER IN FAC
        JMS I 6         /OUTPUT IT
        JMS I 7         /ENTER INTERPRETER
        FGET B          /-B
        FADD TEMP       /-B + SQUARE ROOT
        FDIV A
        FEXT            /ANSWER IN FAC
        JMS I 6         /OUTPUT IT
        JMP BEGIN       /CONTINUE
TWO,    0002            /CONSTANT 2.0
        2000
        0000
MIN1,   0001            /CONSTANT -1.0
        6000
        0000
TEMP,   0
        0
        0
A,      0
        0
        0
B,      0
        0
        0
C,      0
        0
        0
$
```

Both the input-conversion routine and the output-conversion routine use a typeout subroutine that contains the instructions:

```
. . .
TSF
JMP .-1
TLS
CLA
. . .
```

This means that the teleprinter flag must be set when these routines are entered or it must be in the process of typing. A TLS instruction can be put into the initialization section of a program.

THE BASIC PACKAGE

Subroutines

When the floating-point interpreter encounters a 0 op code, it further decodes bits 8-11. If these bits equal 0, the interpreter exits. If the bits are nonzero, they are used in a table look-up to specify the address of a subroutine. The called subroutine may use the floating-point interpreter, the input, or the output, but it may not use bits 8-11 to call another subroutine. The look-up table is on page 16 of the interpreter listing. For example, if the user wished to define a routine that negated the contents of the floating accumulator, the following steps would be taken:

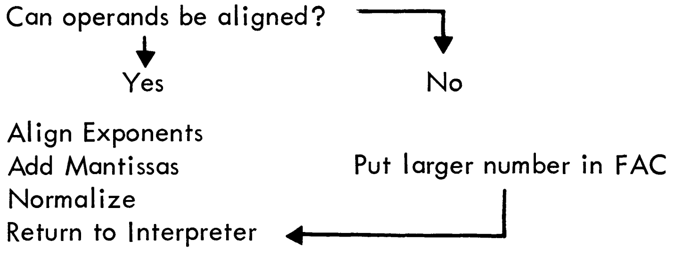
On page 6 of the interpreter listing, there is a routine to negate the floating accumulator. It is a subroutine and has, as entry point, the address 6000. If the negate subroutine were to be called by 0010 in the interpreter mode, the word 6000 would be placed in address 6554 of the calling table. Input and output could be called in the interpreter mode if 7400 were placed in 6555 (0011) and 7200 were placed in 6556 (0012).

```
NEGATE = 0010
INPUT  = 0011
OUTPUT = 0012
SQUARE = 0001
*7
 5600
 *200
  KCC
  TLS
  JMS I 7   /ENTER INTERPRETER
  INPUT     /CALL INPUT ROUTINE
  SQUARE    /SQUARE IT
  NEGATE    /CALL OUTPUT ROUTINE
  OUTPUT    /CALL OUTPUT ROUTINE
  FEXT      /EXIT
  HLT       /HALT
$
```

DESCRIPTION OF BASIC FUNCTION

Addition

Floating-point addition is carried out by first aligning the binary points of the two numbers. This is accomplished by scaling the smaller number to the right. Then the mantissas are both scaled right once so that overflow will not occur into the sign bit. A 2's complement addition of the mantissas is then made. The result is normalized and control returns to the interpreter. This may be represented as:



Subtraction

Floating-point subtraction is accomplished by negating the operand, and then calling the addition subprogram.

Negate Operand
Call Addition

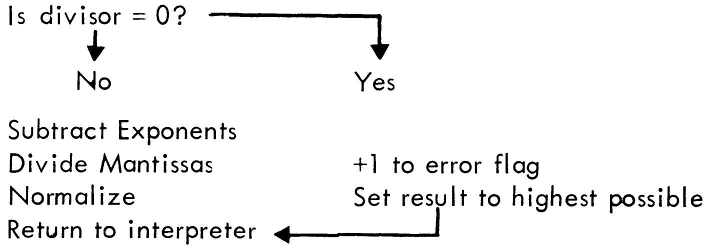
Multiplication

Floating-point multiplication is accomplished by adding the exponents together and then performing a double-precision multiplication. The result is normalized and control returns to the interpreter.

Add Exponents
Multiply Mantissas carrying result to 35 bits
Normalize
Return to Interpreter

Division

Floating-point division is accomplished by subtracting the exponent of the divisor from the exponent of the dividend. The mantissa is divided and the result is normalized. Control returns to the interpreter.



Is divisor = 0?

No

Subtract Exponents

Divide Mantissas

Normalize

Return to interpreter

Yes

+1 to error flag

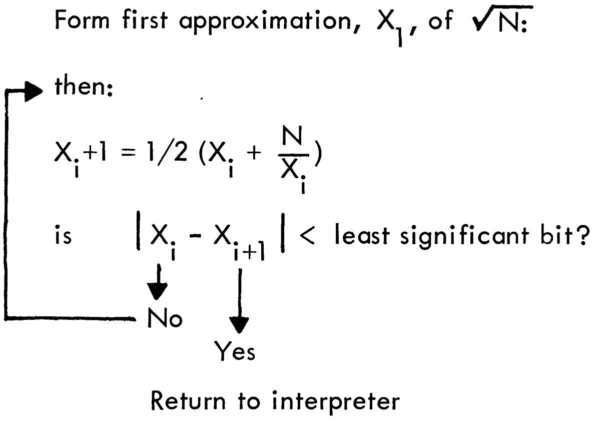
Set result to highest possible

Square

The square routine calls the multiplication routine internally.

Square Root

The square root is calculated using Newton's method in which an initial approximation is made and then each succeeding approximation is calculated. The routine exits when two successive approximations are equal to within the least significant bit of the mantissas.



Form first approximation, X_1, of $\sqrt{N}$:

then:

$X_i+1 = 1/2\ (X_i + \frac{N}{X_i})$

is $|X_i - X_{i+1}| <$ least significant bit?

No

Yes

Return to interpreter

Error Flag

Division by 0 causes C(61) to be incremented by 1. The quotient is set to the highest positive number.

Attempting to extract the square of a negative number causes C(61) to be incremented by 1. The root of the absolute value is taken.

The contents of 61 are set to 0 at the beginning of each divide and square root operation.

SUMMARY OF BASIC PACKAGE

Entry Points

5600	Arithmetic Interpreter
7400	Floating Input
7200	Floating Output

NOTE: Both the input and the output routines require that register 7 contains the interpreter entry point: 5600.

Flags

55	If ≠ 0, type carriage-return/line-feed after output.
56	If ≠ 0, follow each input carriage-return by a line-feed.
57	Contains the input terminating character.
60	Equals 0 if no valid input.
61	Is nonzero if (a) divide by 0 or (b) square root of a negative number.

Commands

FADD	1000	Floating Add
FSUB	2000	Floating Subtract
FMPY	3000	Floating Multiply
FDIV	4000	Floating Divide
FGET	5000	Floating Get
FPUT	6000	Floating Put
FNOR	7000	Floating Normalize
FEXT	0000	Floating Exit
SQUARE	0001	Square
SQROOT	0002	Square Root
	0003	
	....	Expandable
	0017	

Storage

The floating accumulator is in registers:

44 - Exponent

45 - High Order Mantissa

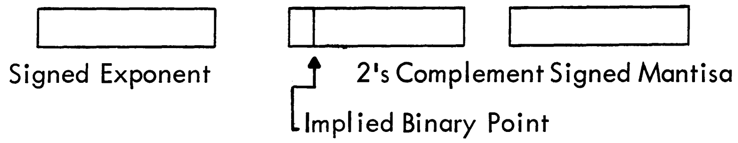
46 - Low Order Mantissa

The basic floating-point package (Digital 8-5A-S) uses memory locations:

7; 40-61; 5600-7577

FIXED TO FLOATING/FLOATING TO FIXED

Since the floating-point package stores numbers in the following format:



and since the exponent indicates where the real exponent is, this information may be used to either convert a floating-point number to fixed point or to convert a fixed-point number to floating point.

For example, assume that there is an integer in the accumulator that is less in magnitude than 2047. To float this number, the following sequence of steps may be employed:

```
          . . .
          DCA 45        /PUT INTO HIGH-ORDER MANTISSA
          DCA 46        /PUT 0 INTO LOW-ORDER MANTISSA
          TAD C13       /11 (10) INTO
C,        DCA 44        /EXPONENT
          JMS I 7       /CALL INTERPRETER
          FNOR          /NORMALIZE
          FEXT          /LEAVE INTERPRETER
          . . .
C13,      0013          /11 (DECIMAL)
```

At point C, we have set the binary point of the integer to the right end of the high order mantissa word, or eleven (decimal) locations to the right of the implicit binary point.

To float this number, the floating accumulator is scaled right until the exponent contains the location of the desired binary point. To fix the floating accumulator as an 11-bit signed integer, the following sequence of coding may be employed:

```
          ....
          CLA
          TAD 44        /FETCH EXPONENT
          SZA SMA       /IS THE NUMBER <1?
          JMP .+3       /NO:
          CLA           /YES:  FIX IT TO 0
          JMP DONE+1
          TAD M13       /NO:  SET BINARY POINT AT
          SNA           /11 (10) PLACES TO RIGHT OF CURRENT POINT
          JMP DONE      /IT IS ALREADY THERE:  ALL DONE
          SMA           /TEST TO SEE IF IT IS TOO LARGE
          JMP ERROR     /YES:  NUMBER >2**11
          DCA 44        /NO:  SET SCALE COUNT
GO,       CLL           /0 TO C(L)
          TAD 45        /FETCH MANTISSA
          SPA           /IS IT <0?
          CML           /YES:  PUT A 1 IN LEFT BIT
          RAR           /SCALE RIGHT
          DCA 45        /RESTORE IT
          ISZ 44        /TEST IF SHIFTED ENOUGH
          JMP 00        /NO:  CONTINUE
DONE,     TAD 45        /ANSWER IN C(AC)
          ....
          ....
M13,      -13           /-11 (DECIMAL)
```

This may be coded as a subroutine.

EXTENDED FLOATING-POINT PACKAGE

The extended floating-point package contains the following additional commands:

0003 Floating Sine
Take the sine of the angle in the floating accumulator (considered to be in radians) and leave the result in the floating accumulator.

0004 Floating Cosine
Take the cosine of the angle in the floating accumulator (considered to be in radians) and leave the result in the floating accumulator.

0005 Floating Arc Tangent
Take the arc tangent of the number in the floating accumulator and leave the result (in radians) in the floating accumulator.

0006 Floating Exponent
Raise e to the C(FAC) power. Leave the result in the floating accumulator.

0007 Floating Logarithm
Take the natural logarithm of the absolute value of the number in the floating accumulator.

Sine

The sine routine uses the following identities:

$SIN(-X)=-SIN(X)$

$SIN(X)=SIN(2\pi N+F)=SIN(F)$

where $0<F<2\pi$ and N is an integer:

$SIN(\pi-F)=-SIN(F)$

Using these identities, F is reduced to the range $-\pi/2<F<\pi/2$.

If $Y = 2F/\pi$ so that $-1<F<1$, then:

$$SIN(X)=(C_1F+C_3F^3+C_5F^5+C_7F^7+C_9F^9)$$

where:

$C_1 = 1.57079631847$

$C_3 = -0.64596371106$

$C_5 = +0.07968967928$

$C_7 = -0.00467376557$

$C_9 = +0.00015148419$

The sine subroutine is valid over the range:

$-2\pi .2047<X<2\pi .2047$

Cosine

The cosine routine uses the identity:

$COS(X)=SIN(\pi/2-X)$

It then uses the SINE routine.

Arc Tangent

The arc tangent routine uses the following identities:

$ARCTAN(-X)=-ARCTAN(X)$

IF X>, 1; then $ARCTAN(X)=\pi/2-ARCTAN(1/X)$

FOR $0<X<1$,

$$\mathrm{ARCTAN}(X) = \frac{X(A_0+A_1X^2+A_2X^4)}{B_0+B_1,X^2+B_2X^4}$$

where:

A_0 = +0.6402481953

A_1 = +0.4229908144

A_2 = +0.0264694361

B_0 = +0.6402487022

B_1 = +0.6363779373

B_2 = +0.1108328778

The result is in the range:

$-\pi/2<\mathrm{ARCTAN}(X)<\pi/2$

$-\infty<X<\infty$

Logarithm

The logarithm routine uses the identities:

For $X<1$;

$\mathrm{LOG}(X)=\mathrm{LOG}(2^N\cdot F)\ (1<F<2)$

$=N\mathrm{LOG}(2)+\mathrm{LOG}(F)$

$$\mathrm{LOG}(1+Y)= \sum_{C=1}^{8} A_iY^i$$

where:

A_1 = +0.9999964239

A_2 = -0.4998741238

A_3 = +0.3317990258

A_4 = +0.2407338084

A_5 = +0.1676540711

A_6 = -0.0953293897

$A_7 = +0.0360884937$

$A_8 = -0.0064553442$

for:

0<X<1, the identity:

LOG(X)=-LOG(1/X) is used.

Floating Exponent

The exponent routine uses the identity for X>0,

$$e^X = 2^{X \cdot \log_2 e} = 2^{N+F} = 2^N \cdot 2^F$$

where N is an integer and <F<1

$$2^F = 1 + \cfrac{2F}{A - F + BF^2 - \cfrac{C}{D+F^2}}$$

where:

A = +9.95459578

B = +0.03465735903

C = +617.97226053

D = +87.417497202

$\text{Log}_2 e = 1.4426950409$

If X<0

$$e^x = \frac{1}{e^{-x}}$$

Example, input A, B, and output:

$Y = \text{LOG}(\text{COS}(A/B) + \sqrt{A \cdot B})$

