FPMP-11
USER'S MANUAL

FPMP-11
USER'S MANUAL

For additional copies, order No. DEC- 11-NFPMA-C-D
from Software Distribution Center, Digital Equipment
Corporation, Maynard, Mass.

digital equipment corporation · maynard. massachusetts

First Printing
September, 1972
Revised, April, 1973
Revised, July, 1973

Your attention is invited to the last two pages of this document. The "How to Obtain Software Information" page explains how to keep up-to-date with DEC's software. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and considered when documenting subsequent manuals.

Changes from the previous version are indicated by a bar (|) in the margin.

The material in this document is for information purposes and is subject to change without notice.

Teletype is a registered trademark of the Teletype Corporation.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

CDP	Digital	LAB-8/e	RAD-8
Computer Lab	DNC	OMNIBUS	RSTS
Comtex	Flip Chip	OS/8	RSX
DEC	IDAC	PDP	RTM
DECtape	Indac	PHA	SABR
Dibol	KA10	PS/8	Typeset 8
		Quickpoint	Unibus

PREFACE

This manual assumes the reader is familiar with PDP-11 assembly language programming and with floating point operations in general.

For background in the papertape system, refer to the PDP-11 Paper Tape Software Programming Handbook (DEC-11-XPTSA-A-D).

TABLE OF CONTENTS

CHAPTER 1
FPMP-11 OVERVIEW

1.1 INTRODUCTION

The Floating-Point Math Package, FPMP-11, is designed to bring the 2/4 word floating point format of the FORTRAN environment to the paper tape software system of the PDP-11. The numerical routines in FPMP-11 are the same as those of the DOS-11 Fortran Object Time System (OTS). TRAP and error handlers have been included to aid in interfacing with the FORTRAN routines.

FPMP-11 provides an easy means of performing basic arithmetic operations such as add, subtract, multiply, divide and compare. It also provides transcendental functions (SIN, COS, etc.), type conversions (integer to floating point, 2 word to 4 word, etc.) and ASCII conversions (ASCII to 2 word floating point, etc.).

Floating-point notation is particularly useful for computations involving numerous multiply and divide operations where operand magnitudes may vary widely. FPMP-11 stores very large and very small numbers by saving only the significant digits and computing an exponent to account for leading and trailing zeros.

To conserve core space in a small system, FPMP-11 can be tailored to include only those routines needed to run a particular user program.

1.2 HARDWARE REQUIREMENTS

The FPMP-11 package is designed for use on any PDP-11 with at least 8K of core, and can be easily reassembled to take advantage of the 11/20 EAE, 11/45 EIS, or 11/45 FPU (refer to section 3.5 for detailed instructions).

1.3 SOFTWARE REQUIREMENTS

LINK-11S (or the DOS LINK-11 linker) is used to link a user program with an FPMP-11 object module to create a load module. PAL-11S (or MACRO-11 under DOS-11) is used whenever the FPMP-11 package is reassembled.

1.4 FLOATING-POINT NOTATION

A floating-point number may be written as a mantissa, which consists of the floating-point number with its decimal point shifted a given

number of places in either direction, and an exponent which indicates the number of places that the decimal point was shifted and the direction of the shift. A negative exponent corresponds to a shift to the right, while a positive exponent corresponds to a shift to the left. Thus, the mantissa multiplied by the base (radix) of the number system in use, raised to a power as supplied in the exponent, gives the value of the number in fixed-point notation. For example, the decimal number 12 in fixed-point notation can be represented as

12 or 12.0

In floating-point notation with a base of 10, the number might appear as

$.12 \times 10^{2}$

where the mantissa is .12 and the exponent, 2.

A fraction, such as twelve ten-thousandths, is represented as

.0012

in fixed-point notation and in floating-point notation as

$.12 \times 10^{-2}$

The minus sign before the exponent indicates that the significant digits of the mantissa are to be shifted right from the decimal point.

In FPMP-11 all numbers are manipulated and stored in binary notation. With a radix of 2, the decimal number 12 is represented as

1100

and in floating-point format as

$.1100 \times 2^{4}$

Multiplication and division are accomplished by shift operations: each one-place shift to the left represents multiplication by two; each equivalent shift to the right represents division by two.

A floating-point number may be represented in an infinite variety of ways, since the decimal point may be shifted any number of places in either direction. If the decimal point is shifted until it appears immediately to the left of the most significant digit, the number is said to be normalized. The mantissa of a normalized floating-point number may be stored as an integer, since the decimal point is understood to appear to the left of the most significant digit. In

computing a mantissa from decimal input, FPMP-11 uses the convention

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

to normalize the input value. Note that when |MANTISSA| is stored as a binary fraction in normalized form, the left most (high order) bit is always a 1. The only exception to the normalization rule is the floating-point zero (either single or double precision) which has a mantissa and exponent both equal to zero.

1.5 FLOATING-POINT NUMBER STORAGE

FPMP-11 floating-point numbers are stored as two 16-bit PDP-11 words (single precision) or four 16-bit PDP-11 words (double precision). The sign of the number is bit 15 of the first word. (0 indicates positive, 1 indicates negative). The binary exponent is stored in bits 14-7 of the first word. The exponent is stored in excess 128 (200_8) code. The value of the exponent is obtained by subtracting 200_8 from bits 14-7 of the first word.

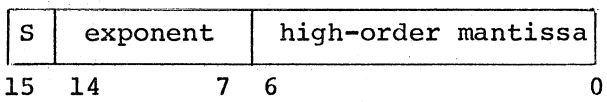
NOTE

The single and double precision formats shown below are limited to normalized numbers. The high-order bit of the mantissa (which is always 1) is omitted from its implied position (bit 7 of WORD n) in order to allow one more bit in the exponent field.

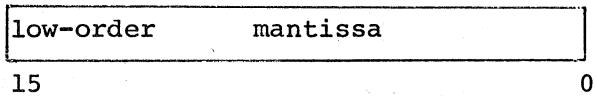
1.5.1 Single Precision

The mantissa and exponent are stored as follows:

WORD n



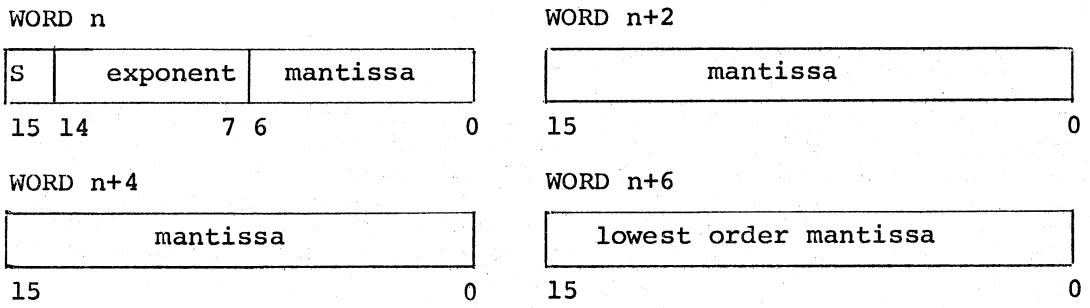
WORD n+2



The first word (lowest core address) contains the sign of the mantissa, the exponent excess 128_{10} and the high-order mantissa (absolute value). The second word is the low-order mantissa (absolute value continued).

1.5.2 Double Precision

Double precision format is identical to single precision format except that it has two additional words (WORD n+4 and WORD n+6) of low-order mantissa.



The list below provides examples of numbers in decimal form, binary floating point notation and single precision internal form.

Decimal Value	Binary Floating Point	Internal Form Single Precision (octal)
1	0.1×2^{1}	040200 000000
2	0.1×2^{2}	040400 000000
5	0.101×2^{3}	040640 000000
10	0.101×2^{4}	041040 000000
$\sqrt{2}$	$0.10110101\ldots \times 2^{1}$	040265 002363
-1	-0.1×2^{1}	140200 000000
0.5	0.1×2^{0}	040000 000000
0.25	0.1×2^{-1}	037600 000000
0.75	0.11×2^{0}	040100 000000
-0.25	-0.1×2^{-1}	137600 000000

CHAPTER 2

DESCRIPTION OF PACKAGE

As distributed the FPMP-11 package contains three sub-packages: the object tape of the single precision functions, the object tape of the double precision functions and the source tapes.

2.1 SINGLE PRECISION PACKAGE

The single precision package is an object module tape (DEC-11-NFPMA-A-PR1) which includes the FPMP-11 TRAP and error handlers and the following OTS routines for two-word floating point operation:

$ADR	Add routine
$SBR	Subtract routine
$MLR	Multiply routine
$DVR	Divide routine
$CMR	Compare routine
SIN	Sine routine
COS	Cosine routine
AINT	Truncation routine
ATAN	Arctangent routine
ATAN2	Arctangent routine with two arguments
SQRT	Square root routine
TANH	Hyperbolic tangent routine
EXP	Exponential routine
ALOG	Natural logarithm routine
ALOG10	Base-10 logarithm routine

and the ASCII input/output conversion routines. There are also routines to load and store the FLAC (FLoating-point ACcumulator) which may be called through the TRAP handler. (Refer to section 3.1.)

The functions are identical to their FORTRAN counterparts and are described in more detail in Appendix D.

2.2 DOUBLE PRECISION PACKAGE

The double precision package is an object module tape (DEC-11-NFPMA-A-PR2) which includes the TRAP and error handlers and the following OTS routines for four-word floating point operations:

$ADD	Add routine
$SBD	Subtract routine
$MLD	Multiply routine
$DVD	Divide routine
$CMD	Compare routine
DSIN	Sine routine
DCOS	Cosine routine
DATAN	Arctangent routine
DATAN2	Arctangent routine with two arguments
DSQRT	Square root routine
DEXP	Exponential routine
DLOG	Natural logarithm routine
DLOG10	Base-10 logarithm routine

and ASCII input/output conversions routines. There are also routines to load and store the FLAC which may be called through the TRAP handler (refer to section 3.1).

Appendix D contains a more detailed description of the functions.

2.3 SOURCE TAPES

The source tapes (DEC-11-NFPMA-A-PA1-PA6) contain the source code for the TRAP handler, the error handler and all the OTS routines described in Appendix D. Conditional assembly instructions are included in the source code to aid in the construction of specially tailored packages. For example, an object tape of only the TRAP and error handlers and the arithmetic functions, add, subtract, multiply and divide can be easily created. Such a package can result in great savings of core when the other functions are not required. (Refer to section 3.5 for information on creating special packages.)

2.4 CONVERSION ROUTINES

The subroutines included in FPMP-11 to perform conversions to and from ASCII strings are those used by FORTRAN to perform Input/Output. The FPMP-11 routines do not perform any actual I/O, but simply convert strings of ASCII characters in memory to the internal form of floating-point numbers or integers used by other FPMP-11 subroutines and convert numbers in internal form to ASCII strings.

In order to effectively use the ASCII conversion routines of FPMP-11, the meaning of the various parameters which must be passed to these routines and the various data formats involved must be understood. Table 2-1 contains the various data formats processed by FPMP-11 conversion routines:

TABLE 2-1

DATA FORMATS

CODE	INTERNAL FORM	EXTERNAL INPUT FORM	EXTERNAL OUTPUT FORM
D	Double Precision	Decimal number with or without a decimal point or exponent field.	Decimal number with a D exponent field and a decimal point.
E	Single Precision	Decimal number with or without a decimal point or exponent field	Decimal number with an E exponent field and a decimal point.
F	Single Precision	Decimal number with or without a decimal point or exponent field	Decimal number with a decimal point
G	Single Precision	Decimal number with or without a decimal point or exponent field.	Decimal number with a decimal point and with or without an E exponent field (see table 2-2)
I	Integer	Decimal number without a decimal point or exponent	Decimal number without a decimal point or exponent
O	Integer	Octal number	Octal number

The following FPMP-11 routines perform the above conversions:

$DCI	D conversion for input
$DCO	D conversion for output
$RCI	E,F, and G conversion for input
$ECO	E conversion for output
$FCO	F conversion for output
$GCO	G conversion for output
$ICI	I conversion for input

$ICO	I conversion for output
$OCI	O conversion for input
$OCO	O conversion for output

Each of these routines requires one or more of the following parameters:

w The width of the ASCII field in characters. The field width w of all output conversions should always be large enough to include spaces for the decimal point, sign, and exponent. In all such conversions, if w is not large enough to accomodate the converted number, asterisks are placed in the ASCII field.

d The decimal position:

a) on input, the decimal point is assumed to be d digits from the right hand end of the ASCII field, if no explicit decimal point is found.

b) on output, d digits appear to the right of the decimal point.

p the scale factor:

a) for F type conversion,
(ASCII number)=(internal no.) $*10^{(\text{scale factor})}$

b) for D and E type conversions, the scale factor multiplies the fraction by a power of ten, but the exponent is adjusted, leaving the number unchanged except in form.

c) for G type conversions, the scale factor is not used unless the magnitude of the number is such that E format is used.

d) In all input operations, the scale factor is not used if there is an exponent in the external field.

NOTE

Input conversion routines handle all blanks as zeros. For example, 3.0E2␣ in a six character field would be considered to be 3.0E20.

TABLE 2-2

G-TYPE OUTPUT CONVERSIONS

Routine $GCO is called with parameters p=P, w=W, and d=D (where P, W, D are integer constants):

Magnitude of data	Resulting Conversion
$0.1 \leq M < 1$	F-type with p=0,w=W-4 and d=D
$1 \leq M < 10$	F-type with p=0,w=W-4 and d=D-1.
$10^{D-2} \leq M < 10^{D-1}$	F-type with p=0,w=W-4 and d=1.
$10^{D-1} \leq M < 10^{D}$	F-type with p=0,w=W-4 and d=0.
All others	E-type with p=P,w=W, and d=D.

Examples:
The following internal numbers are shown converted according to various format parameters (b=blank):

(A) ONE-WORD INTEGERS:

INTERNAL NUMBER (Decimal)	I(w=5)	I(w=7)	O(w=10)
5	bbbb5	bbbbbb5	bbbbbbbbb5
10	bbb10	bbbbb10	bbbbbbbb12
-23	bb-23	bbbb-23	bbbbbbb-27
0	bbbb0	bbbbbb0	bbbbbbbbb0
123,456	*****	b123456	bbbb361100

(B) TWO-WORD FLOATING POINT:

	-----(P=0)-----		
INTERNAL NO.	E(w=10,d=2)	F(w=10,d=2)	G(w=10,d=2)
0	bb0.00E 00	bbbbbb0.00	bb0.00bbbb
1	bb0.10E 01	bbbbbb1.00	bb1.00bbbb
-1	b-0.10E 01	bbbbb-1.00	b-1.00bbbb
0.1	bb0.10E 00	bbbbbb0.10	bb0.10bbbb
555	bb0.55E 03	bbbb555.00	bb0.55E 03
0.001	bb0.10E-02	bbbbbb0.00	bb0.10E-02
	-----(P=1)-----		
0	bb0.00E-01	bbbbbb0.00	bb0.00bbbb
1	bb1.00E 00	bbbbb10.00	bb1.00bbbb
0.1	bb1.00E-01	bbbbbb1.00	bb0.10bbbb

(C) FOUR-WORD FLOATING POINT:

D-type conversion is the only one available for 4-word floating-point numbers. It is similar to E format except that the exponent part prints with a D instead of an E.

CHAPTER 3

USING FPMP-11

The user program can access the FPMP-11 routines by TRAP instruction and/or direct call of the routine. (For information on writing a user program, refer to the Papertape Software Programming Handbook.) The TRAP handler saves and restores the contents of the PDP-11 general registers. The OTS routines normally do not. All FPMP-11 entry points used by the program must be declared with a .GLOBL assembler directive in the user program. (The entry points are listed in Appendix D.) To include user floating point error routines, initialize the global location $ERVEC as described in section 3.4.