```
SQROOT=0002                  /DEFINITIONS TO ASSEMBLER
COS=0004
LOG=0006

*5
  7400
  7200
  5600
/USES EXTENDED INTERPRETER
```

```
*200
KCC
TLS
BEGIN,  JMS I 5         /INPUT A
        JMS I 7         /ENTER INTERPRETER
        FPUT A          /STORE A
        FEXT            /EXIT
        JMS I 5         /INPUT B
        JMS I 7         /ENTER INTERPRETER
        FPUT B          /STORE B
        FMPY A          /A·B
        SQROOT          /EXTRACT ROOT
        FPUT TEMP       /STORE IT
        FGET A          /LOAD FAC WITH A
        FDIV B          /DIVIDE BY B
        COS             /TAKE COSINE
        FADD TEMP       /ADD
        LOG             /TAKE LOG.
        FEXT
        JMS I 6         /OUTPUT ANSWER
        JMP BEGIN
A,      0
        0
        0
B,      0
        0
        0
TEMP,   0
        0
        0
$
```

Output Controller

There is an additional routine available that formats floating-point output. It is available with both the basic floating-point package and the extended package.

The controller routine requires two parameters for output formatting. It is called by an effective JMS 7200. At this point:

C(62) = total number of digits to be outputted. If C(62)=0, output in E format.

C(AC) = number of digits to the right of the decimal point. If it equals 0, do not type a ".".

If the number in the floating accumulator is larger than the field width allows, "X's" will be typed. The sign is typed and leading 0's are suppressed.

Example: If the contents of the floating accumulator are 678.234 (decimal).

C(62)	C(AC)	Output
3	0	+ 678
4	0	+ 678
4	1	+ 678.2
5	2	+ 678.23
6	2	+ 678.23
3	2	+ 678.
2	0	+XX
0	1	+ 0.678234E+03

FLOATING POINT PACKAGE VERSIONS

Four versions of the floating-point package are available:

Digital 8-5A-S

This is the basic floating-point package. It consists of the input/output package and the basic arithmetic instructions. Its core limits are:

7; 40-61; 5600-7577

Digital 8-5B-S

This is the basic floating-point package with the output controller. Its limits are:

7; 40-62; 5400-7577

The output controller does not type a carriage-return/line-feed after output and, hence, the programmer must provide his own routine to do so. Because of the way in which the typeout routine in the floating-point packages are constructed, the user must construct similar typeout routines in order to avoid timing errors in the teleprinter. In other words, the floating-point package uses a routine similar to:

```
TSF
JMP .-1
TLS
```

to do its typing. The user must use a similar routine, i.e., one that waits for the teleprinter flag to be set before executing the TLS instruction.

Digital 8-5C-S

This is the basic floating-point package with the extended functions. In addition, two interpretive commands are provided for input and output. The additional interpretive commands are:

SINE	3
COSINE	4
ARCTANGENT	5
EXPONENTIAL	6
LOG	7
INPUT	13
OUTPUT	14

Its core limits are:

7; 40-61; 4757-7577

Using this package, the routine on page 3-14 and 3-15 for calculating:

$Y=LOG(COS(A/B)+\sqrt{A\cdot B})$ may be rewritten as:

```
SQROOT=2
FCOS=4
FLOG=6
INPUT=13
OUTPUT=14

*7
 5600

*200
KCC
TLS
BEGIN,   JMS I 7      /CALL INTERPRETER
         INPUT        /READ A
         FPUT A       /STORE IT
         INPUT        /READ B
         FPUT B
         FMPY A       /A·B
         SQROOT
         FPUT TEMP /(A·B)**.5
         FGET A
         FDIV B
         FCOS
         FADD TEMP
         FLOG
         OUTPUT
         FEXT
         JMP BEGIN
A,       0
         0
         0
B,       0
         0
         0
TEMP,    0
         0
         0
$
```

Digital 8-5D-S

This is the basic package, the output controller, and the extended functions. Its core limits are:

7; 40-62; 4557-7577

CHAPTER 4

PROGRAM LISTINGS

```
                /FLOATING POINT ARITHMETIC INTERPRETER
                *40
0040    0000    EX1,        0               /OPERAND STORAGE
0041    0000    AC1H,       0
0042    0000    AC1L,       0
0043    0000    OVER1,      0
0044    0000    EXP,        0               /F.A.
0045    0000    HORD,       0
0046    0000    LORD,       0
0047    0000    OVER2,      0
0050    0000    EXP1,       0
0051    0000    QUOL,       0
0052    0000    FPAC1,      0
0053    0000                0
0054    0000                0
                *61
0061    0000    FLAG,       0               /ARITHMETIC ERROR FLAG
                *5600
5600    0000    FPNT,       0
5601    7300                CLA CLL
5602    3043                DCA OVER1
5603    3047                DCA OVER2
5604    1600                TAD I FPNT      /GET NEXT INSTRUCTION
5605    3253                DCA JUMP
5606    1253                TAD JUMP
5607    0263                AND PAGENO      /GET PAGE BIT
5610    7650                SNA CLA         /PAGE ZERO?
5611    5214                JMP .+3         /YES
5612    1261                TAD MASK5       /NO
5613    0200                AND FPNT        /C(FPNT)0-4 CONTAINS PAGE BITS
5614    3256                DCA ADDR
5615    1262                TAD MASK7       /GET 7 BIT ADDRESS
5616    0253                AND JUMP
5617    1256                TAD ADDR
5620    3256                DCA ADDR
5621    1264                TAD INDRCT      /INDIRECT BIT=1?
5622    0253                AND JUMP
5623    7650                SNA CLA
5624    5227                JMP LOOP01      /NO-GO ON
5625    1656                TAD I ADDR      /YES DEFER
5626    3256                DCA ADDR
5627    2200    LOOP01,     ISZ FPNT
5630    1656                TAD I ADDR
5631    3040                DAC EX1         /EXPONENT
5632    1256                TAD ADDR
```

```
5633   3257             DCA SAVE
5634   2257             ISZ SAVE
5635   1657             TAD I SAVE
5636   3041             DCA AC1H          /HIGH ORDER MANTISSA
5637   2257             ISZ SAVE
5640   1657             TAD I SAVE
5641   3042             DAC AC1L          /LOW ORDER MANTISSA
5642   1253             TAD JUMP
5643   7106             CLL RTL
5644   7006             RTL
5645   0260             AND MASK3         /GET BITS 0-2, IE OPCODE
5646   1265             TAD TABLE         /LOOKUP IN TABLE
5647   3254             DCA JUMP2
5650   1654             TAD I JUMP2
5651   3254             DCA JUMP2
5652   5654             JMP I JUMP2       /GO THERE
5653   0000   JUMP,     0
5654   0000   JUMP2,    0
5655   0000   GO2,      0
5656   0000   ADDR,     0
5657   0000   SAVE,     0
5660   0017   MASK3,    0017
5661   7600   MASK5,    7600
5662   0177   MASK7,    0177
5663   0200   PAGENO,   0200
5664   0400   INDRCT,   0400
5665   5666   TABLE,    .+1
5666   5742             EXIT              /TABLE USED IN INTERPRETING
5667   5716             FLAD              /BITS 0-2 OF PSEUDO
5670   5737             FLSU              /INSTRUCTION
5671   5761             FLMY
5672   6305             FLDV              /IF OPCODE=0, GO TO EXIT
5673   5676             FLGT              /AND INTERPRET BITS 8-11
5674   5705             FLPT
5675   5773             NORF
5676   1040   FLGT,     TAD EX1           /FGET=5
5677   3044             DCA EXP
5700   1041             TAD AC1H
5701   3045             DCA HORD
5702   1042             TAD AC1L
5703   3046             DCA LORD
5704   5201             JMP FPNT+1
5705   1044   FLPT,     TAD EXP           /FPUT=6
5706   3656             DCA I ADDR
5707   2256             ISZ ADDR
5710   1045             TAD HORD
5711   3656             DCA I ADDR
5712   2256             ISZ ADDR
5713   1046             TAD LORD
5714   3656             DCA I ADDR
5715   5201             JMP FPNT+1
5716   4771   FLAD,     JMS I ALGN        /FLAD=1 - FIRST ALIGN EXPONENTS
5717   5201             JMP FPNT+1        /RETURN IF NO ALIGNMENT IS POSSIBLE
5720   4772             JMS I UNORM       /LARGER OF THE TWO IS IN F.A.
```

```
5721    7100                CLL
5722    1043                TAD OVER1          /TRIPLE PRECISION ADDITION
5723    1047                TAD OVER2          /SINCE BITS ARE SHIFTED
5724    3047                DCA OVER2          /RIGHT
5725    7004                RAL
5726    1042                TAD AC1L
5727    1046                TAD LORD
5730    3046                DCA LORD
5731    7004                RAL
5732    1041                TAD AC1H
5733    1045                TAD HORD
5734    3045                DCA HORD
5735    4770                JMS I NORM         /NORMALIZE THE RESULT
5736    5201                JMP FPNT+1

5737    4741    FLSU,       JMS I OPMINS       /FSUB=2 - NEGATE THE OPERAND
5740    5316                JMP FLAD           /ADD
5741    6400    OPMINS,     MINUS2
5742    1253    EXIT,       TAD JUMP           /OPCODE=0
5743    0260                AND MASK3          /ARE BITS 8-11=0?
5744    7450                SNA
5745    5600                JMP I FPNT         /YES=FEXT
5746    1360                TAD ACON6          /LOOKUP ON TABLE
5747    3254                DCA JUMP2
5750    1654                TAD I JUMP2
5751    3254                DCA JUMP2
5752    1200                TAD FPNT
5753    3255                DCA GO2
5754    4654                JMS I JUMP2        /CALL AS A SUBROUTINE
5755    1255                TAD GO2            /RESTORE F.P. POINTER
5756    3200                DCA FPNT
5757    5201                JMP FPNT+1         /GET NEXT PSEUDO INSTRUCTION
5760    6544    ACON6,      TABLE6-1           /CALLING CAN BE TO A DEPTH ONE
5761    7201    FLMY,       CLA IAC            /FMPY=3
5762    1040                TAD EX1
5763    1044                TAD EXP            /ADD EXPONENTS TOGETHER
5764    3044                DCA EXP
5765    4767                JMS I MULT         /MULTIPLY
5766    5201                JMP FPNT+1
5767    6221    MULT,       DMULT
5770    6600    NORM,       DNORM
5771    6020    ALGN,       ALIGN
5772    6564    UNORM,      DUNORM
5773    4770    NORF,       JMS I NORM
5774    5201                JMP FPNT+1
                *6000
6000    0000    ACMINS,     0                  /ROUTINE TO PERFORM
6001    7300                CLL CLA
6002    1047                TAD OVER2          /TRIPLE PRECISION NEGATION
6003    7041                CMA IAC            /OF FLOATING AC
6004    3047                DCA OVER2
6005    1046                TAD LORD
6006    7040                CMA
6007    7430                SZL
```

```
6010   7101              CLL IAC
6011   3046              DCA LORD
6012   1045              TAD HORD
6013   7040              CMA
6014   7430              SZL
6015   7101              CLL IAC
6016   3045              DCA HORD
6017   5600              JMP I ACMINS
6020   0000   ALIGN,     0                  /SUBROUTINE TO ALIGN
6021   1045              TAD HORD           /BINARY POINTS FOR
6022   7640              SZA CLA            /ADD-SUBTRACT
6023   5227              JMP .+4
6024   1046              TAD LORD           /IS MANTISSA ZERO?
6025   7650              SNA CLA
6026   5337              JMP NOHERE         /YES
6027   1041              TAD AC1H           /IS OPERAND ZERO?
6030   7640              SZA CLA
6031   5235              JMP .+4
6032   1042              TAD AC1L
6033   7650              SNA CLA
6034   5620              JMP I ALIGN        /BOTH ARE ZERO-EXIT
6035   1040              TAD EX1
6036   7041              CMA IAC
6037   1044              TAD EXP
6040   7450              SNA                /ARE EXPONENTS EQUAL?
6041   5314              JMP DONE           /YES
6042   7500              SMA
6043   7041              CMA IAC
6044   3342              DCA AMOUNT
6045   1342              TAD AMOUNT
6046   1343              TAD TEST2
6047   7700              SMA CLA            /CAN EXPONENTS BE ALIGNED?
6050   5256              JMP .+6            /YES
6051   4316              JMS OUTGO          /NO
6052   7430              SZL
6053   1350              TAD TAG2
6054   1347              TAD TAG1
6055   5325              JMP NOGO
6056   4316              JMS OUTGO
6057   7420              SNL                /SET UP ADDRESSES
6060   1350              TAD TAG2
6061   1347              TAD TAG1
6062   3344              DCA TEST3
6063   1342              TAD AMOUNT
6064   7041              CMA IAC
6065   1744              TAD I TEST3
6066   3744              DCA I TEST3
6067   2344              ISZ TEST3
6070   1344              TAD TEST3
6071   3345              DCA TEST4
6072   2345              ISZ TEST4
6073   1345              TAD TEST4
6074   3346              DCA TEST5
6075   2346              ISZ TEST5
```

```
6076   7100   SHIFT,     CLL             /THIS ROUTINE DOES
6077   1744              TAD I TEST3     /THE ACUTAL SHIFTING
6100   7510              SPA
6101   7020              CML
6102   7010              RAR
6103   3744              DCA I TEST3
6104   1745              TAD I TEST4
6105   7010              RAR
6106   3745              DCA I TEST4
6107   1746              TAD I TEST5
6110   7010              RAR
6111   3746              DCA I TEST5
6112   2342              ISZ AMOUNT
6113   5276              JMP SHIFT
6114   2220   DONE,      ISZ ALIGN
6115   5620              JMP I ALIGN
6116   0000   OUTGO,     0               /DETERMINE WHICH TO SHIFT
6117   1040              TAD EX1
6120   7041              CMA IAC
6121   1044              TAD EXP
6122   7004              RAL
6123   7200              CLA
6124   5716              JMP I OUTGO
6125   3344   NOGO,      DCA TEST3       /CAN'T BE ALIGNED
6126   1744              TAD I TEST3     /LARGEST GOES INTO FAC
6127   3044              DCA EXP
6130   2344              ISZ TEST3
6131   1744              TAD I TEST3
6132   3045              DCA HORD
6133   2344              ISZ TEST3
6134   1744              TAD I TEST3
6135   3046              DCA LORD
6136   5620              JMP I ALIGN
6137   1040   NOHERE,    TAD EX1         /MANTISSA=0
6140   3044              DCA EXP
6141   5314              JMP DONE
6142   0000   AMOUNT,    0
6143   0030   TEST2,     0030
6144   0000   TEST3,     0
6145   0000   TEST4,     0
6146   0000   TEST5,     0
6147   0044   TAG1,      EXP
6150   7774   TAG2,      EX1-EXP
6151   5601   RETN2,     FPNT+1
6152   1362   ERROR1,    TAD GOOF        /DIVISION BY ZERO
6153   3044              DCA EXP         /SET TO LARGEST + VALUE
6154   1362              TAD GOOF
6155   3045              DCA HORD
6156   7040              CMA
6157   3046              DCA LORD
6160   2061              ISZ FLAG        /SET FLAG
6161   5751              JMP I RETN2
6162   3777   GOOF,      3777
```

```
6163   0000   SQUARE,   0
6164   4407             JMS I 0007
6165   6052             FPUT FPAC1
6166   3052             FMPY FPAC1
6167   0000             FEXT
6170   5763             JMP I SQUARE
6171   0000   EXIT6,    0                  /DUMMY SUBROUTINE
6172   5771             JMP I EXIT6
              *6200
6200   0000   DIV1,     0                  /SHIFT FAC RIGHT
6201   7300             CLA CLL
6202   1045             TAD HORD
6203   7510             SPA
6204   7120             CLL CML
6205   7010             RAR
6206   3045             DCA HORD
6207   1046             TAD LORD
6210   7010             RAR
6211   3046             DCA LORD
6212   1047             TAD OVER2
6213   7010             RAR
6214   3047             DCA OVER2
6215   7100             CLL
6216   2044             ISZ EXP
6217   7000             NOP
6220   5600             JMP I DIV1
6221   0000   DMULT,    0                  /DOUBLE PRECISION MULTIPLY
6222   7300             CLA CLL            /SAVE PRODUCT TRIPLE PRECISION
6223   1363             TAD SMACLA
6224   3347             DCA SNSWIT         /CALLS A SINGLE PRECISION
6225   4336             JMS SIGN           /MULTIPLY 3 TIMES
6226   1042             TAD AC1L
6227   3756             DCA I MP2PT
6230   1046             TAD LORD
6231   4755             JMS I MP4PT
6232   7200             CLA
6233   1757             TAD I MP5PT
6234   3047             DCA OVER2
6235   1045             TAD HORD
6236   3756             DCA I MP2PT
6237   1042             TAD AC1L
6240   4755             JMS I MP4PT
6241   1047             TAD OVER2
6242   3047             DCA OVER2
6243   7004             RAL
6244   1757             TAD I MP5PT
6245   3367             DCA D
6246   7004             RAL
6247   3370             DCA KEEP
6250   1041             TAD AC1H
6251   3756             DCA I MP2PT
6252   1046             TAD LORD
6253   4755             JMS I MP4PT
```

```
6254    1047               TAD OVER2
6255    3047               DCA OVER2
6256    7004               RAL
6257    1757               TAD I MP5PT
6260    1367               TAD D
6261    3367               DCA D
6262    7004               RAL
6263    1370               TAD KEEP
6264    3370               DCA KEEP
6265    1045               TAD HORD
6266    3756               DCA I MP2PT
6267    1041               TAD AC1H
6270    4755               JMS I MP4PT
6271    1367               TAD D
6272    3046               DCA LORD
6273    7004               RAL
6274    1757               TAD I MP5PT
6275    1370               TAD KEEP
6276    3045               DCA HORD
6277    4760               JMS I NORMF
6300    3047               DCA OVER2
6301    2365               ISZ SGN
6302    5621               JMP I DMULT
6303    4773               JMS I MINS
6304    5621               JMP I DMULT
6305    1041    FLDV,      TAD AC1H
6306    7640               SZA CLA
6307    5313               JMP .+4
6310    1042               TAD AC1L
6311    7650               SNA CLA
6312    5774               JMP I ERROR       /DIVISION BY ZERO
6313    1040               TAD EX1
6314    7041               CMA IAC
6315    1044               TAD EXP
6316    7001               IAC
6317    3044               DCA EXP           /SUBTRACT EXPONENTS
6320    1362               TAD SPACLA
6321    3347               DCA SNSWIT
6322    4336               JMS SIGN          /SET UP SIGNS
6323    4761               JMS I DIVIDE      /DIVIDE
6324    1757               TAD I MP5PT
6325    1041               TAD AC1H          /ROUND OFF IN 23RD BIT
6326    7630               SZL CLA
6327    7001               IAC
6330    3042               DCA AC1L
6331    3041               DCA AC1H
6332    2365               ISZ SGN           /TEST SIGN
6333    4773               JMS I MINS        /NEGATE
6334    5735               JMP I .+1         /ADD ROUNDING
6335    5720               FLAD+2
6336    0000    SIGN,      0                 /TEST SIGN OF RESULT
6337    1366               TAD REST          /SET UP BY MULTIPLY AND
6340    3365               DCA SGN           /DIVIDE
```

```
6341    1045                TAD HORD
6342    7700                SMA CLA
6343    5346                JMP .+3
6344    4773                JMS I MINS
6345    2365                ISZ SGN
6346    1041                TAD AC1H
6347    7700    SNSWIT,     SMA CLA           /OR SPA CLA
6350    5736                JMP I SIGN
6351    4771                JMS I MINS2
6352    2365                ISZ SGN
6353    7000                NOP
6354    5736                JMP I SIGN
6355    6437    MP4PT,      MP4
6356    6471    MP2PT,      MP2
6357    6465    MP5PT,      MP5
6360    6600    NORMF,      DNORM
6361    6472    DIVIDE,     DUBDIV
6362    7710    SPACLA,     SPA CLA
6363    7700    SMACLA,     SMA CLA
6364    5601    RETURN,     FPNT+1
6365    0000    SGN,        0
6366    7776    REST,       7776
6367    0000    D,          0
6370    0000    KEEP,       0
6371    6400    MINS2,      MINUS2            /-AC1H,AC1L
6372    6420    RAR2,       DIV2              /AC1H,AC1L/2
6373    6000    MINS,       ACMINS
6374    6152    ERROR,      ERROR1
                *6400
6400    0000    MINUS2,     0                 /NEGATE OPERAND
6401    7300                CLA CLL           /TRIPLE PRECISION
6402    1043                TAD OVER1
6403    7041                CMA IAC
6404    3043                DCA OVER1
6405    1042                TAD AC1L
6406    7040                CMA
6407    7430                SZL
6410    7101                CLL IAC
6411    3042                DCA AC1L
6412    1041                TAD AC1H
6413    7040                CMA
6414    7430                SZL
6415    7101                CLL IAC
6416    3041                DCA AC1H
6417    5600                JMP I MINUS2
6420    0000    DIV2,       0                 /SHIFT OPERAND RIGHT
6421    7300                CLA CLL           /TRIPLE PRECISION
6422    1041                TAD AC1H
6423    7510                SPA
6424    7120                CLL CML
6425    7010                RAR
6426    3041                DCA AC1H
6427    1042                TAD AC1L
```