3.1 USING THE TRAP HANDLER WITH FPMP-11

In order to simplify use of the various OTS routines, a TRAP handler is included in the FPMP-11 package. If TRAP calls are being used, the user program must initialize the TRAP vector at location 34$_8$. The TRAP vector can be initialized by putting the following code in the user program.

```
.
.
.
.GLOBL TRAPH
MOV #TRAPH,@#34      ;address of TRAP handler
MOV #340,@#36        ;set priority of operation
```

The TRAP handler, TRAPH, uses software to simulate a floating-point accumulator (FLAC). The FLAC is a pseudo-register which is the implicit destination address of every trapped operation. Operations can be performed on the FLAC by issuing coded TRAP instructions in the user program. In addition to being used with the OTS functions, items can be loaded into and stored from the FLAC.

The FLAC is maintained by the TRAP handler in double precision format; however, it is important to note that single precision operations (e.g. $ADR or SQRT) destroy the contents of the two lowest order words of the FLAC. In particular, these two words are not set to zero. This means that a single precision function can operate on the FLAC while it contains either a single or double precision number, but the result will be single precision and should not be operated on by the double precision routines. A number can be explicitly converted between the single and double precision formats by the FPMP-11 routines $RD and $DR which convert single to double and double to single respectively. These routines, $RD and $DR, can not be called via the TRAP handler.

Because it contains the floating accumulator, the TRAP handler of FPMP-11 is not re-entrant. For this reason, care must be exercised if the TRAP handler is to be called both in a main program and in an interrupt-driven subroutine. To call the TRAP handler to perform floating point operations within an interrupt-driven subroutine, the contents of the FLAC should be pushed onto the processor stack before any other TRAP calls are executed. The FLAC can be pushed onto the stack by executing the instruction "TRAP 73". After all TRAP calls have been completed by the interrupt-driven subroutine, and before returning from the interrupt, the FLAC must be restored from the stack (it must be at the top of the stack) by executing the instruction "TRAP 71". If the double precision routines are being used, the traps are TRAP 74 and TRAP 72 respectively.

Addressing Modes Available in TRAP Calls:

3.1.1 Stack Mode

The operand is considered to be on the top of the R6 stack. (R6 is General Register 6) The operand is popped off for use. (exception: STR and STD push the FLAC onto the stack.)

3.1.2 @R0 Mode

General Register 0 points to the operand. Register 0 is not changed by FPMP-11.

3.1.3 Immediate Mode

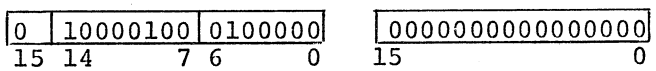
The operand immediately follows the TRAP instruction in the next two or four words depending on whether the operation is single or double precision.

3.1.4 Relative Mode

The address of the operand, relative to the PC, immediately follows the TRAP instruction. For example to address an operand at location A, code the word following the TRAP as .WORD A-.

EXAMPLE:

10_{10} is internally coded as:



which is 041040 000000 (octal). To add 10_{10} to the FLAC in each of the four modes (single precision):

Stack Mode:

```
      .
      .
      .
      MOV #000000,-(SP)          ;PUSH FLOATING
      MOV #041040,-(SP)          ;10 ONTO THE STACK
      TRAP ADR+STACKM            ;ADD TO FLAC
      .
      .
      .
```

Symbols ADR (for single precision add) and STACKM (for stack mode) are assigned values 12_8 and 0_8 respectively. (Refer to page 17.)

@R0 Mode:

```
      .
      .
      .
      MOV #TEN,R0                ;GET ADDRESS OF OPERAND IN R0
      TRAP ADR+ARM               ;ADD TO FLAC
      .
      .
      .
TEN:  .WORD 041040,000000        ;FLOATING POINT TEN
      .
      .
      .
```

Symbols ADR and ARM (@R0 mode) are assigned the values 12_8 and 100_8 respectively.

Immediate Mode:

```
      .
      .
      .
      TRAP ADR+IMMEDM            ;ADD TO FLAC
      .WORD 041040,000000        ;FLOATING POINT TEN
      .
      .
      .
```

Symbols ADR and IMMEDM (immediate mode) equal 12_8 and 200_8 respectively.

Relative Mode:

```
      .
      .
      .
      TRAP ADR+RELM            ;ADD TO FLAC
      .WORD TEN-.              ;RELATIVE ADDRESS OF OPERAND
      .
      .
      .
TEN:  .WORD 041040,000000      ;FLOATING POINT TEN
```

Symbols ADR and RELM (relative mode) equal 12_8 and 300_8 respectively.

To perform the above operations in double precision, use ADD=14_8 instead of ADR, and extend the floating point ten with two more words of zeros (i.e. TEN: .WORD 041040,0,0,0; double precision floating-point ten).

The source form of a TRAP call is:

```
TRAP num + mode
```

where num is the number of the OTS routine to be called (refer to Appendix D for the OTS routine numbers), and mode is one of the following addressing modes:

Mode	
0	Stack mode
100_8	@R0 Mode
200_8	Immediate mode
300_8	Relative mode

The binary form of the TRAP instruction is:

```
WORD:
┌─────────────────┐
│10001001 mmrrrrrr│
└─────────────────┘
15                0
```

Where

mm = addressing mode bits (00 = Stack mode, 01 = @R0 mode, 10 = Immediate mode, 11 = Relative mode)
rrrrrr = OTS routine number

It is suggested that commonly used addressing modes and routine numbers be referenced symbolically. For instance, the statements

```
STACKM=0                 ;STACK MODE
ARM=100                  ;@R0 MODE
IMMEDM=200               ;IMMEDIATE MODE
RELM=300                 ;RELATIVE MODE
```

```
ADR=12                  ;SINGLE PRECISION ADD ROUTINE
SBR=13                  ;SINGLE PRECISION SUBTRACT
MLR=21                  ;SINGLE PRECISION MULTIPLY
DVR=25                  ;SINGLE PRECISION DIVIDE
```

allow TRAP calls to be coded as follows:

```
TRAP ADR+RELM           ;ADD IN RELATIVE MODE
TRAP MLR+ARM            ;MULTIPLY IN @R0 MODE
TRAP SBR+IMMEDM         ;SUBTRACT IN IMMEDIATE MODE
```

Note that single argument, single result functions, such as square root (SQRT) require no addressing (refer to Appendix D); the argument is taken from the FLAC and the result is stored back into the FLAC. Consequently, the addressing mode of a TRAP call to such a function is ignored by the TRAP handler, and no address is used.

The TRAP handler sets the condition codes to reflect the contents of the FLAC after every operation except a compare. After any operation except floating point compare, the condition bits are set as follows:

Condition Codes

FLAC	N	Z	V	C
<0	1	0	0	0
=0	0	1	0	0
>0	0	0	0	0

After a floating point compare (either single or double precision), the condition codes are set as follows:

FLAC<OPR	1	0	0	0
FLAC=OPR	0	1	0	0
FLAC>OPR	0	0	0	0

where OPR is the operand addressed by the TRAP compare instruction.

EXAMPLE:

To calculate

$$X = \frac{-B+SQRT\ (B*B-4*A*C)}{2*A}$$

the following program might be written:

```
      .
      .
      .
      TRAP LDR+RELM           ;LOAD A INTO FLAC
      .WORD A-.               ;RELATIVE ADDRESS OF A
      TRAP MLR+IMM            ;MULTIPLY BY 2.0
FTWO: .WORD 040400,0          ;CONSTANT 2.0
      TRAP STR+RELM           ;STORE FLAC IN TEMP1
      .WORD TEMP1-.           ;RELATIVE ADDRESS OF TEMP1
      TRAP MLR+RELM           ;MPY BY 2.0 TO GET 4*A
      .WORD FTWO-.
      TRAP MLR+RELM           ;MPY BY C
      .WORD C-.
      TRAP STR+RELM           ;STORE FLAC IN TEMP2
      .WORD TEMP2-.
      MOV #B,R0               ;GET ADDRESS OF B INTO R0
      TRAP LDR+ARM            ;LOAD B INTO FLAC (@R0 MODE)
      TRAP MLR+ARM            ;CALCULATE B*B
      TRAP SBR+RELM           ;SUBTRACT 4*A*C (IN TEMP2)
      .WORD TEMP2-.
      TRAP SQRT               ;CALC SQUARE ROOT OF FLAC,
                              ;NO ADDRESSING REQUIRED
      TRAP SBR+ARM            ;ADD MINUS B
      TRAP DVR+RELM           ;DIVIDE BY 2.0*A IN TEMP1
      .WORD TEMP1-.
      TRAP STR+RELM           ;STORE FLAC INTO X
      .WORD X-.
      .
      .
      .
      A: .WORD 040400,0       ;VALUE OF A (2.0)
      B: .WORD 040640,0       ;VALUE OF B (5.0)
      C: .WORD 037600,0       ;VALUE OF C (0.25)
      X: .=.+4                ;LOCATION FOR RESULT
      TEMP1: .=.+4            ;TEMPORARY
      TEMP2: .=.+4            ;TEMPORARY
```

The above example assumes that the TRAP vector (location 34_8) has been initialized as previously described.

3.2 ACCESSING USER ROUTINES VIA THE TRAP HANDLER

Special floating-point functions may be coded as assembly language subroutines and accessed via TRAP calls if one of the following calling conventions is used:

1. POLISH - receive two arguments, either single or double precision, on the stack, and return one result, of the same precision as the arguments, on the stack. Return must be via a

   ```
   JMP  @(R4)+
   ```

2. J5RR - The user routine should be expecting a call of the following form:

```
        JSR   R5,subr    ;jump to subroutine
        BR    A
        .WORD arg        ;single argument's address
    A:
```

arg is the symbolic address of the subroutine's single or double precision argument. Note that the instruction following the JSR is not necessarily a BR. The returned result in registers R0-R3 is stored in the FLAC by the TRAP handler.

Furthermore, user routines to be called by the TRAP handler must reside within the 8K words physically following the beginning of the TRAP handler in memory, and an entry must be made in the TRAP handler's dispatch table. The dispatch table called TBL$42 in TRAPH, is organized as follows:

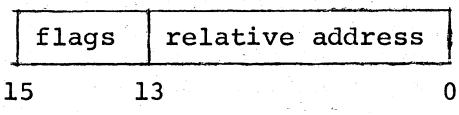
1. There is a one word entry corresponding to each routine-number which can be coded in a TRAP call (total of 64 words).

2. The position of the word in the table corresponds to the routine-number which calls it (e.g. the word at location TBL$42 is referenced by "TRAP 0+mode", while the word at location TBL$42+10. is referenced by "TRAP 5+mode"). In general, the word at location TBL$42+2n is referenced by the call TRAP n+mode.

3. The word at each table location is coded as follows:

   a. 0-indicates no routine corresponds to this table entry.

   b.



bit 15 set to 0 = single precision routine
1 = double precision
bit 14 set to 0 = J5RR mode call
1 = POLISH mode call
bits 13-0 = The address of the entry point of the routine to be called minus the address of the label PT$42 in TRAPH.

In J5RR mode, TRAPH supplies the FLAC as the single argument and stores the result back into the FLAC. No explicit address is accepted in the TRAP instruction. In POLISH mode, TRAPH uses the FLAC as one argument and the location addressed in the TRAP call as the second. The result is stored in the FLAC. Refer to section 3.1 for addressing modes in TRAP calls to FPMP-11.

3.3 DIRECT CALLS TO OTS ROUTINES

Occasionally it is desirable to call OTS routines directly. For instance, some routines cannot be accessed using the TRAP handler (refer to Appendix D). Furthermore, eliminating the TRAP handler overhead decreases the execution time of the user program. Note that when called directly, the OTS routines do not preserve the contents of the general registers, nor do they in general set the condition codes to reflect the result of the operation, these functions are performed by the TRAP handler when it is used.

Any of the OTS routines can be directly called by using its FPMP-11 global entry point and observing the proper calling conventions. Calling conventions fall into a few basic types as follows (the calling conventions for each routine are given in Appendix D):

3.3.1 Polish Mode

Polish mode calls are designed to be most effective in a compiler-generated environment. They are easily produced by a compiler and are particularly efficient in storage space used and interpretation overhead.

The routines that are called with Polish mode are:

Name	No. of Arguments	Location of Result
$ADD	2	4 word sum on top of stack
$ADR	2	2 word sum on top of stack
$CMD	2	sets condition codes
$CMR	2	sets condition codes
$DINT	1	integer result on top stack
$DR	1	2 word result on top of stack
$DVD	2	4 word quotient on top of stack
$DVI	2	integer quotient on top of stack
$DVR	2	2 word quotient on top of stack
$ID	1	4 word result on top of stack
$IR	1	2 word result on top of stack
$INTR	1	result on top of stack
$MLD	2	result on top of stack
$MLI	2	result on top of stack
$MLR	2	result on top of stack
$NGD	1	result on top of stack
$NGI	1	result on top of stack
$NGR	1	result on top of stack
$POPR5	4	result in registers R0-R3
$POPR4	4	result in registers R0-R3
$POPR3	2	result in registers R0,R1
$PSHR1	1	result on top of stack
$PSHR2	1	result on top of stack
$PSHR3	2	result on top of stack
$PSHR4	4	result on top of stack

Name	No. of Arguments	Location of Result
$PSHR5	4	result on top of stack
$RD	1	result on top of stack
$DI	1	result on top of stack
$RI	1	result on top of stack
$SBD	2	result on top of stack
$SBR	2	result on top of stack

Each routine called in Polish mode pops the necessary arguments off the R6 (General Register 6) stack and pushes the final result onto the stack. Multi-word arguments are always pushed onto the stack low-order word first, so that the highest-order word (the one containing the sign and exponent) remains on top of the stack (@SP).

Arguments must be pushed onto the stack before entering Polish mode so that the source operand is on the top of the stack and the destination operand is next down from the top.

Polish mode is entered with a JSR in the form

```
JSR R4,$POLSH
```

where $POLSH is a global subroutine in FPMP-11.

Routines to be used are then called by supplying a word with the address of the routine.

```
.WORD $ADR
```

Exit from Polish mode is by coding a word containing the address of the next instruction to be executed. For example to execute the next instruction in sequence,

```
.WORD .+2
```

Using Polish mode, coding to calculate (A+B)*C with the single precision routines might be written as:

```
.
.
.
.GLOBL $POLSH,$ADR,$MLR
.
.
.
MOV C+2,-(SP)           ;PUSH C ONTO STACK.
MOV C,-(SP)
MOV B+2,-(SP)           ;PUSH B ONTO STACK.
MOV B,-(SP)
MOV A+2,-(SP)           ;PUSH A ONTO STACK.
MOV A,-(SP)
JSR R4,$POLSH           ;ENTER POLISH MODE
.WORD $ADR              ;ADD A TO B AND LEAVE
                        ;THE RESULT ON THE STACK
.WORD $MLR              ;MULTIPLY PREVIOUS SUM BY
                        ;C AND LEAVE RESULT ON STACK.
.WORD .+2               ;LEAVE POLISH MODE.
.
.
.
```

After execution of the above code, the result of the calculation (A+B)*C is on the top of the R6 stack.

The routine "$POLSH" that causes entry into Polish mode is located at global entry $POLSH in FPMP-11. It is coded as follows:

```
          .GLOBL    $POLSH
$POLSH:   TST (SP)+           ;DELETE OLD VALUE OF R4 FROM
                              ;THE TOP OF THE STACK.
          JMP @(R4)+          ;ENTER POLISH MODE.
```

Each routine called in Polish mode takes its operands from the top of the stack and pushes its result, if any, back onto the stack. Each routine returns with a "JMP @(R4)+" which passes control to the next routine in sequence. User routines can be written and called in Polish mode provided they preserve the contents of R4 and return by executing a "JMP @(R4)+". The following is an example of a user subroutine written for Polish calls.

```
DUP:      MOV  2(SP),-(SP)    ;DUPLICATE STACK ITEM
          MOV  2(SP),-(SP)    ;TWO WORD ITEM
          JMP  @(R4)+         ;RETURN
```

When executed, this subroutine duplicates the two-word item on the top of the stack.

3.3.2 J5RR Mode

J5RR is the calling convention used by most of the FORTRAN library functions. J5RR mode calls are of the form

```
     JSR  R5,subroutine
```

All argument addresses are placed in a list following the subprogram call. The generalized standard sequence is:

```
          .GLOBL SUBR
          JSR R5,SUBR
          BR XX
          A
          B
          .
          .
          .
          Z
XX:
```

where A, B...Z are argument addresses.

Subprograms are responsible for not altering the contents of register R5 since it is the parameter list pointer.

The results of subroutines called in J5RR mode are generally stored as follows: integer results are returned in R0, two-word floating point results in R0 and R1 and four-word results in R0-R3.

An example of a call in J5RR mode is this call to calculate the square root of X:

```
          .GLOBL  SQRT
          .
          .
          .
          JSR  R5,SQRT          ;CALL TO SQRT ROUTINE
          BR  A                 ;RETURN POINT
          .WORD  X              ;ADDRESS OF ARGUMENT
A:        .                     ;CONTINUE PROGRAM
          .
          .
X:        .WORD 040400,000000 ;2 WORD FLOATING POINT NUMBER,
                                ;VALUE OF X=2.
```

In this example, the result is returned as a two-word floating point number in R0-R1.

The functions which use J5RR mode calls are:

Function	# of Arguments	Register(s) for Result
ALOG	1	R0,R1
ALOG10	1	R0,R1
AINT	1	R0,R1
ATAN	1	R0,R1
ATAN2	2	R0,R1
DBLE	1	R0-R3
DLOG	1	R0,R3
DLOG10	1	R0,R3
DCOS	1	R0,R3
DSIN	1	R0-R3
DSQRT	1	R0-R3
DATAN	1	R0-R3
DATAN2	2	R0-R3
DEXP	1	R0-R3
EXP	1	R0,R1
FLOAT	1	R0,R1
IFIX	1	R0
IDINT	1	R0
INT	1	R0
SIN	1	R0-R1
COS	1	R0-R1
SNGL	1	R0,R1
TANH	1	R0,R1

3.3.3 JPC Mode

The JPC mode of subroutine call is used for communicating with the ASCII conversion routines in FPMP-11. With JPC mode, the arguments must be pushed onto the stack before the subroutine is called. The call to each individual subroutine is listed in Table 3-1. In general, a JPC mode call is coded as follows:

```
.
.
.
MOV R3,-(SP)          ;push first argument onto stack
.
.
.
MOV #ARG,-(SP)        ;push last argument onto stack
JSR PC,subr           ;call subroutine
.
.
.
```

For example, to convert a ten character ASCII field at location BUFFER to internal single precision format, the following might be coded:

```
.
.
.
MOV #BUFFER,-(SP)     ;PUSH ADDRESS OF FIELD
MOV #10.,-(SP)        ;PUSH LENGTH OF FIELD
CLR -(SP)             ;D-SCALE IS ZERO
CLR -(SP)             ;P-SCALE IS ZERO
JSR PC,$RCI           ;CALL CONVERSION ROUTINE
.
.
.
```

After the above code is executed, the internal representation of the number at location BUFFER is in the top two words of the stack. The ten characters at location BUFFER can be read from an I/O device, or coded as constants: For example,

```
BUFFER: .ASCII /113.25bbbb/
```

or

```
BUFFER: .ASCII /-3.627E+09/
```

TABLE 3-1

ROUTINES WHICH USE THE JPC MODE OF CALL

Name	Description	# of Arg	Call Format	Location Of Result
$DCI	ASCII to dbl. prec.	4	Push addr. of start of ASCII field Push length of ASCII field in bytes Push format scale D (from W.D) position of assumed decimal point Push P format scale JSR PC,$DCI	4 word result on top of stack
$DCO	Dbl. prec. to ASCII	4	Push addr. of start of ASCII field Push length of ASCII field in bytes Push D part of W.D (position of decimal point) Push P scale Push 4 word value to be converted lowest order word first JSR PC,$DCO	ASCII field specified
$ECO	Single prec. to ASCII E format	4	Same calling sequence as $DCO except that a 2 word value is to be converted. JSR PC,$ECO	ASCII field Specified
$FCO	Single prec. to ASCII F format	5	Same calling sequence as $ECO. JSR PC,$FCO	ASCII field specified
$GCO	Single prec. to ASCII G format	5	Same calling as $ECO. JSR PC,$GCO	ASCII field specified.
$ICI	ASCII to integer	2	Push addr. of start of ASCII field Push length in bytes of ASCII field JSR PC,$ICI	Integer result on top of stack
$ICO	Integer to ASCII	3	Push addr. of ASCII field Push length of ASCII field in bytes Push integer value to be converted JSR PC,$ICO	ASCII field specified

TABLE 3-1 (Cont.)

ROUTINES WHICH USE THE JPC MODE OF CALL

Name	Description	# of Arg.	Call Format	Location Of Result
$OCI	ASCII to octal	3	Same calling sequence as $ICI JSR PC,$OCI	Top of stack
$OCO	Octal to ASCII	3	Same calling sequence as $ICO JSR PC,$OCO	ASCII field specified
$RCI	ASCII to single prec.	4	Same calling sequence as $DCI JSR PC,$RCI	Two word result on top of stack

The ASCII input conversion subroutines $RCI, $DCI, $ICI, and $OCI preserve the contents of the general registers and restore them to their original values before returning. The ASCII output conversion subroutines $DCO, $ECO, $FCO, $GCO, $ICO, and $OCO destroy the contents of general registers R0, R1, R2, and R3, but preserve the contents of R4 and R5.

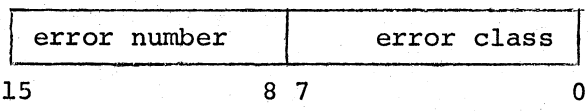
Errors detected by the ASCII input conversion subroutines $RCI, $DCI, $ICI, and $OCI cause the subroutine to return with a zero result and with the C bit set in the condition codes.

3.4 ERRORS

All errors in floating-point operations, such as overflow of the FLAC or an illegal TRAP instruction, are handled by the routines $ERR and $ERRA. These routines save the contents of R0, and load the error code into R0. The routines then perform a JSR PC,@$ERVEC. $ERVEC is a global location which is initially set to contain the address of a HALT instruction but can contain the address of a user error handling routine. If the user error handling routine is to be used, code is inserted in the initialization of the program as explained in section 3.4.1.

The error code generated by the FPMP-11 subroutine is put in R0 in the following format:

R0:



Error codes and their meanings are shown in Table 3-2.

3.4.1 User Error Handling Routines

To include a user error handling routine in a program, the following code must be included in the initialization of the program.

```
.
.
.
.GLOBL $ERVEC
MOV #ERROR,$ERVEC    ;move address of error routine
.
.
.
ERROR: user's error handling routine
.
.
.
```

The error handling routine can be written to terminate with a HALT instruction or, if registers 1 through 5 are saved, to continue the program by executing an RTS PC instruction. The only exception is the halt after an illegal TRAP instruction (error 0,0) from which it is impossible to continue. If such a TRAP occurs, its address is in R1 when the error routine is called.

TABLE 3-2

FPMP-11 ERROR CODES

ERROR CODE (CLASS,#)	ISSUED BY	EXPLANATION
0,0	TRAPH	Illegal TRAP instruction. R1 points to the TRAP instr.
3,1	$ADD	Expon. overflow in double prec. addition
3,2	$ADR	Exponent overflow in real addition
3,3	$DVD	Double prec. div. by zero
3,4	$DVD	Expon. overflow in double precision division
3,5	$DVI	Integer division by 0
3,6	$DVR	Expon. overflow in real division
3,8	$DVR	Real division by zero
3,10	$MLD	Expon. overflow in double prec. mult.
3,11	$NEG	Exponent overflow during negation
3,12	$MLR	Expon. overflow in real multiplication
3,14	$MLI	Product outside of range on integer mult.
3,22	$RI	Real outside range on real to integer conversion
3,23	$DR	Exponent overflow on double to real conversion
4,2	DEXP	DEXP argument greater than 87
4,3	DLOG	DLOG argument less than or equal to zero
4,4	DSQRT	DSQRT argument less than zero
4,5	EXP	EXP argument greater than 87
4,10	ALOG	ALOG argument less than or equal to zero
4,11	SQRT	SQRT argument less than zero
4,12	SNGL	SNGL exponent overflow in round
5,1	$ADD	Expon. underflow in double prec. addition (warning)
5,2	$ADR	Exponent underflow in real addition (warning)
5,3	$DVR	Expon. underflow in real div.(warning)
5,4	DEXP	DEXP argument less than -88.7 (warning)
5,5	EXP	EXP argument less than -88.7 (warning)
5,6	$MLD	Expon. underflow in double prec. mult. (warning)
5,7	$MLR	Expon. underflow in real multiplication
5,8	$DVD	Expon. underflow in double prec. division (warning)

3.5 CREATING SPECIAL PACKAGES

FPMP-11 source code includes PAL-11S conditional assembly instructions which allow tailoring of the FPMP-11 package to include only the functions required by the user program. (Refer to the PAL-11S manual (DEC-11-YRWB-D) for information on conditional assembly instructions.) The desired routines are then assembled to take advantage of whatever hardware features are available.

3.5.1 Assembly Switch Tape

To take advantage of the conditional assembly instructions in the FPMP-11 source code, a separate tape which sets the switches of the desired routines and hardware must be prepared and included in the assembly of the FPMP-11 package.

The switches are set by statements which assign a value to the switch name. For example, to indicate the availability of the 11/45 FPU hardware, the FPU switch is set with the following statement

```
FPU=1
```

When the FPU switch is set, many FPMP-11 routines assemble differently to take advantage of the FPU.

When using the PDP-11/45 FPU option, it is the user's responsibility to set up the FPU TRAP vector (location 244_8) and the FPU status register (refer to the PDP-11/45 Processor Handbook). Refer to Table 3-3 for hardware switch option names.

Significant size and speed advantages can be expected if one of the hardware options is present and its corresponding switch is set. If no hardware option switch is set the assembler assumes the program uses the basic PDP-11 instruction set. In no case should more than one hardware option switch be set during an assembly.

TABLE 3-3

HARDWARE OPTION SWITCHES

Switch Name	Hardware Option
FPU	PDP-11/45 floating point unit
EAE	PDP-11/20 EAE
MULDIV	PDP-11-40 extended instruction set (EIS) or PDP-11/45 processor

NOTE

If the FPU switch is set during an assembly, the assembler being used must be capable of processing the extended op codes which will appear. The present version (V002A) of PAL-11S does not support these op codes. MACRO-11 can be used for assembly when the FPU switch is set.

Each section of code in the FPMP-11 package is assigned a number and the switch to cause a particular section of code to be included is called CND$n.

Table 3-4 lists the sections of the FPMP-11 package, the routines contained in each section and the switch name to be used.

For example, to include the DSQRT routine in the package set the switch with the following code:

```
CND$14=1
.EOT
```

TABLE 3-4

CONDITIONAL FPMP-11 ASSEMBLY CODES

Section No.	Switch Name	Subroutine Contained
1	CND$1	$ADD,$SBD
2	CND$2	$ADR,$SBR
3	CND$3	ALOG,ALOG10
4	CND$4	AINT,$INTR
5	CND$5	$CMD
6	CND$6	$CMR
7	CND$7	DBLE
8	CND$8	$DCI,$RCI
9	CND$9	$DCO,$ECO,$FCO,$GCO
10	CND$10	DLOG,DLOG10
11	CND$11	$DINT
12	CND$12	$DR
13	CND$13	DSIN,DCOS
14	CND$14	DSQRT
15	CND$15	DATAN,DATAN2
16	CND$16	$DVD
17	CND$17	$DVI
18	CND$18	$DVR
19	CND$19	DEXP
20	CND$20	EXP
21	CND$21	$FCALL
22	CND$22	IFIX
23	CND$23	FLOAT
24	CND$24	$ICI,$OCI
25	CND$25	$ICO,$OCO
26	CND$26	INT,IDINT
27	CND$27	$ID,$IR
28	CND$28	$MLD
29	CND$29	$MLI
30	CND$30	$MLR
31	CND$31	$NGI,$NGR,$NGD
32	CND$32	$PSHR5,$PSHR4,$PSHR3,$PSHR2,$PSHR1
33	CND$33	$POPR5,$POPR4,$POPR3
34	CND$34	$RD
35	CND$35	$RI,$DI
36	CND$36	SNGL
37	CND$37	SIN,COS
38	CND$38	TANH
39	CND$39	ATAN,ATAN2
40	CND$40	$POLSH(switch is always set)
41	CND$41	SQRT
42	CND$42	TRAPH
43	CND$43	$ERR,$ERRA(switch is always set)

The CLASS5 switch can be set (CLASS5=1) to have class 5 (warning) messages interpreted by the error handler of FPMP-11. Normally class 5 errors are ignored. Many of the FPMP-11 transcendental and trigonometric functions do not operate properly if the class 5 switch is set.

There are two additional switches which work together with the others. When these switches are set the standard single or double precision TRAP handler packages are assembled. The two switches are:

SINGLE	Assemble the standard single precision (2 word) package when set
DOUBLE	Assemble the standard double precision (4 word) package when set.

The contents of the standard packages are listed in Chapter 2. The SINGLE and DOUBLE switches may be set together to produce a combined package containing both standard packages. It is also possible to include a few double precision subroutines with the standard single precision package or to include some of the non-standard routines (e.g. integer multiply), with the single and/or double precision package. More information on creating these special combinations is given in section 3.5.1.1.

3.5.1.1 Preparing the Assembly Switch Tape

To assemble the FPMP-11 source tape:

1. Decide which FPMP-11 routines are to be included in the resulting package. Refer to Appendix D for a list of available routines.

2. Obtain the switch names for the desired routines from Table 3-4.

3. Decide which, if any, of the hardware option switches is to be set.

4. Create a paper tape or source file (either off-line or using the editor) in the following format; (Refer to the Paper Tape Software Programming Handbook for information on using the editor).

```
switch-name-1 =1     ;FIRST SWITCH TO BE SET
switch-name-2 = 1
.
.
.
switch-name-n = 1    ;LAST SWITCH TO BE SET
.EOT
```

(Where "switch-name-1" thru "switch-name-n" are the names of the switches to be set.) If preparing the tape off line, be sure to

put a carriage return/line feed after each line. For example, to assemble the standard single precision package to take advantage of the EAE, create the following tape:

```
SINGLE=1        ;USE STANDARD 2-WORD PKG.
EAE=1           ;SPECIFY EAE
.EOT
```

To assemble a standard double precision package plus integer multiply and divide, create the following tape:

```
DOUBLE = 1      ;GET STD 4-WORD PKG.
CND$17 = 1      ;INTEGER DIVIDE
CND$29 = 1      ;INTEGER MULTIPLY
.EOT
```

It is not necessary to worry about interdependency among FPMP-11 routines. For example, to create a package containing only the single precision function TANH, the tape

```
CND$38 = 1      ;TANH
.EOT
```

is sufficient. The fact that the TANH function calls the arithmetic routines and other internal functions is resolved by the FPMP-11 source code. In particular, the above switch being set causes the following routines to be included; TANH, EXP, $ADR, $SBR, $MLR, $DVR, $FCALL, $POLSH, $PSHR3, $ERR, $IR, and $RI.