```
6430    7010                RAR
6431    3042                DCA AC1L
6432    1043                TAD OVER1
6433    7010                RAR
6434    3043                DCA OVER1
6435    7100                CLL
6436    5620                JMP I DIV2
6437    0000    MP4,        0                   /SINGLE PRECISION MULTIPLY
6440    3266                DCA MP1             /12 BITS BY 12 BITS
6441    3265                DCA MP5
6442    1270                TAD THIR
6443    3267                DCA MP3
6444    7100                CLL
6445    1266                TAD MP1
6446    7010                RAR
6447    3266                DCA MP1
6450    1265                TAD MP5
6451    7420                SNL
6452    5255                JMP .+3
6453    7100                CLL
6454    1271                TAD MP2
6455    7010                RAR
6456    3265                DCA MP5
6457    2267                ISZ MP3
6460    5245                JMP MP4+6
6461    1266                TAD MP1
6462    7010                RAR
6463    7100                CLL
6464    5637                JMP I MP4
6465    0000    MP5,        0
6466    0000    MP1,        0
6467    0000    MP3,        0
6470    7764    THIR,       7764
6471    0000    MP2,        0
6472    0000    DUBDIV,     0                   /DOUBLE PRECISION DIVIDE
6473    7300                CLA CLL
6474    3051                DCA QUOL
6475    1344                TAD MIF
6476    3267                DCA MP3
6477    5306                JMP DVX
6500    1046    DV3,        TAD LORD
6501    7004                RAL
6502    3046                DCA LORD
6503    1045                TAD HORD
6504    7004                RAL
6505    3045                DCA HORD
6506    1042    DVX,        TAD AC1L
6507    1046                TAD LORD
6510    3271                DCA MP2
6511    7004                RAL
6512    1045                TAD HORD
6513    1041                TAD AC1H
6514    7420                SNL
```

```
6515    5321                JMP DV2-1
6516    3045                DCA HORD
6517    1271                TAD MP2
6520    3046                DCA LORD
6521    7200                CLA
6522    1051    DV2,        TAD QUOL
6523    7004                RAL
6524    3051                DCA QUOL
6525    1047                TAD OVER2
6526    7004                RAL
6527    3047                DCA OVER2
6530    2267                ISZ MP3
6531    5300                JMP DV3
6532    1051                TAD QUOL
6533    3046                DCA LORD
6534    1045                TAD HORD
6535    7106                CLL RTL
6536    3265                DCA MP5
6537    1047                TAD OVER2
6540    3045                DCA HORD
6541    3047                DCA OVER2
6542    1265                TAD MP5
6543    5672                JMP I DUBDIV
6544    7751    MIF,        7751
6545    6163    TABLE6,     SQUARE          /TABLE FOR INTERPRETATION
6546    6656                SQROOT          /OF BITS 8-11
6547    6171                EXIT6           /CONTAINS ABSOLUTE ADDRESSES
6550    6171                EXIT6           /OF PROGRAMS CALLED AS
6551    6171                EXIT6           /SUBROUTINES
6552    6171                EXIT6           /EXIT6=A DUMMY OR NOP
6553    6171                EXIT6
6554    6171                EXIT6
6555    6171                EXIT6
6556    6171                EXIT6
6557    6171                EXIT6
6560    6171                EXIT6
6561    6171                EXIT6
6562    6171                EXIT6
6563    6171                EXIT6

6564    0000    DUNORM,     0
6565    4220                JMS DIV2        /SHIFT OPERAND RIGHT
6566    4772                JMS I RAR1
6567    2040                ISZ EX1
6570    7000                NOP
6571    5764                JMP I DUNORM
6572    6200    RAR1,       DIV1
                *6600
6600    0000    DNORM,      0               /SUBROUTINE TO NORMALIZE
6601    7300                CLA CLL         /FLOATING ACCUMULATOR
6602    3255                DCA AMT1
6603    3254                DCA SIGN1
6604    1045                TAD HORD
6605    7510                SPA             /IS MANTISSA NEGATIVE?
```

```
6606    2254                ISZ SIGN1        /YES
6607    7640                SZA CLA          /IS MANTISSA=0?
6610    5217                JMP GO6          /NO
6611    1046                TAD LORD
6612    7640                SZA CLA
6613    5217                JMP GO6
6614    1047                TAD OVER2
6615    7650                SNA CLA
6616    5251                JMP EXIT2        /YES
6617    1254    GO6,        TAD SIGN1        /NO
6620    7640                SZA CLA          /NEGATIVE?
6621    4653                JMS I NEG        /YES
6622    1045    LOP,        TAD HORD         /WILL SHIFT BE TOO FAR?
6623    7104                RAL CLL
6624    7710                SPA CLA
6625    5241                JMP EXIT1        /YES
6626    1047                TAD OVER2
6627    7104                CLL RAL
6630    3047                DCA OVER2
6631    1046                TAD LORD         /NO SHIFT MANTISSA LEFT
6632    7004                RAL
6633    3046                DCA LORD
6634    1045                TAD HORD
6635    7004                RAL
6636    3045                DCA HORD
6637    2255                ISZ AMT1         /COUNT NO. OF TIMES SHIFTED
6640    5222                JMP LOP
6641    1255    EXIT1,      TAD AMT1         /CORRECT EXPONENT
6642    7041                CMA IAC
6643    1044                TAD EXP
6644    3044                DCA EXP
6645    1254                TAD SIGN1        /NEGATIVE?
6646    7640                SZA CLA
6647    4653                JMS I NEG        /YES
6650    5600                JMP I DNORM
6651    3044    EXIT2,      DCA EXP          /SET TO ZERO
6652    5600                JMP I DNORM
6653    6000    NEG,        ACMINS
6654    0000    SIGN1,      0
6655    0000    AMT1,       0
6656    0000    SQROOT,     0                /FLOATING SQUARE ROOT
6657    3362                DCA FLAG1        /TAKES ROOT OF ABSOLUTE
6660    4407                JMS I 007
6661    6052                FPUT FPAC1       /VALUE
6662    0000                FEXT             /NEWTON'S METHOD IS USED
6663    1045                TAD HORD
6664    7710                SPA CLA
6665    5351                JMP SQEND1       /NUMBER IS NEGATIVE
6666    1044                TAD EXP
6667    7100                CLL
6670    7510                SPA
6671    7020                CML
6672    7010                RAR
6673    3356                DCA ITER1        /MAKE FIRST APPROXIMATION
```

```
6674    7430                 SZL
6675    2356                 ISZ ITER1
6676    7000                 NOP
6677    1361                 TAD SQCON1
6700    3357                 DCA ITER1+1
6701    3360                 DCA ITER1+2
6702    1053                 TAD FPAC1+1
6703    7640                 SZA CLA
6704    5310                 JMP CLCU
6705    1054                 TAD FPAC1+2
6706    7650                 SNA CLA
6707    5354                 JMP SQEND         /NUMBER=0
6710    4407    CLCU,        JMS I 0007
6711    5052                 FGET FPAC1
6712    4356                 FDIV ITER1
6713    1356                 FADD ITER1
6714    0000                 FEXT
6715    7240                 CLA CMA
6716    1044                 TAD EXP
6717    3044                 DCA EXP
6720    1044                 TAD EXP
6721    7041                 CMA IAC
6722    1356                 TAD ITER1
6723    7640                 SZA CLA           /ARE EXPONENTS EQUAL?
6724    5345                 JMP ROOTGO        /NO
6725    1045                 TAD HORD          /ARE HIGH-ORDER MANTISSAS EQUAL?
6726    7041                 CMA IAC
6727    1357                 TAD ITER1+1
6730    7640                 SZA CLA
6731    5345                 JMP ROOTGO        /NO
6732    1046                 TAD LORD
6733    7041                 CMA IAC
6734    1360                 TAD ITER1+2       /DO LOW-ORDER MANTISSAS AGREE?
6735    7500                 SMA
6736    7041                 CMA IAC           /WITHIN ONE BIT?
6737    7001                 IAC
6740    7710                 SPA CLA
6741    5345                 JMP ROOTGO        /NO
6742    1362                 TAD FLAG1
6743    3061                 DCA FLAG
6744    5656                 JMP I SQROOT
6745    4407    ROOTGO,      JMS I 0007
6746    6356                 FPUT ITER1
6747    0000                 FEXT
6750    5310                 JMP CLCU
6751    4653    SQEND1,      JMS I NEG         /NEGATE FAC
6752    2362                 ISZ FLAG1
6753    5260                 JMP SQROOT+2
6754    3044    SQEND,       DCA EXP
6755    5656                 JMP I SQROOT
6756    0000    ITER1,       0
6757    0000                 0
6760    0000                 0
6761    3015    SQCON1,      3015
6762    0000    FLAG1,       0
```

ACMINS	6000
ACON6	5760
AC1H	0041
AC1L	0042
ADDR	5656
ALGN	5771
ALIGN	6020
AMOUNT	6142
AMT1	6655
CLCU	6710
D	6367
DIVIDE	6361
DIV1	6200
DIV2	6420
DMULT	6221
DNORM	6600
DONE	6114
DUBDIV	6472
DUNORM	6564
DVX	6506
DV2	6522
DV3	6500
ERROR	6374
ERROR1	6152
EXIT	5742
EXIT1	6641
EXIT2	6651
EXIT6	6171
EXP	0044
EXP1	0050
EX1	0040
FLAD	5716
FLAG	0061
FLAG1	6762
FLDV	6305
FLGT	5676
FLMY	5761
FLPT	5705
FLSU	5737
FPAC1	0052
FPNT	5600
GOOF	6162
GO2	5655
GO6	6617
HORD	0045
INDRCT	5664
ITER1	6756
JUMP	5653
JUMP2	5654
KEEP	6370
LOOP01	5627
LOP	6622
LORD	0046
MASK3	5660
MASK5	5661
MASK7	5662
MIF	6544
MINS	6373
MINS2	6371
MINUS2	6400
MP1	6466
MP2	6471
MP2PT	6356
MP3	6467
MP4	6437
MP4PT	6355
MP5	6465
MP5PT	6357
MULT	5767
NEG	6653
NOGO	6125
NOHERE	6137
NORF	5773
NORM	5770
NORMF	6360
OPMINS	5741
OUTGO	6116
OVER1	0043
OVER2	0047
PAGENO	5663
QUOL	0051
RAR1	6572
RAR2	6372
REST	6366
RETN2	6151
RETURN	6364
ROOTGO	6745
SAVE	5657
SGN	6365
SHIFT	6076
SIGN	6336
SIGN1	6654
SMACLA	6363
SNSWIT	6347
SPACLA	6362
SQCON1	6761
SQEND	6754
SQEND1	6751
SQROOT	6656
SQUARE	6163
TABLE	5665
TABLE6	6545
TAG1	6147
TAG2	6150
TEST2	6143
TEST3	6144
TEST4	6145
TEST5	6146
THIR	6470
UNORM	5772

```
                /FLOATING POINT I/O ROUTINES
                /REQUIRES FLOATING POINT INTERPRETER
                /ENTRY AT ØØØ7

                *7
ØØØ7    56ØØ    FPNT,       56ØØ

                *44
ØØ44    ØØØØ    EXPONT,     Ø
ØØ45    ØØØØ    HORDER,     Ø
ØØ46    ØØØØ    LORDER,     Ø

                *52
ØØ52    ØØØØ    FPAC1,      Ø
ØØ53    ØØØØ                Ø
ØØ54    ØØØØ                Ø
ØØ55    7777    SWIT1,      7777        /IF = Ø, NO CR-LF AFTER OUTPUT
ØØ56    7777    SWIT2,      7777        /IF = Ø, NO LF AFTER CR IN INPUT
ØØ57    ØØØØ    CHAR,       Ø           /CONTAINS LAST CHARACTER READ
ØØ6Ø    ØØØØ    DSWIT,      Ø           /= Ø IF NO CONVERSION TOOK PLACE

                *6767
6767    ØØØØ    PRCHAR,     Ø
677Ø    1Ø56                TAD SWIT2
6771    765Ø                SNA CLA
6772    5767                JMP I PRCHAR
6773    1377                TAD LFED
6774    4776                JMS I OPUT
6775    5767                JMP I PRCHAR
6776    7344    OPUT,       OUT
6777    Ø212    LFED,       Ø212
```

```
                   /DOUBLE PRECISION DECIMAL-BINARY
                   /INPUT AND CONVERSION
                   *7ØØØ

7ØØØ   ØØØØ   DECONV,    Ø
7ØØ1   72ØØ              CLA                   /INITIALIZE MANTISSA
7ØØ2   3Ø45              DCA HORDER
7ØØ3   3Ø46              DCA LORDER
7ØØ4   3265              DCA SIGN
7ØØ5   3266              DCA DNUMBR
7ØØ6   4342              JMS INPUT
7ØØ7   1336              TAD PLUS              /TEST FOR SIGN
7Ø1Ø   745Ø              SNA
7Ø11   5217              JMP DECON
7Ø12   1335              TAD MINUS
7Ø13   744Ø              SZA
7Ø14   522Ø              JMP .+4
7Ø15   724Ø              CLA CMA
7Ø16   3265              DCA SIGN              /IF-, SET SWITCH
7Ø17   4342   DECON,     JMS INPUT
7Ø2Ø   72ØØ              CLA
7Ø21   1Ø57              TAD CHAR              /IS IT A DIGIT
7Ø22   1337              TAD MIN9
7Ø23   75ØØ              SMA
7Ø24   56ØØ              JMP I DECONV          /NO
7Ø25   134Ø              TAD PLUS12
7Ø26   751Ø              SPA
7Ø27   56ØØ              JMP I DECONV          /NO
7Ø3Ø   3263              DCA DIGIT             /YES
7Ø31   1Ø45              TAD HORDER
7Ø32   Ø341              AND MASK              /OVERFLOW?
7Ø33   744Ø              SZA
7Ø34   5217              JMP DECON             /YES-IGNORE
7Ø35   2Ø6Ø              ISZ DSWIT
7Ø36   2266              ISZ DNUMBR            /INDEX NUMBER OF DIGITS
7Ø37   4241              JMS MULT1Ø
7Ø4Ø   5217              JMP DECON             /CONTINUE
7Ø41   ØØØØ   MULT1Ø,    Ø                     /ROUTINE TO MULTIPLY
7Ø42   1Ø46              TAD LORDER           /DOUBLE PRECISION WORD
7Ø43   3261              DCA LTEMP             /BY TEN (DECIMAL)
7Ø44   1Ø45              TAD HORDER            /REMAIN=REMAINDER
7Ø45   3262              DCA HTEMP
7Ø46   3264              DCA REMAIN
7Ø47   4267              JMS MULT2             /CALL SUBROUTINE TO
7Ø5Ø   4267              JMS MULT2             /MULTIPLY BY TWO
7Ø51   43Ø3              JMS DUBLAD            /CALL DOUBLE ADD
7Ø52   4267              JMS MULT2
7Ø53   1263              TAD DIGIT             /ADD LAST DIGIT RECEIVED
7Ø54   3261              DCA LTEMP
7Ø55   3262              DCA HTEMP
7Ø56   43Ø3              JMS DUBLAD
7Ø57   1264              TAD REMAIN            /EXIT WITH REMAINDER
7Ø6Ø   5641              JMP I MULT1Ø          /IN AC

7Ø61   ØØØØ   LTEMP,     Ø                     /DOUBLE PRECISION WORD
7Ø62   ØØØØ   HTEMP,     Ø
7Ø63   ØØØØ   DIGIT,     Ø                     /STORAGE FOR DIGIT
7Ø64   ØØØØ   REMAIN,    Ø
7Ø65   ØØØØ   SIGN,      Ø                     /=Ø IF PLUS: =7777 IF MINUS
7Ø66   ØØØØ   DNUMBR,    Ø                     /=NUMBER OF DIGITS
7Ø67   ØØØØ   MULT2,     Ø                     /MULTIPLY LORDER, HORDER BY 2
```

```
7Ø7Ø  73ØØ            CLA CLL
7Ø71  1Ø46            TAD LORDER
7Ø72  7ØØ4            RAL
7Ø73  3Ø46            DCA LORDER
7Ø74  1Ø45            TAD HORDER
7Ø75  7ØØ4            RAL
7Ø76  3Ø45            DCA HORDER
7Ø77  1264            TAD REMAIN
71ØØ  7ØØ4            RAL
71Ø1  3264            DCA REMAIN
71Ø2  5667            JMP I MULT2
71Ø3  ØØØØ  DUBLAD,   Ø                     /DOUBLE PRECISION ADDITION
71Ø4  73ØØ            CLA CLL
71Ø5  1Ø46            TAD LORDER
71Ø6  1261            TAD LTEMP
71Ø7  3Ø46            DCA LORDER
711Ø  7ØØ4            RAL
7111  1Ø45            TAD HORDER
7112  1262            TAD HTEMP
7113  3Ø45            DCA HORDER
7114  7ØØ4            RAL
7115  1264            TAD REMAIN
7116  3264            DCA REMAIN
7117  57Ø3            JMP I DUBLAD
712Ø  ØØØØ  MSIGN,    Ø                     /ROUTINE TO FORM
7121  73ØØ            CLA CLL               /2'S COMPLEMENT IF
7122  2265            ISZ SIGN              /MINUS
7123  572Ø            JMP I MSIGN           /SIGN=ØØØØ: EXIT
7124  1Ø46            TAD LORDER
7125  7Ø41            CMA IAC
7126  3Ø46            DCA LORDER
7127  1Ø45            TAD HORDER
713Ø  7Ø4Ø            CMA
7131  743Ø            SZL
7132  7ØØ1            IAC
7133  3Ø45            DCA HORDER
7134  572Ø            JMP I MSIGN

7135  7776  MINUS,    253-255               /TEST FOR SIGN
7136  7525  PLUS,     -253
7137  75Ø6  MIN9,     -272                  /TEST FOR DIGIT
714Ø  ØØ12  PLUS12,   272-26Ø
7141  76ØØ  MASK,     76ØØ                  /TEST FOR OVERFLOW
            /INPUT A CHARACTER, IF CR, TEST
            /INPUT SWITCH TO SEE IF LF SHOULD
            /BE TYPED. IF RUBOUT, RESTART INPUT
7142  ØØØØ  INPUT,    Ø                     /INPUT A CHARACTER
7143  72ØØ            CLA
7144  6Ø31            KSF
7145  5344            JMP .-1
7146  6Ø36            KRB
7147  3Ø57            DCA CHAR
715Ø  1Ø57            TAD CHAR
7151  4766            JMS I OUTPUT
7152  1Ø57            TAD CHAR
7153  745Ø            SNA
7154  5343            JMP INPUT+1           /IGNORE BLANKS
7155  137Ø            TAD MRBOUT
7156  745Ø            SNA
7157  5767            JMP I RESTRT          /RUBOUT-RESTART INPUT
716Ø  1371            TAD MINCR
```

```
7161  765Ø           SNA CLA
7162  4765           JMS I PRINT           /CR - SEE IF TO BE FOLLOWED
7163  1Ø57           TAD CHAR              /BY LF
7164  5742           JMP I INPUT           /EXIT ROUTINE

7165  6767  PRINT,   PRCHAR
7166  7344  OUTPUT,  OUT
7167  74Ø1  RESTRT,  FLINTP+1
717Ø  74Ø1  MRBOUT,  -377
7171  Ø162  MINCR,   377-215


            /FLOATING OUTPUT "E" FORMAT
            /USES:    TSF
            /         JMP .-1
            /         TLS
            *72ØØ
72ØØ  ØØØØ  FLOUTP,  Ø
72Ø1  4217           JMS FOUTCN            /CONVERT MANTISSA AND OUTPUT
72Ø2  1324           TAD BEXP
72Ø3  3Ø44           DCA EXPONT
72Ø4  1343           TAD CHE
72Ø5  4344           JMS OUT
72Ø6  4737           JMS I FEXPPT          /CONVERT EXPONENT AND OUTPUT
72Ø7  1Ø55           TAD SWIT1             /PRINT CR-LF?
721Ø  765Ø           SNA CLA
7211  56ØØ           JMP I FLOUTP          /NO-EXIT
7212  1341           TAD CARRTN /YES
7213  4344           JMS OUT
7214  1342           TAD LNFEED
7215  4344           JMS OUT
7216  56ØØ           JMP I FLOUTP          /EXIT
            /THIS WHOLE SUBROUTINE MAY BE ALTERED TO BUFFER
            /THE OUTPUT DIGITS : CHANGE JMS OUTDG TO DCA I 1Ø, ETC.
7217  ØØØØ  FOUTCN,  Ø
722Ø  73ØØ           CLA CLL
7221  1Ø45           TAD HORDER            /NUMBER>Ø??
7222  771Ø           SPA CLA
7223  722Ø           CLA CML               /NO SET LINK
7224  1327           TAD SPLUS             /YES
7225  743Ø           SZL
7226  133Ø           TAD SMINUS /NO
7227  4344           JMS OUT
723Ø  4352           JMS OUTDG             /OUTPUT "Ø"
7231  1331           TAD PERIOD
7232  4344           JMS OUT               /OUTPUT "."
7233  73ØØ           CLA CLL
7234  1Ø45           TAD HORDER
7235  77ØØ           SMA CLA
7236  5242           JMP FGO1
7237  7Ø4Ø           CMA                   /NUMBER IS NEGATIVE
724Ø  3733           DCA I SNPT            /NEGATE
7241  4732           JMS I MSNPT
7242  724Ø  FGO1,    CLA CMA               /SUBTRACT 1 FROM BINARY EXPONENT
7243  1Ø44           TAD EXPONT            /COMPENSATE AT FGO4
7244  3Ø44           DCA EXPONT
7245  3324           DCA BEXP              /INITIALIZE DECIMAL EXPONENT
7246  1Ø44  FGO2,    TAD EXPONT            /IS -4<EXPONENT<-1
7247  75ØØ           SMA
725Ø  5263           JMP FGO3              /TOO LARGE: MULTIPLY BY 1/1Ø
7251  1326           TAD FOUR
```

```
7252  77ØØ            SMA CLA
7253  527Ø            JMP FGO4
7254  44Ø7            JMS I FPNT               /TOO SMALL-TIMES TEN
7255  374Ø            FMPY I TENPT             /TEN
7256  ØØØØ            FEXT
7257  724Ø            CLA CMA
726Ø  1324            TAD BEXP
7261  3324            DCA BEXP
7262  5246            JMP FGO2
7263  44Ø7  FGO3,     JMS I FPNT
7264  3372            FMPY C.1Ø                /ONE TENTH
7265  ØØØØ            FEXT
7266  2324            ISZ BEXP
7267  5246            JMP FGO2
727Ø  3734  FGO4,     DCA I DPT                /MULTIPLY BY TWO
7271  4736            JMS I M2PT               /IE.SHIFT LEFT
7272  4735            JMS I M1ØPT              /MULTIPLY BY TEN
7273  741Ø            SKP
7274  4357  FGO5A,    JMS DIVTWO               /COMPENSATE FOR
7275  2Ø44            ISZ EXPONT               /BINARY EXPONENT
7276  5274            JMP FGO5A
7277  745Ø            SNA                      /IS FIRST DIGIT A ZERO
73ØØ  5311            JMP FGO7                 /YES, IGNORE
73Ø1  4352  FGO6,     JMS OUTDG                /MULTIPLICATIONS YIELD
73Ø2  1325            TAD MINUS7               /DECIMAL DIGITS AS HIGH
73Ø3  3Ø44            DCA EXPONT               /ORDER REMAINDERS
73Ø4  4735  FGO6A,    JMS I M1ØPT              /IE. .672X1Ø=6+.72.. ETC
73Ø5  4352            JMS OUTDG
73Ø6  2Ø44            ISZ EXPONT               /7 DIGITS OUTPUT??
73Ø7  53Ø4            JMP FGO6A                /NO: CONTINUE
731Ø  5617            JMP I FOUTCN             /YES:EXIT