5. Assemble the FPMP-11 package with PAL-11S loading the FPMP-11 source tapes (1 thru 6) last. Refer to Appendix B.

6. The object module produced by PAL-11S can now be used as described in section 3.6.

NOTE

Because of limitations in the symbol table size in the 8K version (V002A) of PAL-11S, it is not possible to include all FPMP-11 routines in a single assembly. The error message produced by the assembler is "S" and the assembly is aborted. It is possible however, to assemble as much as the standard single and double precision packages together. If the integer and conversion routines not included in the standard packages are needed along with both standard packages, they can be assembled separately by PAL-11S, and the resulting tape then linked with the standard packages using the LINK-11S linker. If this procedure is used, the linker produces error messages because of the multiple occurrence of the labels $POLSH, $V20A, $ERR, and $ERRA. These are non-fatal errors and can be ignored.

3.6 LOADING INSTRUCTIONS

The FPMP-11 package can be used as distributed by linking the object tapes (single or double precision) with the user object program or by using the source tapes to assemble a user-tailored package and then linking the package to the user program.

The Bootstrap and Absolute Loaders must be resident in core before any of the other programs can be loaded. Refer to Appendix A for loading instructions.

The object tape of the user program produced by PAL-11S (or DOS MACRO-11), and the FPMP-11 object tape are linked with LINK-11S (or DOS LINK-11) (refer to Appendix C for LINK-11S instructions). LINK-11S requires two passes and produces a tape called a load module which contains the user program and the FPMP-11 routines.

Use the Absolute Loader to load this module and execute the program. (Refer to Appendix A for details on using the ABS Loader.)

CHAPTER 4
SAMPLE PROGRAM

The following sample program illustrates most of the FPMP-11 modes of calls. Note that execution of this sample program requires the use of the Input/Output Executive (IOX) program which must be loaded before the sample program. This program inputs three F10.0 numbers, stores them as A,B and C and prints the numbers stored for verification. The roots of $AX^2+BX+C=\emptyset$ are calculated using the formula $X=\frac{-B\pm\sqrt{B^2-4AC}}{2A}$. If A=Ø the program halts.

```
                        .TITLE XAMPLE
          000000        R0=%0
          000001        R1=%1
          000002        R2=%2
          000003        R3=%3
          000004        R4=%4
          000005        R5=%5
          000006        SP=%6
          000007        PC=%7
          000100        ARM=100
          000200        IMM=200
          000300        RELM=300
          000071        LDR=71
          000073        STR=73
          000012        ADR=12
          000013        SBR=13
          000021        MLR=21
          000025        DVR=25
          000046        SQRT=46
          000011        MSGLEN=9.
          000002        RESET=2
          000011        READOP=11
          000004        WAITR=4
          000012        WRITE=12
                        .GLOBL  TRAPH,$RCI,$FCO
000000 012706 BEGIN:    MOV     #2000,SP;       INITIALIZE STACK
       002000
000004 000004           IOT
000006 000000           .WORD   0;              INIT THE IOX PACKAGE
000010    002           .BYTE   RESET,0
000011    000
000012 000004           IOT
000014 000504'          .WORD   TITLE
000016    012           .BYTE   WRITE,1;        WRITE THE TITLE
000017    001
000020 012737'          MOV     #TRAPH,@#34;    INITIALIZE TRAP VECTOR
       000000
       000034
000026 012737           MOV     #340,@#36
       000340
       000036
000034 004767 RESTAR:   JSR     PC,READ;        READ ONE INPUT LINE INTO BUFFR
       000374
000040 012701'          MOV     #BUFFR+6,R1;    GET ADDR OF BEGINNING OF BUFFER
       000646
000044 012700'          MOV     #A,R0;          GET ADDR OF VAR 'A'
       000454
```

```
000050 010146 ILOOP:  MOV     R1,-(SP);       SAVE R1
000052 010046         MOV     R0,-(SP);       SAVE R0
000054 010146         MOV     R1,-(SP);       PUSH ADDR OF ASCII STRING READ
000056 012746         MOV     #10.,-(SP);     PUSH LENGTH
       000012
000062 005046         CLR     -(SP);          D FORMAT SCALE
000064 005046         CLR     -(SP);          P SCALE
000066 004767'        JSR     PC,$RCI;        CONVERT ONE NUMBER (F10.0)
       000000
000072 104471         TRAP    LDR;            LOAD FLAC FROM TOP OF STACK
000074 104573         TRAP    STR+ARM;        STORE INTO VARIABLE A, B, OR C
000076 012600         MOV     (SP)+,R0;       RESTORE R0
000100 012601         MOV     (SP)+,R1;       AND R1
000102 022020         CMP     (R0)+,(R0)+;    INCR R0 BY 4
000104 062701         ADD     #10.,R1;        INCR BUFFER POINTER TO NEXT VAR
       000012
000110 012705'        MOV     #MSGBLK,R5
       000567
000114 004767         JSR     PC,PRINT;       CALL PRINT SUBROUTINE
       000174
000120 020027'        CMP     R0,#C;          LAST VAR?
       000464
000124 101751         BLOS    ILOOP;          LOOP
000126 104771         TRAP    LDR+RELM;       LOAD A INTO FLAC
000130 000324         .WORD   A-.             ;RELATIVE ADDRESS OF A
000132 001547         BEQ     END;            EXIT IF A = 0
000134 104712         TRAP    ADR+RELM;       A + A TO GIVE 2*A
000136 000316         .WORD   A-.
000140 104773         TRAP    STR+RELM;       STORE 2*A INTO TEMP1
000142 000326         .WORD   TEMP1-.
000144 104621         TRAP    MLR+IMM;        MPY BY 2 TO GET 4*A (IMMED MODE
000146 040400         .WORD   040400,000000;  CONST 2.0
000150 000000
000152 104721         TRAP    MLR+RELM;       MPY BY C
000154 000310         .WORD   C-.
000156 104773         TRAP    STR+RELM;       STORE 4*A*C IN TEMP2
000160 000314         .WORD   TEMP2-.
000162 012700'        MOV     #B,R0;          GET ADDRESS OF VARIABLE 'B'
       000460
000166 104571         TRAP    LDR+ARM;        LOAD B INTO FLAC
000170 104521         TRAP    MLR+ARM;        MPY BY B TO GET B**2
000172 104713         TRAP    SBR+RELM;       SUBTRACT 4*A*C
000174 000300         .WORD   TEMP2-.
000176 001430         BEQ     ROOT1;          BRANCH IF ONLY ONE ROOT
000200 002441         BLT     IMAG;           B**2 - 4*A*C < 0 ???
000202 104446         TRAP    SQRT;           TAKE SQRT OF FLAC
000204 104773         TRAP    STR+RELM;       SAVE SQRT(B**2-4*A*C) IN TEMP2
000206 000266         .WORD   TEMP2-.
000210 104513         TRAP    SBR+ARM;        ADD MINUS B
000212 104725         TRAP    DVR+RELM;       DIVIDE BY 2*A (IN TEMP1)
000214 000254         .WORD   TEMP1-.
000216 012705'        MOV     #MSG1,R5;       ADDR OF "ROOT 1 = " MESSAGE
       000534
000222 004767         JSR     PC,PRINT        ;CALL PRINT SUBROUTINE
       000066
000226 104671         TRAP    LDR+IMM;        ZERO THE FLAC (IMMEDIATE MODE)
000230 000000 ZERO:   .WORD   0,0             ;FLOATING POINT ZERO
000232 000000
000234 104513         TRAP    SBR+ARM;        - B
000236 104713         TRAP    SBR+RELM;       -SQRT(B**2-4*A*C)
000240 000234         .WORD   TEMP2-.
000242 104725         TRAP    DVR+RELM;       DIVIDE BY 2*A
000244 000224         .WORD   TEMP1-.
000246 012705'        MOV     #MSG2,R5;       ADDR OF "ROOT 2 = "
       000545
```

```
000252 004767          JSR     PC,PRINT
       000036
000256 000666          BR      RESTAR;         BRANCH TO GO AGAIN
000260 104771 ROOT1:   TRAP    LDR+RELM;       ZERO THE FLAC
000262 177746          .WORD   ZERO-.
000264 104513          TRAP    SBR+ARM;        GET - B
000266 104725          TRAP    DVR+RELM;       DIV BY 2*A
000270 000200          .WORD   TEMP1-.
000272 012705'         MOV     #MSG3,R5;       "ROOT =    "
       000556
000276 004767          JSR     PC,PRINT
       000012
000302 000654          BR      RESTAR
000304 000004 IMAG:    IOT;                    WRITE IMAGINARY ROOTS MESSAGE
000306 000600'         .WORD   MSG4;           ADDRESS OF MESSAGE BUFFER
000310    012          .BYTE   WRITE,1;        WRITE TO SLOT 1
000311    001
000312 000650          BR      RESTAR

               ;       PRINT SUBROUTINE
000314 010546 PRINT:   MOV     R5,-(SP);       SAVE REGS
000316 010446          MOV     R4,-(SP)
000320 010346          MOV     R3,-(SP)
000322 010246          MOV     R2,-(SP)
000324 010146          MOV     R1,-(SP)
000326 010046          MOV     R0,-(SP)
000330 104773          TRAP    STR+RELM;       STORE THE FLAC
000332 000146          .WORD   TEMP3-.
000334 012704          MOV     #MSGLEN,R4;     CHAR COUNT FOR MESSAGE
       000011
000340 012703'         MOV     #OBUF+6,R3;     ADDR OF OUTPUT FIELD
       000774
000344 112523 MLOOP:   MOVB    (R5)+,(R3)+;    MOV THE CHAR MESSAGE
000346 005304          DEC     R4
000350 001375          BNE     MLOOP
000352 010346          MOV     R3,-(SP);       ADDR OF OUTPUT FIELD FOR CONV
000354 012746          MOV     #20.,-(SP);     LEN OFFIELD
       000024
000360 012746          MOV     #10.,-(SP);     DECIMAL PLACES
       000012
000364 005046          CLR     -(SP);          P SCALE
000366 016746          MOV     TEMP3+2,-(SP);  PUSH VALUE TO BE CONVERTED
       000110
000372 016746          MOV     TEMP3,-(SP)
       000102
000376 004767'         JSR     PC,$FCO;        CALL CONVERSION ROUTINE IN FPMP
       000000
000402 000004          IOT;                    CALL THE IOX PACKAGE
000404 000766'         .WORD   OBUF;           WRITE THE OUTPUT BUFFER
000406    012          .BYTE   WRITE,1;        TO SLOT 1 (KB)
000407    001
000410 000004 WAITO:   IOT;                    CALL IOX
000412 000410'         .WORD   WAITO;          CREATE WAIT LOOP
000414    004          .BYTE   WAITR,1;        WAIT FOR SLOT 1 (KB)
000415    001
000416 012600          MOV     (SP)+,R0;       RESTORE REGS
```

```
000420 012601          MOV     (SP)+,R1
000422 012602          MOV     (SP)+,R2
000424 012603          MOV     (SP)+,R3
000426 012604          MOV     (SP)+,R4
000430 012605          MOV     (SP)+,R5
000432 000207          RTS     PC;             RETURN

               ;       READ SUBROUTINE
000434 000004 READ:    IOT;                    CALL IOX FOR READ
000436 000640'         .WORD   BUFFR;          ADDR OF INPUT BUFFER
000440     011         .BYTE   READOP,0;       READ SLOT 0 (KB)
000441     000
000442 000004 WAITI:   IOT;                    CALL IOX
000444 000442'         .WORD   WAITI;          CREATE WAIT LOOP
000446     004         .BYTE   WAITR,0;        WAIT FOR SLOT 0 (KB)
000447     000
000450 000207          RTS     PC;             RETURN
000452 000000 END:     HALT;                   FINISHED
000454 000000 A:       .WORD   0,0
000456 000000
000460 000000 B:       .WORD   0,0
000462 000000                  ,
000464 000000 C:       .WORD   0,0
000466 000000
000470 000000 TEMP1:   .WORD   0,0
000472 000000
000474 000000 TEMP2:   .WORD   0,0
000476 000000
000500 000000 TEMP3:   .WORD   0,0
000502 000000
000504 000022 TITLE:   .WORD   18.
000506 000000          .WORD   0
000510 000022          .WORD   18.
000512     015         .BYTE   15,12
000513     012
000514     124         .ASCII  /TEST OF FPMP11/
000515     105
000516     123
000517     124
000520     040
000521     117
000522     106
000523     040
000524     106
000525     120
000526     115
000527     120
000530     061
000531     061
000532     015         .BYTE   15,12
000533     012
000534     122 MSG1:   .ASCII  /ROOT 1 = /
000535     117
000536     117
000537     124
000540     040
```

```
000541     061
000542     040
000543     075
000544     040
000545     122 MSG2:    .ASCII  /ROOT 2 = /
000546     117
000547     117
000550     124
000551     040
000552     062
000553     040
000554     075
000555     040
000556     122 MSG3:    .ASCII  /ROOT =   /
000557     117
000560     117
000561     124
000562     040
000563     075
000564     040
000565     040
000566     040
000567     040 MSGBLK:  .ASCII  /         /
000570     040
000571     040
000572     040
000573     040
000574     040
000575     040
000576     040
000577     040
         000600          .EVEN
000600 000032 MSG4:      .WORD   26.
000602     000           .BYTE   0,0
000603     000
000604 000032            .WORD   26.
000606     015           .BYTE   15,12
000607     012
000610     122           .ASCII  /ROOTS ARE IMAGINARY***/
000611     117
000612     117
000613     124
000614     123
000615     040
000616     101
000617     122
000620     105
000621     040
000622     111
000623     115
000624     101
000625     107
000626     111
000627     116
000630     101
000631     122
```

```
000632     131
000633     052
000634     052
000635     052
000636     015                 .BYTE   15,12
000637     012
           000640              .EVEN

000640 000120 BUFFR:   .WORD   80.
000642     000         .BYTE   0,0
000643     000
000644 000000          .WORD   0
           000766      .=.+80.

000766 000040 OBUF:    .WORD   32.
000770     000         .BYTE   0,0
000771     000
000772 000040          .WORD   32.
000774     040         .ASCII  /          /
000775     040
000776     040
000777     040
001000     040
001001     040
001002     040
001003     040
001004     040
           001031      .=.+20.
001031     015         .BYTE   15,12,12
001032     012
001033     012
           000000      .END    BEGIN

A          000454R     ADR     = 000012    ARM     = 000100
B          000460R     BEGIN     000000R   BUFFR     000640R
C          000464R     DVR     = 000025    END       000452R
ILOOP      000050R     IMAG      000304R   IMM     = 000200
LDR      = 000071      MLOOP     000344R   MLR     = 000021
MSGBLK     000567R     MSGLEN  = 000011    MSG1      000534R
MSG2       000545R     MSG3      000556R   MSG4      000600R
OBUF       000766R     PC      =%000007    PRINT     000314R
READ       000434R     READUP  = 000011    RELM    = 000300
RESET    = 000002      RESTAR    000034R   ROOT1     000260R
R0       =%000000      R1      =%000001    R2      =%000002
R3       =%000003      R4      =%000004    R5      =%000005
SBR      = 000013      SP      =%000006    SQRT    = 000046
STR      = 000073      TEMP1     000470R   TEMP2     000474R
TEMP3      000500R     TITLE     000504R   TRAPH   = ****** G
WAIT1      000442R     WAITU     000410R   WAITR   = 000004
WRITE    = 000012      ZERO      000230R   $FCO    = ****** G
$RCI     = ****** G    .       = 001034R
```

```
TEST OF FPMP11
      2.0        4.00    2.0                ;Teletype input in three
                  2.0000000000              ;10-character fields

                  4.0000000000

                  2.0000000000              ;program verification
                                            ;of input
ROOT =           -1.0000000000              ;result

      12.5       3.25      5.43
                 12.5000000000

                  3.2500000000

                  5.4299998283

ROOTS ARE IMAGINARY***

    3.E-01    .06E002    40E-001
                 0.3000000119

                 6.0000000000

                 4.0000000000

ROOT 1 =         -0.6905062795

ROOT 2 =        -19.3094921112

         5          15          3
                  5.0000000000

                 15.0000000000

                  3.0000000000

ROOT 1 =         -0.2154767066

ROOT 2 =         -2.7845234871

         0       4.0        3.75
                  0.0000000000

                  4.0000000000

                  3.7500000000
```

PROGRAM OUTPUT

APPENDIX A

BOOTSTRAP AND ABSOLUTE LOADERS

A.1 THE BOOTSTRAP LOADER

A.1.1 Loading the Bootstrap Loader

The Bootstrap Loader should be toggled into the highest core memory bank.

xx7744	016701
xx7746	000026
xx7750	012702
xx7752	000352
xx7754	005211
xx7756	105711
xx7760	100376
xx7762	116162
xx7764	000002
xx7766	xx7400
xx7770	005267
xx7772	177756
xx7774	000765
xx7776	yyyyyy

xx represents the highest available memory bank. For example, the first location of the loader would be one of the following, depending on memory size, and xx in all subsequent locations would be the same as the first.

Location	Memory Bank	Memory Size
017744	0	4K
037744	1	8K
057744	2	12K
077744	3	16K
117744	4	20K
137744	5	24K
157744	6	28K

The contents of location xx7776 (yyyyyy) in the instruction column above should contain the device status register address of the papertape reader to be used when loading the bootstrap formatted tapes specified as follows:

Teletype Paper Tape Reader -- 177560
High-speed Paper Tape Reader -- 177550

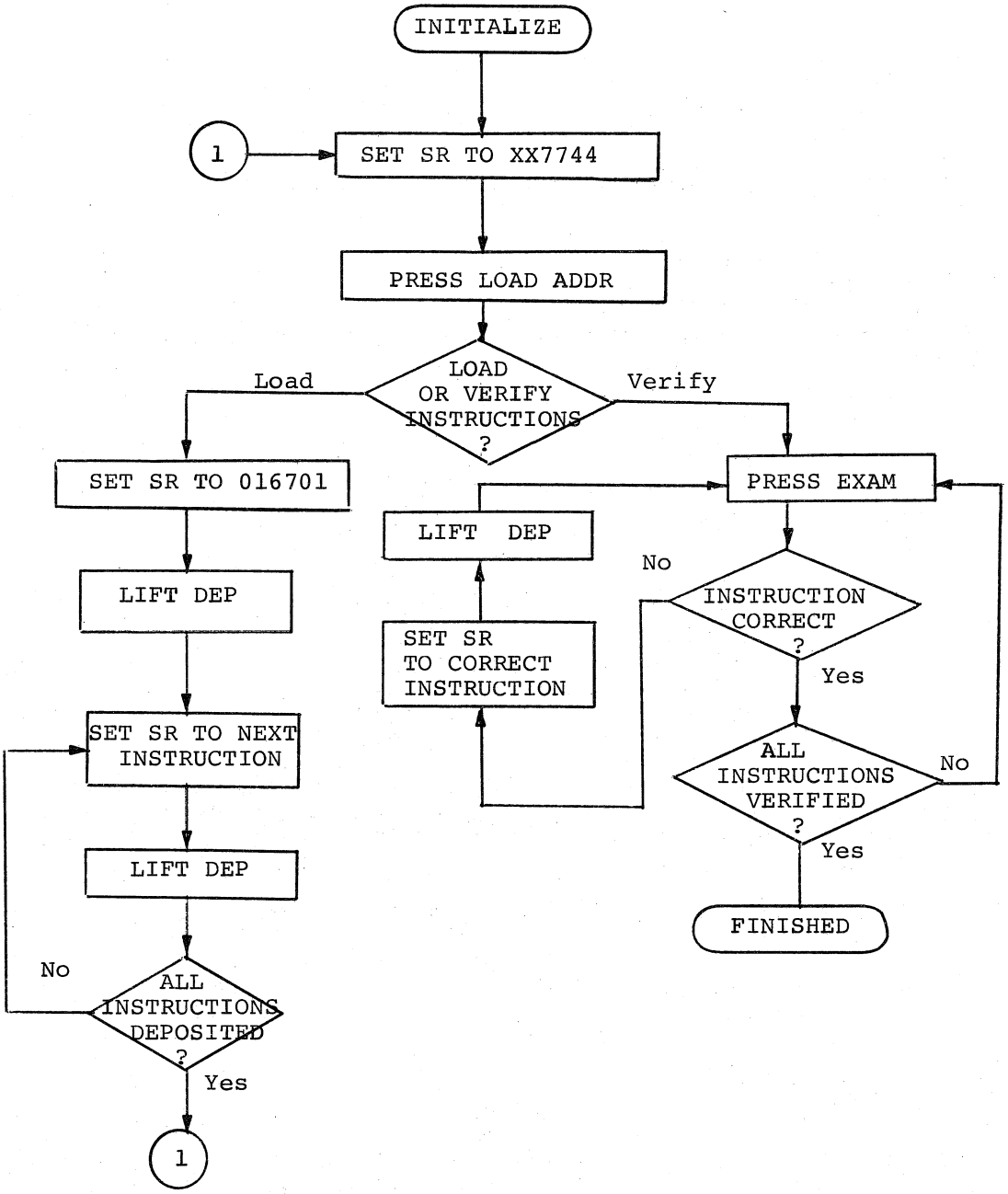


Figure A-1 Loading and Verifying the Bootstrap Loader

A.1.2 Loading with the Bootstrap Loader

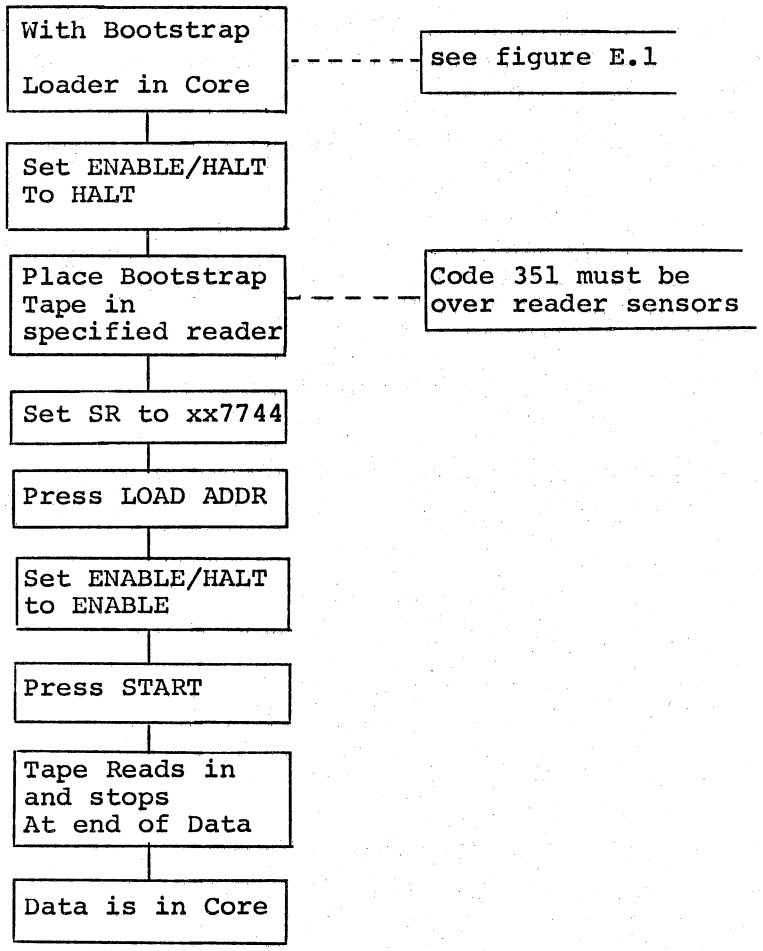


Figure A-2. Loading Bootstrap Tapes into Core

A.2 THE ABSOLUTE LOADER

A.2.1 Loading the Absolute Loader

The Bootstrap Loader is used to load the Absolute Loader into core. (See Figure A-2.) The Absolute Loader occupies locations xx7474 through xx7743, and its starting address is xx7500.

A.2.2 Loading with the Absolute Loader

When using the Absolute Loader, there are three types of loads available: normal, relocated to specific address, and continued relocation.

Optional switch register settings for the three types of loads are listed below.

Type of Load	Switch Register Bits 1-14	Switch Register Bit 0
Normal	(ignored)	0
Relocated - continue loading where left off	0	1
Relocated - load in specified area of core	nnnnn (specified address)	1

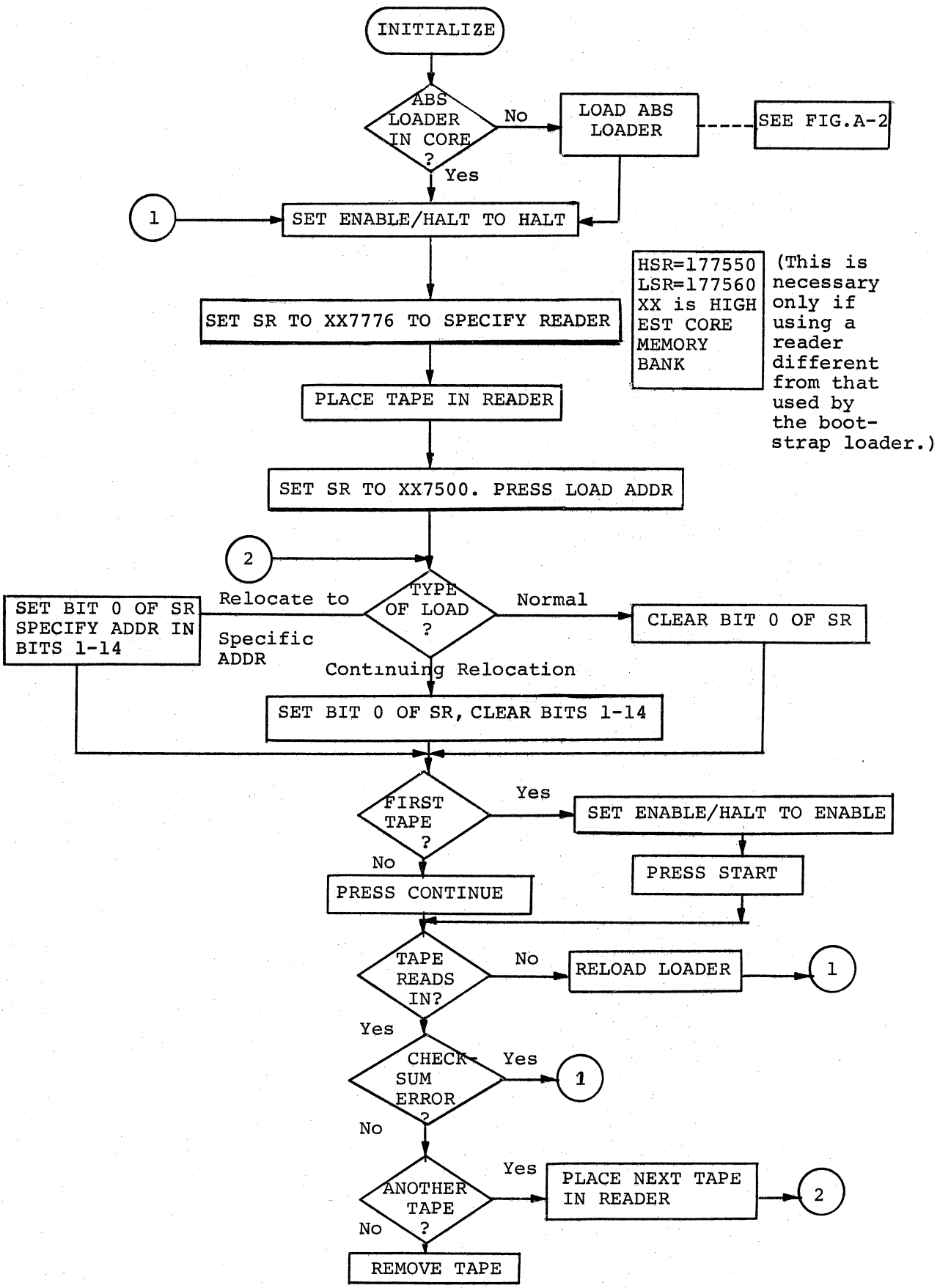


Figure A-3 Loading with the Absolute Loader

APPENDIX B

USING THE PAL-11S ASSEMBLER

Run the assembler according to the directions in Section B.1. If another program is being assembled along with FPMP-11, it should be read before the FPMP-11 package. This other program must be followed by a .EOT instruction and must not define any FPMP-11 labels or conditional switches. After any user program being assembled with FPMP-11 has been read, the assembler prints EOF? and pauses. Place the switch setting tape previously created (refer to section 3.5.1.1) in the reader and type the RETURN key. At the end of this tape the assembler again prints EOF? Place the first source tape of the FPMP-11 package in the reader and type the RETURN key. After this source tape has been read and the assembler prints EOF?, place the next source tape in sequence in the reader and type RETURN. Repeat this sequence until all source tapes have been read. When the last tape has been read, the assembler proceeds to Pass 2. All of the tapes must be read again using the same procedure as above. The assembler produces the FPMP-11 object module on the binary output device specified in the initial assembler dialogue.

B.1 ASSEMBLER OPERATING PROCEDURES

Loading:	Use Absolute Loader. The start address of the Loader must be in the console switches.
Storage Requirements:	PAL-11S uses 8K memory.
Starting:	Immediately upon loading, PAL-11S is in control and initiates dialogue.

Initial Dialogue:

Printout	Inquiry
*S	What is the input device of the source symbolic tape?
*B	What is the output device of the binary object tape?
*L	What is the output device of the assembly listing?
*T	What is the output device of the symbol table?

Each of these questions may be answered by any one of the following characters:

Character	Answer Indicated
T	Teleprinter keyboard
L	Low-speed reader or punch
H	High-speed reader or punch
P	Line Printer

Each of these answers may be followed by the other characters indicating options:

Option Typed	Function to be performed
/1	on pass 1
/2	on pass 2
/3	on pass 3
/E	errors to be listed on the Teleprinter on the same pass (meaningful only for *B or *L).

Each answer is terminated by typing the RETURN key. Answering with a RETURN alone deletes the function.

Dialogue During Assembly:

Printout	Response
EOF ?	Place next tape in reader and type RETURN. A .END statement may be forced by typing E followed by RETURN.
END ?	Start next pass by placing first tape in reader and typing RETURN.
EOM ?	If the end-of-medium is on the listing device, the device may be readied and the assembly may be continued by typing RETURN. If the end-of-medium is on the binary device, the assembler will discontinue the assembly and restart itself.

Restarting:

Type CTRL/P. The initial dialogue will be started again.

For more detailed information on the PAL-11S Assembler, refer to the PDP-11 PAL-11S Assembler and LINK-11S Linker Programmer's Manual (DEC-11-YRWB-D).