7311  724Ø  FGO7,     CLA CMA                  /IGNORE FIRST DIGIT
7312  1324            TAD BEXP                 /SUBTRACT 1 FROM
7313  3324            DCA BEXP                 /DECIMAL EXPONENT
7314  1Ø45            TAD HORDER
7315  764Ø            SZA CLA
7316  5322            JMP .+4                  /IS MANTISSA ZERO?
7317  1Ø46            TAD LORDER
732Ø  765Ø            SNA CLA
7321  3324            DCA BEXP                 /YES:EXP=Ø
7322  724Ø            CLA CMA
7323  53Ø2            JMP FGO6+1

7324  ØØØØ  BEXP,     Ø                        /CONTAINS DECIMAL EXPONENT
7325  7772  MINUS7,   7772                     /NUMBER OF DIGTS OUTPUT
7326  ØØØ4  FOUR,     ØØØ4
7327  Ø253  SPLUS,    253
733Ø  ØØØ2  SMINUS,   255-253
7331  Ø256  PERIOD,   256
7332  712Ø  MSNPT,    MSIGN
7333  7Ø65  SNPT,     SIGN                     /POINTERS
7334  7Ø63  DPT,      DIGIT
7335  7Ø41  M1ØPT,    MULT1Ø
7336  7Ø67  M2PT,     MULT2
7337  7522  FEXPPT,   FEXC
734Ø  75Ø4  TENPT,    TEN
7341  Ø215  CARRTN,   Ø215
7342  Ø212  LNFEED,   Ø212
7343  Ø3Ø5  CHE,      3Ø5
```

```
7344  ØØØØ  OUT,      Ø                       /OUTPUT ONE ASCII CHARACTER
7345  6Ø41            TSF
7346  5345            JMP .-1
7347  6Ø46            TLS
735Ø  72ØØ            CLA
7351  5744            JMP I OUT

7352  ØØØØ  OUTDG,    Ø                       /OUTPUT ONE DIGIT
7353  1356            TAD C26Ø
7354  4344            JMS OUT
7355  5752            JMP I OUTDG

7356  Ø26Ø  C26Ø,     Ø26Ø

7357  ØØØØ  DIVTWO,   Ø                       /DIVIDE BY TWO IE.
736Ø  711Ø            CLL RAR                 /ROTATE RIGHT
7361  3344            DCA OUT                 /TEMPORARY STORAGE
7362  1Ø45            TAD HORDER
7363  7Ø1Ø            RAR
7364  3Ø45            DCA HORDER
7365  1Ø46            TAD LORDER
7366  7Ø1Ø            RAR
7367  3Ø46            DCA LORDER
737Ø  1344            TAD OUT
7371  5757            JMP I DIVTWO
7372  7775  C.1Ø,     7775                    /CONSTANT .1Ø USED IN
7373  3146            3146                    /FLOATING OUTPUT-PROVIDES
7374  3147            3147
            /FLOATING POINT INPUT
            *74ØØ
74ØØ  ØØØØ  FLINTP,   Ø
74Ø1  724Ø            CLA CMA                 /INITIALIZE "PERIOD SWITCH"
74Ø2  3313            DCA PRSW
74Ø3  3Ø6Ø            DCA DSWIT
74Ø4  4716            JMS I DPCVPT            /7777 = NO PERIOD
74Ø5  72ØØ            CLA
74Ø6  1Ø57            TAD CHAR
74Ø7  1312            TAD PER
741Ø  764Ø            SZA CLA
7411  522Ø            JMP FIGO1
7412  1313            TAD PRSW                /PERIOD FOUND
7413  765Ø            SNA CLA                 /SECOND PERIOD
7414  5222            JMP FIGO2               /YES,TERMINATE
7415  3721            DCA I DPN               /NO - SET NUMBER OF DIGITS TO Ø
7416  3313            DCA PRSW                /SET PERIOD SWITCH TO Ø
7417  5717            JMP I DPCSPT            /CONVERT REST OF STRING

742Ø  1313  FIGO1,    TAD PRSW                /PERIOD READ IN PREVIOUSLY?
7421  765Ø            SNA CLA
7422  1721  FIGO2,    TAD I DPN               /YES:-NUMBER OF DIGITS IN SEXP
7423  7Ø41            CMA IAC                 /NO
7424  3314            DCA SEXP
7425  472Ø            JMS I MSGNPT            /TEST SIGN
7426  1311  FIGO3,    TAD C27
7427  3Ø44            DCA EXPONT
743Ø  44Ø7            JMS I FPNT              /NORMALIZE F.P. NUMBER
7431  7ØØØ            FNOR
7432  6Ø52            FPUT FPAC1              /SAVE NUMBER
7433  ØØØØ            FEXT
7434  1Ø57            TAD CHAR
7435  131Ø            TAD MINUSE
```

```
7436  764Ø            SZA CLA                /"E" READ IN?
7437  5252            JMP ENDFI              /NO
744Ø  4716            JMS I DPCVPT           /YES - CONVERT DECIMAL EXPONENT
7441  472Ø            JMS I MSGNPT           /TEST SIGN
7442  1Ø45            TAD HORDER             /EXPONENT TOO LARGE??
7443  751Ø            SPA
7444  7ØØ1            IAC
7445  764Ø            SZA CLA
7446  5277            JMP EXCESS             /YES
7447  1Ø46            TAD LORDER             /NO:DECIMAL POINT IS
745Ø  1314            TAD SEXP               /C(SEXP)PLACES TO RIGHT
7451  3314            DCA SEXP               /OF LAST DIGIT
            /END OF FLOATING POINT INPUT
            /COMPENSATE FOR DECIMAL EXPONENTS

7452  44Ø7  ENDFI,    JMS I FPNT             /RESTORE MANTISSA
7453  5Ø52            FGET FPAC1
7454  ØØØØ            FEXT
7455  1314            TAD SEXP
7456  745Ø            SNA
7457  56ØØ            JMP I FLINTP
746Ø  77ØØ            SMA CLA
7461  527Ø            JMP FIGO4
7462  44Ø7            JMS I FPNT             /. IS TO THE LEFT:
7463  37Ø7            FMPY I PC.1Ø           /TIMES .1ØØØ
7464  ØØØØ            FEXT
7465  2314            ISZ SEXP
7466  5255            JMP ENDFI+3
7467  56ØØ            JMP I FLINTP

747Ø  44Ø7  FIGO4,    JMS I FPNT             /. IS TO THE RIGHT:
7471  33Ø4            FMPY TEN               /MULTIPLY BY 1Ø
7472  ØØØØ            FEXT
7473  724Ø            CLA CMA
7474  1314            TAD SEXP
7475  3314            DCA SEXP
7476  5255            JMP ENDFI+3
7477  1315  EXCESS,   TAD C3777
75ØØ  3Ø44            DCA EXPONT
75Ø1  1315            TAD C3777
75Ø2  3Ø45            DCA HORDER
75Ø3  56ØØ            JMP I FLINTP
75Ø4  ØØØ4  TEN,      ØØØ4
75Ø5  24ØØ            24ØØ
75Ø6  ØØØØ            ØØØØ
75Ø7  7372  PC.1Ø,    C.1Ø                   /.1Ø
751Ø  7473  MINUSE,   -3Ø5
7511  ØØ27  C27,      ØØ27
7512  7522  PER,      -256
7513  ØØØØ  PRSW,     Ø
7514  ØØØØ  SEXP,     Ø                      /CONTAINS DECIMAL EXPONENT
7515  3777  C3777,    3777

7516  7ØØØ  DPCVPT,   DECONV
7517  7Ø17  DPCSPT,   DECON
752Ø  712Ø  MSGNPT,   MSIGN
7521  7Ø66  DPN,      DNUMBR
            /OUTPUT THE EXPONENT

7522  ØØØØ  FEXC,     Ø
7523  73ØØ            CLA CLL
```

```
7524  1Ø44                 TAD EXPONT
7525  751Ø                 SPA
7526  7Ø61                 CMA IAC CML
7527  3Ø44                 DCA EXPONT
753Ø  1366                 TAD C253
7531  743Ø                 SZL
7532  1367                 TAD C255
7533  4774                 JMS I DGPT
7534  3Ø45                 DCA HORDER
7535  1Ø44                 TAD EXPONT
7536  2Ø45                 ISZ HORDER
7537  137Ø                 TAD M144
754Ø  75ØØ                 SMA
7541  5336                 JMP .-3
7542  1371                 TAD C144
7543  3Ø44                 DCA EXPONT
7544  7Ø4Ø                 CMA
7545  1Ø45                 TAD HORDER
7546  744Ø                 SZA
7547  4774                 JMS I DGPT
755Ø  3Ø45                 DCA HORDER
7551  1Ø44                 TAD EXPONT
7552  2Ø45                 ISZ HORDER
7553  1372                 TAD M12
7554  75ØØ                 SMA
7555  5352                 JMP .-3
7556  1373                 TAD C12
7557  3Ø46                 DCA LORDER
756Ø  724Ø                 CLA CMA
7561  1Ø45                 TAD HORDER
7562  4774                 JMS I DGPT
7563  1Ø46                 TAD LORDER
7564  4774                 JMS I DGPT
7565  5722                 JMP I FEXC

7566  7773  C253,          Ø253-26Ø
7567  ØØØ2  C255,          255-253
757Ø  7634  M144,          7634
7571  Ø144  C144,          Ø144
7572  7766  M12,           7766
7573  ØØ12  C12,           ØØ12
7574  7352  DGPT,          OUTDG
```

```
BEXP     7324
CARRTN   7341
CHAR     ØØ57
CHE      7343
C.1Ø     7372
C12      7573
C144     7571
C253     7566
C255     7567
C26Ø     7356
C27      7511
C3777    7515
DECON    7Ø17
DECONV   7ØØØ
DGPT     7574
DIGIT    7Ø63
DIVTWO   7357
DNUMBR   7Ø66
DPCSPT   7517
DPCVPT   7516
DPN      7521
DPT      7334
DSWIT    ØØ6Ø
DUBLAD   71Ø3
ENDFI    7452
EXCESS   7477
EXPONT   ØØ44
FEXC     7522
FEXPPT   7337
FGO1     7242
FGO2     7246
FGO3     7263
FGO4     727Ø
FGO5A    7274
FGO6     73Ø1
FGO6A    73Ø4
FGO7     7311
FIGO1    742Ø
FIGO2    7422
FIGO3    7426
FIGO4    747Ø
FLINTP   74ØØ
FLOUTP   72ØØ
FOUR     7326
FOUTCN   7217
FPAC1    ØØ52
FPNT     ØØØ7
```

```
HORDER   ØØ45
HTEMP    7Ø62
INPUT    7142
LFED     6777
LNFEED   7342
LORDER   ØØ46
LTEMP    7Ø61
MASK     7141
MINCR    7171
MINUS    7135
MINUSE   751Ø
MINUS7   7325
MIN9     7137
MRBOUT   717Ø
MSGNPT   752Ø
MSIGN    712Ø
MSNPT    7332
MULT1Ø   7Ø41
MULT2    7Ø67
M1ØPT    7335
M12      7572
M144     757Ø
M2PT     7336
OPUT     6776
OUT      7344
OUTDG    7352
OUTPUT   7166
PC.1Ø    75Ø7
PER      7512
PERIOD   7331
PLUS     7136
PLUS12   714Ø
PRCHAR   6767
PRINT    7165
PRSW     7513
REMAIN   7Ø64
RESTRT   7167
SEXP     7514
SIGN     7Ø65
SMINUS   733Ø
SNPT     7333
SPLUS    7327
SWIT1    ØØ55
SWIT2    ØØ56
TEN      75Ø4
TENPT    734Ø
```