B.2 ASSEMBLER ERROR CODES

Error Code	Meaning
A	Addressing error. An address within the instruction is incorrect. Also includes relocation errors.
B	Bounding error. Instructions or word data are being assembled at an odd address in memory.
D	Doubly-defined symbol referenced. Reference was made to a symbol which is defined more than once.
I	Illegal character detected. Illegal characters which are also non-printing are replaced by a ? on the listing.
L	Line buffer overflow. All extra characters beyond 72 are ignored.
M	Multiple definition of a label. A label was encountered which was equivalent (in the first six characters) to a previously encountered label.
N	Number containing an 8 or 9 was not terminated by a decimal point.
P	Phase error. A label's definition or value varies from one pass to another.
Q	Questionable syntax. There are missing arguments or the instruction scan was not completed, or a carriage return was not followed by a linefeed or form feed.
R	Register-type error. An invalid use of or reference to a register has been made.
S	Symbol table overflow. When the quantity of user-defined symbols exceeds the allocated space available in the user's symbol table, the assembler outputs the current source line with the S error code, then returns to the command string interpreter to await the next command string to be typed.
T	Truncation error. More than the allotted number of bits were input so the leftmost bits were truncated. T error does not occur for the result of an expression.
U	Undefined symbol. An undefined symbol was entered during the evaluation of an expression. Relative to the expression, the undefined symbol is assigned a value of zero.

APPENDIX C

USING LINK-11S

C.1 LOADING AND COMMAND STRING

The Linker is loaded by the Absolute Loader and is self-starting. It uses a simple command dialogue which allows the object module, load module and load map devices to be specified. During pass 1 and pass 2, the Linker asks for each object module individually.

For illustration purposes, the non-printing characters carriage return, line feed and space are represented as <CR>, <LF> and <SPACE>.

Operation begins by the linker typing its name and version. This is followed by the input option printed as *I<SPACE>. The responses are:

<CR>	Read object module from HSR.
H<CR>	Read object module from HSR.
L<CR>	Read object module from LSR.

The input option is followed by the output option *O<SPACE>. The responses are:

<CR>	Punch load module on HSP.
H<CR>	Punch load module on HSP.
L<CR>	Punch load module on LSP.

LINK-11 asks if a load map is desired by typing *M<SPACE>. The legal responses are <CR> for no map, T<CR> or H<CR> or P<CR> for a map on the teleprinter, high-speed punch, or line printer, respectively.

The next two options concern the placement of the relocated object program in memory. The standard version of the Linker assumes it is linking for an 8K machine. It relocates the program such that it is as high as possible in 8K but leaves room for the Absolute and Boot Loaders. (These assumed values may be changed by altering parameters HGHMEM (highest legal memory address +1) and ALODSZ (number of bytes allocated for Absolute Loader and Boot Loader) and reassembling the linker.) The *T and *B options control the relocation of a program. After the option *T<SPACE> is printed, respond as follows:

<CR>	Relocate so that program is up against the current top of memory. If the top has not been changed, then the top is the assembled-in top (HGHMEM-ALODSZ). The standard assumption is 16384.-112.=16272 (37460_8).
n<CR>	n is an octal number (unsigned) which defines a new top address.

If a new top is specified, the *B option is suppressed.

After the option *B<SPACE> is printed respond as follows:

<CR>	Use current top of memory.
n<CR>	n is an unsigned octal number which defines the bottom address of the program. That is, a new top of memory is calculated so that the bottom of the program corresponds with n.

Once a top of memory has been calculated (by *T or *B), that value is used until it is changed.

LINK-11 indicates the start of pass one by printing PASS 1. The input is requested by the Linker, one tape at a time by printing *<SPACE>. The legal responses are:

<CR>	Read a tape and request more input.
U<CR>	List all undefined globals on the teleprinter and request more input.
E<CR>	End of input. If there are undefined globals, list them on the teleprinter and request more input. Otherwise print the load map if requested, and enter pass 2.
C<CR>	End of input. Assign 0 to any undefined globals, print the load map (if requested), and enter pass 2.

The Linker indicates the start of pass 2 by printing PASS 2 and requests each input tape as in pass 1.

A <CR> is the only useful response to an asterisk (*) on pass 2. The modules must be read on pass 2 in the same order as pass 1. When the last module has been read, the Linker automatically finishes the load module and restarts itself.

Leader and trailer are punched on the load module.

If the low-speed punch (LSP) is being used for the load module output, it should be turned on before pass 2 begins, i.e., turn it on before typing E<CR> or C<CR>. The echo of these characters (and the load map) if printed on the Teletype are punched on the load module but may be easily removed since leader is punched on the load module. The LSP can also be turned on while leader is being punched (after the linker has typed PASS 2) to keep the load map, etc., from being punched onto the tape.

NOTE

On all command string options, except for *T and *B, the linker examines only the last character typed preceding the carriage return. Thus,

ABCDEFGH<CR>

is equivalent to H<CR>.

C.2 ERROR PROCEDURE AND MESSAGES

C.2.1 Restarting

CTRL/P is used for two purposes by LINK11-S. If a CTRL/P is typed while a load map is being printed, the load map is aborted and the Linker continues. CTRL/P typed at any other time causes the Linker to restart itself.

C.2.2 Non-Fatal Errors

Message	Explanation
?MODULE NAME xxxxxx NOT UNIQUE	Non-unique object module name - this error is detected during pass 1 and results in an error message and the module is rejected. The Linker will then ask for more input.
?MAP DEVICE EOM. TYPE <CR> TO CONTINUE	Load map device EOM - this error allows the user an option to fix the device and continue or abort the map listing. Any response, terminated by <CR> or <LF> causes the Linker to continue. A CTRL/P causes the map be to aborted.

Message	Explanation
?BYTE RELOC ERROR AT ABS ADDRESS xxxxxx.	A byte relocation error - the Linker tries to relocate and link byte quantities. However, relocation usually fails and linking may fail. Failure is defined as the high byte of the relocated value (or the linked value) not being all zero. In such a case, the value is truncated to 8 bits. The Linker automatically continues.
?LOAD xxxxxx NEXT	If the object modules are not read in the same order on pass 2 as pass 1, the Linker indicates which module should be loaded next by typing this message and asking for more input.
?xxxxxx MULTIPLY DEFINED BY MODULE xxxxxx.	Multiply-defined globals were discovered, during pass 1. The second definition is ignored and the Linker continues.

C.2.3 Fatal Errors

The Linker restarts after any of the following:

Message	Explanations
?SYMBOL TABLE OVERFLOW - MODULE xxxxxx, SYMBOL xxxxxx	Symbol Table overflow.
?SYSTEM ERROR xx	System Error. Where xx is an identifying number as follows:

Number	Meaning
01	Unrecognized symbol table entry found.
02	A relocation directory references a global name which cannot be found in the symbol table.

Number	Meaning
03	A relocation directory contains a location counter modification command which is not last.
04	Object module does not start with a GSD.
05	The first entry in the GSD is not the module name.
06	An RLD references a section name which cannot be found.
07	The TRA specification references a nonexistent module name.
08	The TRA specification references a non-existent section name.
09	An internal jump table index is out of range.
10	A checksum error occurred on the object module.
11	An object module binary block is too big (more than 64 words of data).
12	A device error occurred on the load module output device.

All system errors except for numbers 10 and 12 indicate a program failure either in the Linker or the program which generated the object module. Error 05 can occur if a tape is read which is not an object module.

C.2.4 Error Halts

LINK-11 loads all of its unused TRAP vectors with the code:

```
.WORD     .+2,HALT
```

so that if the TRAP occurs, the processor halts in the second word of the vector. The address of the halt, displayed in the console lights, therefore indicates the cause of the halt.

Address of HALT (octal)	Meaning
12	Reserved instruction executed.
16	Trace TRAP occurred.
26	Power fail TRAP.
32	EMT executed.

A halt at address 40 indicates an IOXLPT detected error. R0 (displayed in the console lights) contains an identfying code:

Code in R0	Meaning
0	Illegal memory reference, SP overflow or illegal instruction.
1	Illegal IOX command.
2	Slot number out of range.
3	Device number illegal.
4	Referenced slot not INITed.
5	Illegal data mode.

IOXLPT also sets R1 as follows:

If the error code is 0, R1 contains the PC at the time of the error.

If the error code is 1-5, R1 points to some element in the IOT argument list or to the instruction following the argument list, depending on whether IOXLPT has finished decoding all the arguments when it detects the error.

APPENDIX D

SUMMARY OF FPMP-11 ROUTINES

This appendix lists all the global entry points of FPMP-11 and provides a brief description of the purpose of each. Sections D.1 and D.2 are for reference when it is desired to call FPMP-11 routines directly (i.e., without the use of the TRAP handler). Entry names preceded by an octal number can be referenced via the TRAP handler. The number is the "routine number" referred to throughout this manual. If the number is enclosed in parentheses, the routine cannot be accessed by the present TRAP handler, but has been assigned a number for future use.

Examples of the calling conventions are:

```
POLISH MODE:    .
                .
                .
                JSR R4,$POLSH       ;enter Polish mode
                $subr1              ;call desired subroutines
                $subr2
                .
                .
                .
                $subrn              ;call last subroutine desired
                .WORD   .+2         ;leave Polish mode.
                .
                .
                .
-------------------------------------------------------------
J5RR:           .
                .
                .
                JSR R5,subr         ;call desired subroutine
                BR      XX
                .WORD   arg1        ;subroutine argument address
                .WORD   arg2
                .
                .
                .
                .WORD   argn        ;last argument
     XX:        .                   ;return point
                .
                .
-------------------------------------------------------------
```

```
JPC:        .
            .
            .
            push args onto stack
            JSR PC,subr
            .
            .
            .
```

D.1 OTS ROUTINES

These are the routines taken from the FORTRAN operating time system. The codes used in the following table are:

S = Routine is included in the standard single precision (2-word) package.
D = Routine is included in the standard double precision (4-word) package.
SD = Routine is included in both standard packages.

Octal codes shown in parentheses are not yet implemented.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
$ADD	14	D	2	Polish	The double precision add routine. Adds the top stack item (4 words) to the second item (4 words) and leaves the four word sum in their place.
$ADR	12	S	2	Polish	The single precision add routine. Same as $ADD except it uses 2 word numbers.
AINT	26	S	1	J5RR	Returns sign of argument * greatest real integer = absolute value of the argument in R0,R1.
ALOG	53	S	1	J5RR	Calculates natural logarithm of its single argument and returns a two word result in R0,R1.
ALOG10	54	S	1	J5RR	Same as ALOG, except calculates base-10 logarithm.
ATAN	42	S	1	J5RR	Returns the arctangent of its argument in R0,R1.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
ATAN2	(43)	S	2	J5RR	Returns ARCTAN(ARG1/ARG2) in R0,R1.
$CMD	16	D	2	Polish	Compares top 4 word items on the stack, flushes the two items, and returns the following condition codes: 4(SP) @SP N=1,Z=0 4(SP) = @SP N=0,Z=1 4(SP) @SP N=0,Z=0
$CMR	17	S	2	Polish	Same as $CMD except it is for 2 word arguments.
COS	37	S	1	J5RR	Single precision version of DCOS.
DATAN	44	D	1	J5RR	Double precision version of ATAN.
DATAN2	(45)	D	2	J5RR	Double precision version of ATAN2.
DBLE	(34)		1	J5RR	Returns in R0-R3 the double precision equivalent of the single precision (two word) argument.
$DCI	(57)	SD	4	JPC	ASCII to double conversion. Calling sequence: Push address of start of ASCII field. Push length of ASCII field in bytes. Push format scale D (from W.D) position of assumed decimal point (see FORTRAN manual). Push P format scale (see FORTRAN manual). JSR PC,$DCI. Returns 4 word result on top of stack.
$DCO	(61)	SD	5	JPC	Double precision to ASCII conversion. Calling sequence: Push address of start of ASCII field. Push length in bytes of ASCII field (W part of W.D) Push D part of W.D (position of decimal point). Push P scale. Push 4 word value to be converted, lowest order word first. JSR PC,$DCO.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
DCOS	41	D	1	J5RR	Calculates the cosine of its double precision argument and returns the double precision result in R0-R3.
DEXP	52	D	1	J5RR	Calculates the exponential of its double precision argument, and returns the double precision result in R0-R3.
$DI	(11)	SD		Polish	Converts double precision number on the top of the stack to integer. Leaves result on stack.
$DINT	(76)	D	1	Polish	OTS internal function to find the integer part of a double precision number.
DLOG	55	D	1	J5RR	Double precision (4 word) version of ALOG.
DLOG10	56	D	1	J5RR	Double precision (4 word) version of ALOG10.
$DR	(6)		1	Polish	Replaces the double precision item at the top of the stack with its two word, rounded form.
DSIN	40	D	1	J5RR	Calculates the sine of its double precision arg. and returns the double precision result in R0-R3.
DSQRT	47	D	1	J5RR	Calculates the square root of its double precision arg. and returns the double precision result in R0-R3.
$DVD	23	D	2	Polish	The double precision division routine. Divides the second 4-word item on the stack by the top item and leaves the quotient in their place.
$DVI	(24)		2	Polish	The integer division routine. Calculates 2(SP)/@SP and returns the integer quotient on the top of the stack.
$DVR	25	S	2	Polish	The single precision division routine. Same as $DVD, but for 2 word floating point numbers.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
$ECO	(62)	SD	5	JPC	Single precision to ASCII conversion according to E format. Same calling sequence as $DCO except that a 2-word value is to be converted.
EXP	51	S	1	J5RR	Single precision version of DEXP. Returns result in R0,R1.
$FCALL	-	S			Internal OTS routine.
$FCO	(64)	SD	5	JPC	Same as $ECO except uses F format conversion.
FLOAT	(32)		1	J5RR	Returns in R0-R1, the real equivalent of its integer argument.
$GCO	(63)	SD	5	JPC	Same as $ECO except uses G format conversion.
$ICI	(65)		2	JPC	ASCII to integer conversion. Calling sequence: Push address of start of ASCII field. Push length in bytes of ASCII field. JSR PC,$ICI Returns with integer result on top of stack.
$ICO	(67)		3	JPC	Integer to ASCII conversion. Calling sequence: Push address of ASCII field. Push length in bytes of ASCII field. Push integer value to be converted. JSR PC,$ICO Error will return with C bit set on. R0-R3 destroyed.
IDINT	(31)		1	J5RR	Returns sign of arg * greatest integer <= |arg| in R0. Arg is double precision.
$ID	(5)	SD	1	Polish	Convert full word argument on the top of the stack to double precision and return result as top 4-words of stack.
IFIX	(35)		1	J5RR	Returns the truncated and fixed real argument in R0.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
INT	(30)		1	J5RR	Same as IDINT for single precision args.
$INTR	(27)	S	1	Polish	Same function as AINT, but called in Polish mode with argument and returns result on the stack.
$IR	(4)	SD	1	Polish	Convert full word argument on the top of the stack to single precision and return result as top 2-words of stack.
$MLD	22	D	2	Polish	Double precision multiply. Replaces the top two doubles on the stack with their product.
$MLI	(20)		2	Polish	Integer multiply. Replaces the top 2 integers on the stack with their full word product.
$MLR	21	S	2	Polish	Single precision multiply. Replaces the top two singles on the stack with their product.
$NGD	(3)	SD	2	Polish	Negate the double precision number on the top of the stack.
$NGI	(1)	SD	1	Polish	Negate the integer on the top of the stack.
$NGR	(2)	SD	1	Polish	Negate the single precision number on the top of the stack.
$OCI	(66)		2	JPC	ASCII to octal conversion. Same call as $ICI.
$OCO	(70)		3	JPC	Octal to ASCII conversion. Same call as $ICO.
$POLSH	-	SD	-	-	Called whenever it is desired to enter Polish mode from normal in-line code. It must be called via a JSR R4,$POLSH.
$POPR3	-	D	-	Polish	Internal routine to pop 2-words from the stack and place them into R0,R1.
$POPR4	-	D	-	Polish	Internal routine to pop 4-words from the stack and place them in R0-R3.