```
                    EXTENDED FUNCTIONS
          /FLOATING POINT PACKAGE

          GETSGN=TAD 45
          CLFPNT=JMS I 7

          /FLOATING POINT EXPONENTIAL

          *5ØØØ
5ØØØ ØØØØ FEXP,    Ø
5ØØ1 1Ø45          GETSGN
5ØØ2 771Ø          SPA CLA
5ØØ3 431Ø          JMS FNEG
5ØØ4 33Ø4          DCA SIGN             /C(SIGN)=-1 IF X<Ø
5ØØ5 44Ø7          CLFPNT
5ØØ6 3273          FMPY LG2E
5ØØ7 6315          FPUT X
5Ø1Ø ØØØØ          FEXT
5Ø11 47Ø6          JMS I FIXIT
5Ø12 1Ø45          GETSGN
5Ø13 33Ø5          DCA FLAG2
5Ø14 47Ø7          JMS I FLOAT
5Ø15 44Ø7          CLFPNT
5Ø16 632Ø          FPUT XSQR
5Ø17 5315          FGET X
5Ø2Ø 232Ø          FSUB XSQR
5Ø21 6315          FPUT X
5Ø22 3315          FMPY X
5Ø23 632Ø          FPUT XSQR
5Ø24 127Ø          FADD D
5Ø25 6323          FPUT TEMP
5Ø26 5265          FGET C
5Ø27 4323          FDIV TEMP
5Ø3Ø 2315          FSUB X
5Ø31 1257          FADD A
5Ø32 6323          FPUT TEMP
5Ø33 5262          FGET B
5Ø34 332Ø          FMPY XSQR
5Ø35 1323          FADD TEMP
5Ø36 6323          FPUT TEMP
5Ø37 5315          FGET X
5Ø4Ø 4323          FDIV TEMP
5Ø41 33Ø1          FMPY TWO
5Ø42 1276          FADD ONE
5Ø43 ØØØØ          FEXT
5Ø44 13Ø5          TAD FLAG2
5Ø45 1Ø44          TAD 44
5Ø46 3Ø44          DCA 44
5Ø47 23Ø4          ISZ SIGN
5Ø5Ø 56ØØ          JMP I FEXP
5Ø51 44Ø7          CLFPNT
5Ø52 6315          FPUT X
5Ø53 5276          FGET ONE
5Ø54 4315          FDIV X
5Ø55 ØØØØ          FEXT
5Ø56 56ØØ          JMP I FEXP
          /CONSTANTS FOR FEXP

5Ø57 ØØØ4 A,       ØØØ4
5Ø6Ø 2372          2372
5Ø61 14Ø2          14Ø2
```

```
5Ø62  7774  B,         7774
5Ø63  2157             2157
5Ø64  5157             5157
5Ø65  ØØ12  C,         ØØ12
5Ø66  5454             5454
5Ø67  Ø343             Ø343
5Ø7Ø  ØØØ7  D,         ØØØ7
5Ø71  2566             2566
5Ø72  5341             5341
5Ø73  ØØØ1  LG2E,      ØØØ1
5Ø74  27Ø5             27Ø5
5Ø75  2435             2435
5Ø76  ØØØ1  ONE,       ØØØ1
5Ø77  2ØØØ             2ØØØ
51ØØ  ØØØØ             ØØØØ
51Ø1  ØØØ2  TWO,       ØØØ2
51Ø2  2ØØØ             2ØØØ
51Ø3  ØØØØ             ØØØØ
51Ø4  ØØØØ  SIGN,      Ø
51Ø5  ØØØØ  FLAG2,     Ø
51Ø6  4757  FIXIT,     FIX
51Ø7  5563  FLOAT,     FLOA
            /NEGATION SUBROUTINE
511Ø  ØØØØ  FNEG,      Ø
5111  4714             JMS I ACMINS          /CALL SUBROUTINE IN
5112  724Ø             CLA CMA               /INTERPRETER
5113  571Ø             JMP I FNEG
5114  6ØØØ  ACMINS,    6ØØØ                  /POINTER TO INTERPRETER

            /TEMPORARY STORAGE
5115  ØØØØ  X,         Ø
5116  ØØØØ             Ø
5117  ØØØØ             Ø
512Ø  ØØØØ  XSQR,      Ø
5121  ØØØØ             Ø
5122  ØØØØ             Ø
5123  ØØØØ  TEMP,      Ø
5124  ØØØØ             Ø
5125  ØØØØ             Ø
5126  ØØØØ  ZER,       Ø
5127  ØØØØ             Ø
513Ø  ØØØØ             Ø
            /MAIN ALGORITHM FOR ARCTANGENT
5131  44Ø7  ARCALG,    CLFPNT
5132  5315             FGET X
5133  3315             FMPY X
5134  632Ø             FPUT XSQR
5135  3373             FMPY BET2
5136  137Ø             FADD BET1
5137  332Ø             FMPY XSQR
514Ø  1365             FADD BETZ
5141  6323             FPUT TEMP
5142  5362             FGET ALF2
5143  332Ø             FMPY XSQR
5144  1357             FADD ALF1
5145  332Ø             FMPY XSQR
5146  1354             FADD ALFZ
5147  3315             FMPY X
515Ø  4323             FDIV TEMP
5151  ØØØØ             FEXT
5152  5753             JMP I .+1
5153  5226             ARCRTN
```

```
                /CONSTANTS - FLOATING ARC TANGENT
5154  ØØØØ  ALFZ,     ØØØØ
5155  2437            2437
5156  1643            1643
5157  7777  ALF1,     7777
516Ø  33Ø4            33Ø4
5161  4434            4434
5162  7773  ALF2,     7773
5163  33Ø6            33Ø6
5164  5454            5454
5165  ØØØØ  BETZ,     ØØØØ
5166  2437            2437
5167  1646            1646
517Ø  ØØØØ  BET1,     ØØØØ
5171  2427            2427
5172  2323            2323
5173  7775  BET2,     7775
5174  3427            3427
5175  7Ø52            7Ø52
            /FLOATING POINT ARC TANGENT
            *52ØØ
52ØØ  ØØØØ  ARTN,     Ø
52Ø1  1Ø45            GETSGN
52Ø2  771Ø            SPA CLA
52Ø3  4642            JMS I NEGIT
52Ø4  3243            DCA FLAG1
52Ø5  44Ø7            CLFPNT
52Ø6  6644            FPUT I X1
52Ø7  5644            FGET I X1
521Ø  2645            FSUB I CON1
5211  ØØØØ            FEXT
5212  1Ø45            GETSGN
5213  771Ø            SPA CLA
5214  5223            JMP GO
5215  44Ø7            CLFPNT
5216  5645            FGET I CON1
5217  4644            FDIV I X1
522Ø  6644            FPUT I X1
5221  ØØØØ            FEXT
5222  724Ø            CLA CMA
5223  3251  GO,       DCA FLOG           /FLAG
5224  5625            JMP I .+1          /CALL ALGORITHM
5225  5131            ARCALG
5226  2251  ARCRTN,   ISZ FLOG           /RETURN HERE
5227  5235            JMP EXIT
523Ø  44Ø7            CLFPNT
5231  6644            FPUT I X1
5232  5246            FGET PIOT
5233  2644            FSUB I X1
5234  ØØØØ            FEXT
5235  2243  EXIT,     ISZ FLAG1
5236  56ØØ            JMP I ARTN
5237  4642            JMS I NEGIT
524Ø  72ØØ            CLA
5241  56ØØ            JMP I ARTN
            /CONSTANTS FOR ARCTANGENT

5242  511Ø  NEGIT,    FNEG
5243  ØØØØ  FLAG1,    Ø
5244  5115  X1,       X
5245  5Ø76  CON1,     ONE
```

```
5246  0001  PIOT,      0001
5247  3110             3110
5250  3755             3755
            /FLOATING LOGARITHM

5251  0000  FLOG,      0
5252  1045             GETSGN
5253  7440             SZA
5254  5261             JMP OK
5255  4407             CLFPNT                /INDICATE ERROR
5256  4775             FDIV I ZERO           /DIVIDE BY ZERO
5257  0000             FEXT
5260  5651             JMP I FLOG
5261  7710  OK,        SPA CLA
5262  4642             JMS I NEGIT
5263  4407             CLFPNT
5264  6774             FPUT I TEM
5265  2645             FSUB I CON1
5266  0000             FEXT
5267  1045             GETSGN
5270  7450             SNA
5271  5350             JMP ZERGO
5272  7710             SPA CLA
5273  5341             JMP INVERT
5274  3243  START,     DCA FLAG1
5275  7040             CMA
5276  1774             TAD I TEM
5277  3045             DCA 45
5300  4777             JMS I FLOATP
5301  4407             CLFPNT
5302  3776             FMPY I LOG2
5303  6644             FPUT I X1
5304  0000             FEXT
5305  7001             IAC
5306  3774             DCA I TEM
5307  4407             CLFPNT
5310  5774             FGET I TEM
5311  2645             FSUB I CON1
5312  6774             FPUT I TEM
5313  3773             FMPY I L8
5314  1772             FADD I L7
5315  3774             FMPY I TEM
5316  1771             FADD I L6
5317  3774             FMPY I TEM
5320  1770             FADD I L5
5321  3774             FMPY I TEM
5322  1365             FADD L4
5323  3774             FMPY I TEM
5324  1362             FADD L3
5325  3774             FMPY I TEM
5326  1357             FADD L2
5327  3774             FMPY I TEM
5330  1354             FADD L1
5331  3774             FMPY I TEM
5332  1644             FADD I X1
5333  0000             FEXT
5334  2243             ISZ FLAG1
5335  5651             JMP I FLOG
5336  4642             JMS I NEGIT
5337  7200             CLA
5340  5651             JMP I FLOG
```

```
5341  4407  INVERT,   CLFPNT
5342  5645            FGET I CON1
5343  4774            FDIV I TEM
5344  6774            FPUT I TEM
5345  ØØØØ            FEXT
5346  724Ø            CLA CMA
5347  5274            JMP START
535Ø  4407  ZERGO,    CLFPNT
5351  5775            FGET I ZERO
5352  ØØØØ            FEXT
5353  5651            JMP I FLOG
5354  ØØØØ  L1,       ØØØØ
5355  3777            3777
5356  7742            7742
5357  7777  L2,       7777
536Ø  4ØØØ            4ØØØ
5361  41ØØ            41ØØ
5362  7777  L3,       7777
5363  2517            2517
5364  Ø3Ø7            Ø3Ø7
5365  7776  L4,       7776
5366  4113            4113
5367  7211            7211
537Ø  5547  L5,       LOG5
5371  5552  L6,       LOG6
5372  5555  L7,       LOG7
5373  556Ø  L8,       LOG8
5374  5123  TEM,      TEMP
5375  5126  ZERO,     ZER
5376  5544  LOG2,     LOGE2
5377  5563  FLOATP,   FLOA
            /FLOATING POINT SINE
            *54ØØ
54ØØ  ØØØØ  FSIN,     Ø
54Ø1  1Ø45            GETSGN              /X>Ø?
54Ø2  774Ø            SMA SZA CLA
54Ø3  521Ø            JMP MOD             /YES
54Ø4  1Ø45            GETSGN
54Ø5  77ØØ            SMA CLA             /NO  X=Ø?
54Ø6  56ØØ            JMP I FSIN          /YES SIN(Ø)=Ø
54Ø7  4741            JMS I NEGT          /NO:SIN(-X)=-SIN(X)
541Ø  3343  MOD,      DCA PNTR            /REDUCE X MODULO 2 PI
5411  44Ø7            CLFPNT
5412  4315            FDIV TWOPI
5413  6724            FPUT I XSQ2
5414  ØØØØ            FEXT
5415  4742            JMS I FIXR
5416  4363            JMS FLOA
5417  44Ø7            CLFPNT
542Ø  6723            FPUT I X2
5421  5724            FGET I XSQ2
5422  2723            FSUB I X2
5423  3315            FMPY TWOPI
5424  6723            FPUT I X2
5425  232Ø            FSUB PI             /X<PI?
5426  ØØØØ            FEXT
5427  1Ø45            GETSGN
543Ø  771Ø            SPA CLA
5431  5241            JMP PCHECK          /YES
5432  44Ø7            CLFPNT              /NO SIN(X-PI)=-SIN(X)
5433  6723            FPUT I X2
5434  ØØØØ            FEXT
```