NAME	OCTAL CODE	PKG	# OF ARGU	MODE	DESCRIPTION
$POPR5	-	D	-	Polish	Internal routine to pop 4-words from the stack and place them in registers R0-R3.
$PSHR1	-	SD		Polish	Internal routine to push the contents of R0 onto the stack.
$PSHR2	-	SD	-	Polish	Same as $PSHR1.
$PSHR3	-	SD	-	Polish	Push R0,R1 onto stack.
$PSHR4	-	SD	-	Polish	Push R0-R3 onto stack.
$PSHR5	-	SD	-	Polish	Same as $PSHR4.
$RCI	(60)	SD	4	JPC	ASCII to single precision conversion. Same calling sequence as $DCI. Returns 2-word result on top of stack.
$RD	(7)			Polish	Converts the single precision number on the top of the stack to double precision format. Leaves result on stack.
$RI	(10)	SD		Polish	Converts single precision number on the top of the stack to integer. Leaves result on stack.
$SBD	15	D		Polish	The double precision subtract routine. Subtracts the double precision number on the top of the stack from the second double precision number on the stack and leaves the result on the top of the stack in their place.
$SBR	13	S		Polish	Same as $SBD but for single precision.
SIN	36	S	1	J5RR	Single precision version of DSIN.
SNGL	(33)		1	J5RR	Rounds double precision argument to single precision. Returns result in R0, R1.
SQRT	46	S	1	J5RR	Single precision version of DSQRT.
TANH	50	S	1	J5RR	Single precision hyperbolic tangent function. Returns (EXP(2*ARG)-1) / (EXP(2*ARG)+1) in R0,R1.

D.2 NON-OTS ROUTINES

These routines are written especially for FPMP-11 and should not be called directly by the user.

NAME	OCTAL CODE	PKG	DESCRIPTION
$ERR	-	SD	Internal error handler.
$ERRA	-	SD	Similar to $ERR.
$LDR	71	S	Load FLAC, single precision.
$LDD	72	D	Load FLAC, double precision.
$STR	73	S	Store FLAC, single precision.
$STD	74	D	Store FLAC, double precision.
TRAPH	-	SD	The TRAP handler routines and tables.

D.3 ROUTINES ACCESSED VIA TRAP HANDLER

The following is a table of the FPMP-11 routines which can be accessed via TRAPH, the trap handler. Each routine name (entry point) is preceded by its TRAP code number to be used to access it, and followed by a brief description of its operation when called via the TRAP handler. Those entries which are preceded by an asterisk (*) perform operations only on the FLAC, and address no operands. For example, a TRAP call to the single precision square root routine can be coded as follows:

```
        .
        .
        .
      TRAP   46
        .
        .
        .
```

The net effect of the above TRAP instruction is to replace the contents of the FLAC with its square root and then set the condition codes to reflect the result. Note that since the FLAC is implicitly addressed in this instruction, the TRAP call supplies no other address. For such a TRAP call, the addressing mode bits (bits 6 and 7 of the TRAP instruction) are ignored.

All entries not marked by an asterisk require an operand when called. The operand is addressed in one of the 4 addressing modes explained in section 3.1.1. The addressing mode is specified in bit 6-7 of the TRAP instruction.

("Operand" is the contents of the location addressed in the TRAP call.)

	OCTAL CODE	NAME	DESCRIPTION
	14	$ADD	Double precision addition routine. Adds operand to the FLAC. Assumes 4-word operand.
	12	$ADR	Single precision addition routine. Adds operand to the FLAC. Assumes 2-word operand.
*	26	AINT	Replaces contents of the FLAC by its integer part. SIGN(FLAC) * greatest integer <= |FLAC|. Assumes 2-word argument in FLAC.
*	53	ALOG	Replaces contents of the FLAC by its natural logarithm. Assumes 2-word argument in FLAC.
*	54	ALOG10	Same as ALOG, except calculates base-10 log.
*	42	ATAN	Replaces contents of the FLAC by its arctangent. Assumes 2-word argument in FLAC.
	16	$CMD	Compares operand to the contents of the FLAC, and returns the following condition codes. FLAC<operand, N=1,Z=0 FLAC=operand, N=0,Z=1 FLAC>operand, N=0,Z=0 Assumes 4-word operands.
	17	$CMR	Same as $CMD, but for 2-word operands.
*	37	COS	Same as DCOS, but for 2-word argument.
*	44	DATAN	Same as ATAN, but for 4-word argument.
*	52	DEXP	Replaces the contents of the FLAC by its exponential. Assumes 4-word argument in the FLAC.
*	55	DLOG	Same as ALOG, but for 4-word argument.
*	56	DLOG10	Same as ALOG10, but for 4-word argument.
*	41	DCOS	Replaces the contents of the FLAC by its cosine. Assumes 4-word argument in the FLAC.

	OCTAL CODE	NAME	DESCRIPTION
*	40	DSIN	Same as DCOS, but calculates sine instead of cosine.
*	47	DSQRT	Replaces the contents of the FLAC by its square root. Assumes 4-word argument in the FLAC.
	23	$DVD	Double precision division routine. Divides the FLAC by the operand and stores the result in the FLAC. Assumes 4-word operands.
	25	$DVR	Same as $DVD, but for 2-word operands.
*	51	EXP	Same as DEXP, but for 2-word argument.
	72	$LDD	Same as $LDR, but assumes 4-word operand.
	71	$LDR	Replaces the contents of the FLAC by the operand. Assumes 2-word operand.
	22	MLD	Double precision multiplication routine. Multiplies the contents of the FLAC by the operand and stores the result in the FLAC. Assumes 4-word operands.
	21	$MLR	Same as $MLD, but for 2-word operands.
	15	$SBD	The double precision subtraction routine. Subtracts the operand from the contents of the FLAC. Assumes a 4-word operand.
	13	$SBR	Same as $SBD, but for 2-word operand.
*	36	SIN	Same as DSIN, but for 2-word argument.
*	46	SQRT	Same as DSQRT, but for 2-word argument.
	73	$STR	Stores the contents of the FLAC into the operand location. The contents of the FLAC are unchanged.
	74	$STD	Same as $STR, but assumes 4-word operand location.
*	50	TANH	Replaces the contents of the FLAC by its hyperbolic tangent. Assumes 2-word argument.

APPENDIX E
FPMP-11 SOURCE LISTING

This source listing of FPMP-11 is included for documentation of the logic only. The sources provided to users do not have comments because of size restrictions.

```
1           000001          SINGLE=1
2           000001          DOUBLE=1
3           000001          CND$7=1
4           000001          CND$12=1
5           000001          CND$17=1
6           000001          CND$22=1
7           000001          CND$23=1
8           000001          CND$24=1
9           000001          CND$25=1
10          000001          CND$26=1
11          000001          CND$29=1
12          000001          CND$34=1
13          000001          CND$36=1
14                          .EOT
15
```

```
1                ;PRODUCT CODE              DEC-11-NFPMA-A-LA
2
3                ;COMPUTER                  PDP-11
4
5                ;CONFIGURATION             PAPER TAPE CONFIGURATION IS MINIMUM
6                ;                          8192 WORDS MEMORY
7
8                ;SOFTWARE REQUIREMENTS     PAL-11S (OR MACRO-11)
9                ;                          LINK-11S (OR LINK-11)
10
11               ;PROGRAM NAME              FPMP-11
12
13               ;VERSION                   VERSION LEVEL  1
14               ;                          PATCH LEVEL    A
15
16               ;DESCRIPTION               FLOATING POINT MATH PACKAGE
17               ;                          PLUS TRAP HANDLER
18               ;                          (FLOATING POINT SUBROUTINES TAKEN FROM
19               ;                          DOS-11 FORTRAN IV OTS)
20
21               ;AUTHOR                    E. PETERS (TRAP HANDLER & PACKAGE
22               ;                                     INTEGRATION)
23
24               ;DATE                      AUGUST, 1972
25
26               ;                          COPYRIGHT 1972, DIGITAL EQUIPMENT CORP.,
27               ;                          MAYNARD, MASSACHUSETTS 01754
```

```
1          000000'          .CSECT
2
3                     ;     CONDITIONALS TO GENERATE THE STANDARD PACKAGES,
4                           .IFDF   SINGLE; SINGLE PRECISION PACKAGE?
5          000001           CND$2=1          ;$ADR,$SBR
6          000001           CND$3=1          ;ALOG,ALOG10
7          000001           CND$4=1          ;AINT
8          000001           CND$6=1          ;$CMR
9          000001           CND$18=1         ;$DVR
10         000001           CND$20=1         ;EXP
11         000001           CND$30=1         ;$MLR
12         000001           CND$37=1         ;SIN,COS
13         000001           CND$38=1         ;TANH
14         000001           CND$39=1         ;ATAN,ATAN2
15         000001           CND$41=1         ;SQRT
16         000001           CND$44=1
17         000001           CND$46=1
18                          .ENDC
19
20                          .IFDF   DOUBLE; DOUBLE PRECISION PACKAGE?
21         000001           CND$1=1          ;$ADD,$SBD
22         000001           CND$5=1          ;$CMD
23         000001           CND$10=1         ;DLOG,DLOG10
24         000001           CND$13=1         ;DSIN,DCOS
25         000001           CND$14=1         ;DSQRT
26         000001           CND$15=1         ;DATAN,DATAN2
27         000001           CND$16=1         ;$DVD
28         000001           CND$19=1         ;DEXP
29         000001           CND$28=1         ;$MLD
30         000001           CND$45=1         ;$LDD
31         000001           CND$47=1         ;$STD
32                          .ENDC
33
34                          .IFDF   SINGLE!DOUBLE
35         000001           CND$8=1          ;$DCI,$RCI
36         000001           CND$9=1          ;$ECO,$FCO,$GCO,$DCO
37         000001           CND$31=1         ;$NGI,$NGR,$NGD
38         000001           CND$42=1         ;TRAPH
39                          .ENDC
```

```
1                       .IFDF   CND$38; TANH?
2       000001          CND$2=1         ;$ADR,$SBR
3       000001          CND$18=1        ;$DVR
4       000001          CND$20=1        ;EXP
5       000001          CND$21=1        ;$FCALL
6       000001          CND$30=1        ;$MLR
7       000001          CND$32=1        ;$PSHR3
8                       .ENDC
9
10                      .IFNDF  FPU
11                      .IFDF   CND$3!CND$20!CND$37!CND$39
12      000001          CND$2=1         ;$ADR,$SBR
13      000001          CND$18=1        ;$DVR
14      000001          CND$30=1        ;$MLR
15                      .IFDF   CND$37; SIN,COS?
16      000001          CND$4=1         ;$INTR
17                      .ENDC
18                      .ENDC
19                      .IFDF   CND$10!CND$13!CND$15!CND$19
20      000001          CND$1=1         ;$ADD,$SBD
21      000001          CND$16=1        ;$DVD
22      000001          CND$28=1        ;$MLD
23      000001          CND$33=1        ;$POPR4
24                      .IFDF   CND$13; DSIN,DCOS?
25      000001          CND$11=1        ;$DINT
26                      .ENDC
27                      .ENDC
28                      .IFDF   CND$3!CND$10;   ALOG OR DLOG?
29      000001          CND$27=1        ;$IR,$ID
30                      .ENDC
31                      .IFDF   CND$19!CND$20;  EXP OR DEXP?
32      000001          CND$27=1        ;$IR,$ID
33      000001          CND$35=1        ;$RI,$DI
34                      .ENDC
35                      .IFDF   CND$14; DSQRT?
36      000001          CND$1=1         ;$ADD
37      000001          CND$16=1        ;$DVD
38                      .ENDC
39                      .IFDF   CND$41; SQRT?
40      000001          CND$2=1         ;$ADR
41      000001          CND$18=1        ;$DVR
42                      .ENDC
43                      .ENDC
44
45                      .IFDF   CND$23; FLOAT?
46      000001          CND$27=1        ;$IR,$ID
47      000001          CND$33=1        ;$POPR3
48                      .ENDC
49                      .IFDF   CND$22!CND$26;  IFIX, INT, OR IDINT?
50      000001          CND$35=1        ;$RI,$DI
51                      .ENDC
52                      .IFDF   CND$39; ATAN OR ATAN2?
53      000001          CND$33=1        ;POPR3
54                      .ENDC
```

```
                        .TITLE  TRAP02
                        .IFDF   CND$42
                        .GLOBL  TRAPH,$ERRA
                ;       THE FPMP-11 TRAP HANDLER
          000000        R0=%0
          000001        R1=%1
          000002        R2=%2
          000003        R3=%3
          000004        R4=%4
          000005        R5=%5
          000006        SP=%6
          000007        PC=%7

00000 042766 TRAPH: BIC     #17,2(SP);      CLEAR ALL USER COND CODES
      000017
      000002
00006 005046        CLR     -(SP);          SPACE FOR ADDR MODE
00010 010546        MOV     R5,-(SP);       SAVE THE REGISTERS
00012 010446        MOV     R4,-(SP)
00014 010346        MOV     R3,-(SP)
00016 010246        MOV     R2,-(SP)
00020 010146        MOV     R1,-(SP)
00022 010046        MOV     R0,-(SP)
00024 016603        MOV     20(SP),R3;      GET USER'S STATUS WORD
      000020
00030 042703        BIC     #20,R3;         CLEAR T-BIT FOR US
      000020
00034 010337        MOV     R3,@#177776;    ESTABLISH AS CURRENT STATUS
      177776
00040 016601        MOV     16(SP),R1;      GET USER'S PC
      000016
00044 010105        MOV     R1,R5;          COPY USER'S PC
00046 014104        MOV     -(R1),R4;       PICK UP TRAP INSTRUCTION
00050 010403        MOV     R4,R3;          COPY
00052 042704        BIC     #177700,R4;     CALC TABLE INDEX
      177700
00056 006304        ASL     R4;             TIMES TWO
00060 016404        MOV     TBL$42(R4),R4;  GET TABLE ENTRY
      000500!
00064 001556        BEQ     ERR$42;         ERROR: NO ENTRY IN TABLE
00066 010402        MOV     R4,R2;          COPY TABLE ENTRY
00070 042702        BIC     #140000,R2;     CLEAR MODE BITS
      140000
00074 060702        ADD     PC,R2;          RELOCATE ROUTINE ADDRESS
00076 032704 PT$42: BIT     #40000,R4;      ADDRESSING REQUIRED
      040000
00102 001514        BEQ     NAD$42;         BRANCH IF NONE REQUIRED
00104 106103        ROLB    R3;             TEST OPERAND ADDRESS MODE
00106 100122        BPL     PLM$42;         BRANCH IF BIT 6 EQUALS 0
00110 103004        BCC     STK$42;         BRANCH IF @R0 MODE
                ;       RELATIVE MODE
00112 010500        MOV     R5,R0;          COPY USER'S PC
00114 062500        ADD     (R5)+,R0;       CALC ACTUAL OPERAND ADDRESS
00116 010566 UPC$42: MOV    R5,16(SP);      UPDATE USER'S PC
      000016
00122 012705 STK$42: MOV    #FAC$42+6,R5;   ADDRESS OF FLAC
      000440!
```

```
46 00126 005704          TST     R4;             SINGLE OR DOUBLE?
47 00130 002403          BLT     ST4$42;         BRANCH IF DOUBLE
48 00132 005015          CLR     @R5;            CLEAR LAST 2 WORDS OF FLAC
49 00134 005045          CLR     -(R5)
50 00136 005725          TST     (R5)+;          INCR R5
51 00140 011546 ST4$42:  MOV     @R5,-(SP);      PUST THE FLAC
52 00142 014546          MOV     -(R5),-(SP)
53 00144 014546          MOV     -(R5),-(SP)
54 00146 014546          MOV     -(R5),-(SP)
55 00150 005704          TST     R4;             SINGLE OR DOUBLE?
56 00152 002402          BLT     ST6$42;         BRANCH IF DOUBLE
57 00154 022020          CMP     (R0)+,(R0)+;    INCR R0 BY 4
58 00156 000404          BR      OT2$42
59 00160 062700 ST6$42:  ADD     #8.,R0
           000010
60 00164 014046          MOV     -(R0),-(SP);    PUSH OPERAND
61 00166 014046          MOV     -(R0),-(SP)
62 00170 014046 OT2$42:  MOV     -(R0),-(SP)
63 00172 014046          MOV     -(R0),-(SP)
64
65                       ;       CALL ROUTINE IN POLISH MODE.
66                       ;       THIS IS NOT A STANDARD POLISH CALL
67                       ;       IN ORDER TO REDUCE OVERHEAD.
68 00174 012704          MOV     #ADR$42,R4;     ADDRESS OF RETURN ADDR
           000202'
69 00200 000112          JMP     @R2;            CALL SUBROUTINE
70 00202 000204'ADR$42:  .WORD   .+2;            RETURN ADDRESS
71                       ;       NOW POP RESULT TO FLAC
72 00204 012705          MOV     #FAC$42,R5;     ADDR OF FLAC
           000432'
73 00210 012625          MOV     (SP)+,(R5)+
74 00212 012625          MOV     (SP)+,(R5)+
75 00214 012625          MOV     (SP)+,(R5)+
76 00216 012625          MOV     (SP)+,(R5)+
77 00220 011700 RET$42:  MOV     @PC,R0;         MAKE R0 POSITIVE
78 00222 012705          MOV     #FAC$42,R5;     ADDR OF FLAC
           000432'
79 00226 005725          TST     (R5)+;          TEST THE FLAC
80 00230 002410          BLT     NEG$42;         BRANCH IF FLAC MINUS
81 00232 003013          BGT     PLS$42;         BRANCH IF PLUS
82 00234 005725          TST     (R5)+
83 00236 001011          BNE     PLS$42
84 00240 005725          TST     (R5)+
85 00242 001007          BNE     PLS$42
86 00244 005725          TST     (R5)+
87 00246 001005          BNE     PLS$42
88 00250 005000          CLR     R0;             FLAG FLAC AS ZERO
89 00252 005400 NEG$42:  NEG     R0;             FLAC IS NEG
90 00254 053766 CMF$42:  BIS     @#177776,20(SP) ;SET USER'S CONDS
           177776
           000020
91 00262 005700 PLS$42:  TST     R0;             SET COND CODES
92 00264 012600 CM1$42:  MOV     (SP)+,R0;       RESTORE USER'S REGS
93 00266 012601          MOV     (SP)+,R1
94 00270 012602          MOV     (SP)+,R2
95 00272 012603          MOV     (SP)+,R3
96 00274 012604          MOV     (SP)+,R4
```

```
97  00276 012605          MOV     (SP)+,R5
98  00300 005726          TST     (SP)+;          TEST IF STACK MODE
99  00302 001413          BEQ     RTI$42;         NO, SO RETURN
100 0304  100006          BPL     RT2$42;         BRANCH IF SINGLE PREC
101 0306  012666          MOV     (SP)+,6(SP);    POP USER' ARG,
          000006
102 0312  012666          MOV     (SP)+,6(SP)
          000006
103 0316  022626          CMP     (SP)+,(SP)+
104 0320  000002          RTI                     ;RETURN TO USER
105 0322  012666 RT2$42:  MOV     (SP)+,2(SP);    POP TWO WORD ARG,
          000002
106 0326  012666          MOV     (SP)+,2(SP)
          000002
107 0332  000002 RTI$42:  RTI
108
109               ;       ROUTINE TO MAKE J5RR CALLS
110 0334  004512 NAD$42:  JSR     R5,@R2;         CALL SUBROUTINE
111 0336  012705          MOV     (PC)+,R5;       PICK UP ADDR OF FLAC
112 0340  000432'         .WORD   FAC$42;         ARG ADDRESS (FLAC)
113 0342  010025          MOV     R0,(R5)+;       STORE RESULT INTO FLAC
114 0344  010125          MOV     R1,(R5)+
115 0346  010225          MOV     R2,(R5)+
116 0350  010325          MOV     R3,(R5)+
117 0352  000722          BR      RET$42;         GO DO STANDARD RETURN
118
119               ;       MORE MODE CHECKING
120 0354  103010 PLM$42:  BCC     STM$42;         BRANCH IF STACK MODE
121               ;       IMMEDIATE MODE
122 0356  010500          MOV     R5,R0;          ADDR IS USER'S PC
123 0360  005704          TST     R4;             SINGLE OR DOUBLE
124 0362  002003          BGE     PL1$42;         BRANCH IF SINGLE
125 0364  062705          ADD     #8.,R5;         UPDATE USER'S PC
          000010
126 0370  000652          BR      UPC$42
127 0372  022525 PL1$42:  CMP     (R5)+,(R5)+;    UPDATE PC
128 0374  000650          BR      UPC$42
129               ;       STACK MODE
130 0376  010600 STM$42:  MOV     SP,R0
131 0400  062700          ADD     #22,R0;         CALC ADDR OF ARG ON STACK
          000022
132 0404  005266          INC     14(SP);         FLAG STACK MODE
          000014
133 0410  005704          TST     R4;             SINGLE OR DOUBLE?
134 0412  002243          BGE     STK$42;         BRANCH IF SINGLE
135 0414  005466          NEG     14(SP);         FLAG DOUBLE
          000014
136 0420  000640          BR      STK$42
137
138               ;       ERROR: ROUTINE NOT AVAILABLE IN PACKAGE
139 0422  005000 ERR$42:  CLR     R0;             SIGNAL TRAPH ERROR
140 0424  004567          JSR     R5,$ERRA;       R1 POINTS TO BAD TRAP INSTR
          021366
141 0430  000774          BR      ERR$42;         HARD STOP
142
143               ;       FLOATING ACCUMULATOR
144 0432  000000 FAC$42:  .WORD   0,0,0,0
```