```
5435  1343             TAD PNTR
5436  7650             SNA CLA
5437  7040             CMA
5440  3343             DCA PNTR
5441  4407   PCHECK,   CLFPNT                  /X<PI/2?
5442  5723             FGET I X2
5443  2714             FSUB I PI2
5444  0000             FEXT
5445  1045             GETSGN
5446  7710             SPA CLA
5447  5255             JMP PALG                /YES
5450  4407             CLFPNT                  /NO
5451  5320             FGET PI                 /SIN(X)=SIN(PI-X)
5452  2723             FSUB I X2
5453  6723             FPUT I X2
5454  0000             FEXT

5455  4407   PALG,     CLFPNT
5456  5723             FGET I X2
5457  4714             FDIV I PI2
5460  6723             FPUT I X2
5461  3723             FMPY I X2
5462  6724             FPUT I XSQ2
5463  5325             FGET C9
5464  3724             FMPY I XSQ2
5465  1330             FADD C7
5466  3724             FMPY I XSQ2
5467  1333             FADD C5
5470  3724             FMPY I XSQ2
5471  1336             FADD C3
5472  3724             FMPY I XSQ2
5473  1714             FADD I PI2
5474  3723             FMPY I X2
5475  0000             FEXT
5476  2343             ISZ PNTR
5477  5600             JMP I FSIN
5500  4741             JMS I NEGT
5501  7200             CLA
5502  5600             JMP I FSIN
             /FLOATING POINT COSINE

5503  0000   FCOS,     0
5504  4407             CLFPNT                  /COS(X)=SIN(PI/2-X)
5505  6723             FPUT I X2
5506  5714             FGET I PI2
5507  2723             FSUB I X2
5510  0000             FEXT
5511  1303             TAD FCOS
5512  3200             DCA FSIN
5513  5201             JMP FSIN+1

             /CONSTANTS AND POINTERS

5514  5246   PI2,      PIOT                    /PI/2
5515  0003   TWOPI,    0003
5516  3110             3110
5517  3755             3755
5520  0002   PI,       0002
5521  3110             3110
5522  3755             3755
5523  5115   X2,       X
5524  5120   XSQ2,     XSQR
```

```
                /SINE CONSTANTS
5525  7764  C9,         7764
5526  2366              2366
5527  5735              5735
553Ø  7771  C7,         7771
5531  5466              5466
5532  6317              6317
5533  7775  C5,         7775
5534  2431              2431
5535  5Ø53              5Ø53
5536  ØØØØ  C3,         ØØØØ
5537  5325              5325
554Ø  Ø42Ø              Ø42Ø
5541  511Ø  NEGT,       FNEG
5542  4757  FIXR,       FIX
5543  ØØØØ  PNTR,       Ø

                /LOGARITHM CONSTANTS

5544  ØØØØ  LOGE2,      ØØØØ
5545  2613              2613
5546  4414              4414
5547  7776  LOG5,       7776
555Ø  2535              2535
5551  33Ø1              33Ø1
5552  7775  LOG6,       7775
5553  4746              4746
5554  Ø771              Ø771
5555  7774  LOG7,       7774
5556  2236              2236
5557  43Ø4              43Ø4
556Ø  7771  LOG8,       7771
5561  4544              4544
5562  1735              1735
            /FLOAT C(45)
5563  ØØØØ  FLOA,       Ø
5564  73ØØ              CLA CLL
5565  3Ø46              DCA 46
5566  1374              TAD C13
5567  3Ø44              DCA 44
557Ø  44Ø7              CLFPNT
5571  7ØØØ              FNOR
5572  ØØØØ              FEXT
5573  5763              JMP I FLOA
5574  ØØ13  C13,        ØØ13
            *4757
            /FIX C(FAC)
4757  ØØØØ  FIX,        Ø
476Ø  1Ø44              TAD 44
4761  754Ø              SMA SZA
4762  5365              JMP .+3
4763  72ØØ              CLA
4764  5375              JMP FIXEND
4765  1377              TAD M13
4766  3Ø44              DCA 44
4767  1Ø44  LOOP,       TAD 44
477Ø  77ØØ              SMA CLA
4771  5757              JMP I FIX
4772  4774              JMS I .+2
4773  5367              JMP LOOP
4774  62ØØ              62ØØ                   /DIV1 IN INTERPRETER
```

```
4775  3Ø45  FIXEND,   DCA 45
4776  5757            JMP I FIX
4777  7765  M13,      -13
```

```
A         5Ø57                          LOG2      5376
ACMINS    5114                          LOG5      5547
ALFZ      5154                          LOG6      5552
ALF1      5157                          LOG7      5555
ALF2      5162                          LOG8      556Ø
ARCALG    5131                          LOOP      4767
ARCRTN    5226                          L1        5354
ARTN      52ØØ                          L2        5357
B         5Ø62                          L3        5362
BETZ      5165                          L4        5365
BET1      517Ø                          L5        537Ø
BET2      5173                          L6        5371
C         5Ø65                          L7        5372
CLFPNT    44Ø7                          L8        5373
CON1      5245                          MOD       541Ø
C13       5574                          M13       4777
C3        5536                          NEGIT     5242
C5        5533                          NEGT      5541
C7        553Ø                          OK        5261
C9        5525                          ONE       5Ø76
D         5Ø7Ø                          PALG      5455
EXIT      5235                          PCHECK    5441
FCOS      55Ø3                          PI        552Ø
FEXP      5ØØØ                          PIOT      5246
FIX       4757                          PI2       5514
FIXEND    4775                          PNTR      5543
FIXIT     51Ø6                          SIGN      51Ø4
FIXR      5542                          START     5274
FLAG1     5243                          TEM       5374
FLAG2     51Ø5                          TEMP      5123
FLOA      5563                          TWO       51Ø1
FLOAT     51Ø7                          TWOPI     5515
FLOATP    5377                          X         5115
FLOG      5251                          XSQR      512Ø
FNEG      511Ø                          XSQ2      5524
FSIN      54ØØ                          X1        5244
GETSGN    1Ø45                          X2        5523
GO        5223                          ZER       5126
INVERT    5341                          ZERGO     535Ø
LG2E      5Ø73                          ZERO      5375
LOGE2     5544
```

```
                /FLOATING OUTPUT PROGRAM
                /IF(62)=0, THEN OUTPUT IN E FORMAT
                /ELSE, C(62)=NUMBER OF DIGITS
                /C(AC)=NUMBER OF PLACES TO RIGHT OF .
                /IF C(AC)=0, THEN DON'T OUTPUT POINT
                /SIGN AND . NOT COUNTED IN OUTPUT
                /CONTENTS OF 15 LOST DURING OPERATION

                *7201
7201  4777                JMS I 7377 /ALTERATIONS FLOATING OUTPUT
7202  4217                JMS 7217
7203  1324                TAD 7324
7204  4776                JMS I 7376

                *7207
7207  7200                CLA


                *7227
7227  3415                DCA I 15
7230  7000                NOP
7231  7000                NOP
7232  7000                NOP

                *7301
7301  3415                DCA I 15

                *7305
7305  3415                DCA I 15

                *7376
7376  5412                TGO
7377  5400                EDIT
                *5400
5400  0000      EDIT,     0
5401  3352                DCA SAC
5402  1062                TAD 62
5403  7041                CMA IAC
5404  3347                DCA COUNT1
5405  1357                TAD M8
5406  3350                DCA COUNT2
5407  1363                TAD SAD1
5410  3015                DCA 15
5411  5600                JMP I EDIT
5412  0000      TGO,      0
5413  3044                DCA 44
5414  1363                TAD SAD1
5415  3015                DCA 15
5416  1415                TAD I 15
5417  4761                JMS I OUT1
5420  1062                TAD 62
5421  7650                SNA CLA
5422  5327                JMP EFORM
5423  2212                ISZ TGO
5424  2212                ISZ TGO
5425  1044      TRYAGN,   TAD 44
5426  7510                SPA
5427  5300                JMP MINS
5430  1352                TAD SAC
5431  7041                CMA IAC
5432  1062                TAD 62
```

```
5433  7510           SPA
5434  5266           JMP ERR
5435  7450           SNA
5436  5245           JMP G01
5437  7041           CMA IAC
5440  3351           DCA CNTR
5441  1353           TAD SPCE
5442  4321           JMS OUT
5443  2351           ISZ CNTR
5444  5241           JMP .-3

5445  1044  G01,     TAD 44
5446  7041           CMA IAC
5447  7450           SNA
5450  5256           JMP G02
5451  3351           DCA CNTR
5452  4337           JMS GET
5453  4321           JMS OUT
5454  2351           ISZ CNTR
5455  5252           JMP .-3

5456  1352  G02,     TAD SAC
5457  7650           SNA CLA
5460  5263           JMP .+3
5461  1354           TAD PERIOD
5462  4761           JMS I OUT1
5463  4337           JMS GET
5464  4321           JMS OUT
5465  5263           JMP .-2
5466  1352  ERR,     TAD SAC
5467  7700           SMA CLA
5470  5274           JMP ERGO
5471  1356           TAD CHX
5472  4321           JMS OUT
5473  5271           JMP .-2

5474  7240  ERGO,    CLA CMA
5475  1352           TAD SAC
5476  3352           DCA SAC
5477  5225           JMP TRYAGN

5500  7200  MINS,    CLA
5501  1062           TAD 62
5502  7041           CMA IAC
5503  1352           TAD SAC
5504  7450           SNA
5505  5313           JMP G03
5506  3351           DCA CNTR
5507  1353           TAD SPCE
5510  4321           JMS OUT
5511  2351           ISZ CNTR
5512  5307           JMP .-3

5513  1354  G03,     TAD PERIOD
5514  4761           JMS I OUT1
5515  4321           JMS OUT
5516  2044           ISZ 44
5517  5315           JMP .-2
5520  5263           JMP G02+5
5521  0000  OUT,     0
5522  4762           JMS I OUT2
```

```
5523  2347                 ISZ COUNT1
5524  5721                 JMP I OUT
5525  1355                 TAD CHE
5526  5612                 JMP I TGO

5527  4762  EFORM,         JMS I OUT2
5530  1354                 TAD PERIOD
5531  4761                 JMS I OUT1
5532  1360                 TAD M7
5533  3347                 DCA COUNT1
5534  4337                 JMS GET
5535  4321                 JMS OUT
5536  5334                 JMP .-2

5537  0000  GET,           0
5540  2350                 ISZ COUNT2
5541  5345                 JMP .+4
5542  7240                 CLA CMA
5543  3350                 DCA COUNT2
5544  5737                 JMP I GET
5545  1415                 TAD I 15
5546  5737                 JMP I GET
5547  0000  COUNT1,        0
5550  0000  COUNT2,        0
5551  0000  CNTR,          0
5552  0000  SAC,           0
5553  7760  SPCE,          240-260
5554  0256  PERIOD,        256
5555  0305  CHE,           305
5556  0050  CHX,           330-260
5557  7770  M8,            -10
5560  7771  M7,            -7
5561  7344  OUT1,          7344
5562  7352  OUT2,          7352
5563  5563  SAD1,          BUFFER-1
            BUFFER,
BUFFER   5564
CHE      5555
CHX      5556
CNTR     5551
COUNT1   5547
COUNT2   5550
EDIT     5400
EFORM    5527
ERGO     5474
ERR      5466
GET      5537
GO1      5445
GO2      5456
GO3      5513
MINS     5500
M7       5560
M8       5557
OUT      5521
OUT1     5561
OUT2     5562
PERIOD   5554
SAC      5552
SAD1     5563
SPCE     5553
TGO      5412
TRYAGN   5425
```

digital
EQUIPMENT
CORPORATION
MAYNARD, MASSACHUSETTS

PRINTED IN U.S.A.