8

```
     0434 000000
     0436 000000
     0440 000000
145                          .IFDF   CND$6
146                          ;COMPARISON FUDGE
147  0442 012704 CMR$42: MOV      #CAR$42,R4;     ADDR OF RETURN ADDR
          000452;
148  0446 000167          JMP      $CMR
          002666
149  0452 000454;CAR$42: .WORD    .+2
150  0454 053766          BIS      @#177776,24(SP) ;SET USER COND
          177776
          000024
151  0462 022626          CMP      (SP)+,(SP)+;    POP STACK
152  0464 000677          BR       CM1$42
153                          .ENDC
154
155                          .IFDF   CND$5
156  0466 012704 CMD$42: MOV      #CAD$42,R4
          000476;
157  0472 000167          JMP      $CMD
          002544
158  0476 000254;CAD$42: .WORD    CMF$42
159                          .ENDC
160       040000             PMODE=40000
161       100000             DMODE=100000
162  0500 000000 TBL$42: .WORD    0,0,0,0,0,0,0,0 ;0-7
     0502 000000
     0504 000000
     0506 000000
     0510 000000
     0512 000000
     0514 000000
     0516 000000
163  0520 000000          .WORD    0,0             ;10-11
     0522 000000
164                          .IFDF   CND$2
165  0524 041712          .WORD    $ADR-PT$42+PMODE        ;12
166  0526 041706          .WORD    $SBR-PT$42+PMODE        ;13
167                          .ENDC
168                          .IFNDF  CND$2
169                          .WORD    0,0                    ;12-13
170                          .ENDC
171                          .IFDF   CND$1
172  0530 140606          .WORD    $ADD-PT$42+PMODE+DMODE  ;14
173  0532 140602          .WORD    $SBD-PT$42+PMODE+DMODE  ;15
174                          .ENDC
175                          .IFNDF  CND$1
176                          .WORD    0,0                    ;14-15
177                          .ENDC
178                          .IFDF   CND$5
179  0534 140370          .WORD    CMD$42-PT$42+PMODE+DMODE        ;16
180                          .ENDC
181                          .IFNDF  CND$5
182                          .WORD    0                      ;16
183                          .ENDC
184                          .IFDF   CND$6
```

```
185 0536 040344          .WORD   CMR$42-PT$42+PMODE          ;17
186                      .ENDC
187                      .IFNDF  CND$6
188                      .WORD   0                           ;17
189                      .ENDC
190 0540 000000          .WORD   0                           ;20
191                      .IFDF   CND$30
192 0542 057064          .WORD   $MLR-PT$42+PMODE            ;21
193                      .ENDC
194                      .IFNDF  CND$30
195                      .WORD   0                           ;21
196                      .ENDC
197                      .IFDF   CND$28
198 0544 156050          .WORD   $MLD-PT$42+PMODE+DMODE      ;22
199                      .ENDC
200                      .IFNDF  CND$28
201                      .WORD   0                           ;22
202                      .ENDC
203                      .IFDF   CND$16
204 0546 152112          .WORD   $DVD-PT$42+PMODE+DMODE      ;23
205                      .ENDC
206                      .IFNDF  CND$16
207                      .WORD   0                           ;23
208                      .ENDC
209 0550 000000          .WORD   0                           ;24
210                      .IFDF   CND$18
211 0552 053160          .WORD   $DVR-PT$42+PMODE            ;25
212                      .ENDC
213                      .IFNDF  CND$18
214                      .WORD   0                           ;25
215                      .ENDC
216                      .IFDF   CND$4
217 0554 003026          .WORD   AINT-PT$42                  ;26
218                      .ENDC
219                      .IFNDF  CND$4
220                      .WORD   0                           ;26
221                      .ENDC
222 0556 000000          .WORD   0,0,0,0,0,0,0               ;27-35
    0560 000000
    0562 000000
    0564 000000
    0566 000000
    0570 000000
    0572 000000
223                      .IFDF   CND$37
224 0574 017766          .WORD   SIN-PT$42,COS-PT$42         ;36-37
    0576 017732
225                      .ENDC
226                      .IFNDF  CND$37
227                      .WORD   0,0                         ;36-37
228                      .ENDC
229                      .IFDF   CND$13
230 0600 107654          .WORD   DSIN-PT$42+DMODE            ;40
231 0602 107576          .WORD   DCOS-PT$42+DMODE            ;41
232                      .ENDC
233                      .IFNDF  CND$13
234                      .WORD   0,0                         ;40-41
```

```
                        .ENDC
                        .IFDF   CND$39
0604 021062             .WORD   ATAN-PT$42              ;42
                        .ENDC
                        .IFNDF  CND$39
                        .WORD   0
                        .ENDC
0606 000000             .WORD   0                       ;43
                        .IFDF   CND$15
0610 111040             .WORD   DATAN-PT$42+DMODE       ;44
                        .ENDC
                        .IFNDF  CND$15
                        .WORD   0                       ;44
                        .ENDC
0612 000000             .WORD   0                       ;45
                        .IFDF   CND$41
0614 021552             .WORD   SQRT-PT$42              ;46
                        .ENDC
                        .IFNDF  CND$41
                        .WORD   0                       ;46
                        .ENDC
                        .IFDF   CND$14
0616 110356             .WORD   DSQRT-PT$42+DMODE       ;47
                        .ENDC
                        .IFNDF  CND$14
                        .WORD   0                       ;47
                        .ENDC
                        .IFDF   CND$38
0620 020306             .WORD   TANH-PT$42              ;50
                        .ENDC
                        .IFNDF  CND$38
                        .WORD   0                       ;50
                        .ENDC
                        .IFDF   CND$20
0622 014456             .WORD   EXP-PT$42               ;51
                        .ENDC
                        .IFNDF  CND$20
                        .WORD   0                       ;51
                        .ENDC
                        .IFDF   CND$19
0624 113612             .WORD   DEXP-PT$42+DMODE        ;52
                        .ENDC
                        .IFNDF  CND$19
                        .WORD   0                       ;52
                        .ENDC
                        .IFDF   CND$3
0626 002452             .WORD   ALOG-PT$42              ;53
0630 002446             .WORD   ALOG10-PT$42            ;54
                        .ENDC
                        .IFNDF  CND$3
                        .WORD   0,0                     ;53-54
                        .ENDC
                        .IFDF   CND$10
0632 106556             .WORD   DLOG-PT$42+DMODE        ;55
0634 106552             .WORD   DLOG10-PT$42+DMODE      ;56
                        .ENDC
                        .IFNDF  CND$10
```

```
292                             .WORD   0,0                             ;55-56
293                             .ENDC
294 0636 000000                 .WORD   0,0,0,0,0,0,0,0,0,0             ;57-70
    0640 000000
    0642 000000
    0644 000000
    0646 000000
    0650 000000
    0652 000000
    0654 000000
    0656 000000
    0660 000000
295                             .IFDF   CND$44
296 0662 061746                 .WORD   $LDR-PT$42+PMODE                ;71
297                             .ENDC
298                             .IFNDF  CND$44
299                             .WORD   0                               ;71
300                             .ENDC
301                             .IFDF   CND$45
302 0664 161760                 .WORD   $LDD-PT$42+PMODE+DMODE          ;72
303                             .ENDC
304                             .IFNDF  CND$45
305                             .WORD   0                               ;72
306                             .ENDC
307                             .IFDF   CND$46
308 0666 062000                 .WORD   $STR-PT$42+PMODE                ;73
309                             .ENDC
310                             .IFNDF  CND$46
311                             .WORD   0                               ;73
312                             .ENDC
313                             .IFDF   CND$47
314 0670 162054                 .WORD   $STD-PT$42+PMODE+DMODE          ;74
315                             .ENDC
316                             .IFNDF  CND$47
317                             .WORD   0                               ;74
318                             .ENDC
319 0672 000000                 .WORD   0,0,0                           ;75-77
    0674 000000
    0676 000000
320                             .ENDC
321                             .TITLE  $ADD05
322                             .IFDF   CND$1
323                             .GLOBL  $ADD,$SBD,$ERR
324                     ;       $ADD  --- THE DOUBLE PRECISION ADD ROUTINE
325                     ;       $ADD    V005A
326                     ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
327                     ;       ADD THE TOP STACK ITEM TO THE SECOND ITEM
328                     ;       AND LEAVE THE SUM IN THEIR PLACE.
329                     ;       $SBD  --- THE DOUBLE PRECISION SUBTRACT ROUTINE
330                     ;       SUBTRACT THE TOP STACK ITEM FROM THE SECOND ITEM
331                     ;       AND LEAVE THE DIFFERENCE IN PLACE OF THEM
332       000000                R0=%0
333       000001                R1=%1
334       000002                R2=%2
335       000003                R3=%3
336       000004                R4=%4
337       000005                R5=%5
```

```
338        000006          SP=%6
339        000007          PC=%7
340        000006          A1=6
341        000010          B1=8.
342        000012          C1=10.
343        000014          D1=12.
344        000016          A2=14.
345        000020          B2=16.
346        000022          C2=18.
347        000024          D2=20.
348        000000          SIGNS=0.
349        177304          MQ=177304
350        177312          NOR=177312
351        177314          LSH=177314
352        177316          ASH=177316
353        000000          F0=%0
354 0700 062716 $SBD:      ADD     #100000,@SP     ;NEGATE TOP STACK ITEM
         100000
355                        .IFDF   FPU
356              $ADD:     .WORD   170011  ;;SETD
357                        .WORD   172426  ;;LDD   (SP)+,F0        ;GET OPERAND
358                        .WORD   172026  ;;ADDD  (SP)+,F0        ;ADD
359                        .WORD   174046  ;;STD   F0,-(SP)        ;SUM TO STACK
360                        JMP     @(R4)+
361                        .ENDC
362                        .IFNDF  FPU
363 0704 010446 $ADD:      MOV     R4,-(SP)
364 0706 010546           MOV     R5,-(SP)
365 0710 005046           CLR     -(SP)   ;CLEAR SIGNS
366 0712 005004           CLR     R4      ;CLEAR EXPONENTS
367 0714 005005           CLR     R5
368 0716 006366           ASL     D1(SP)  ;SHIFT OUT SIGN OF TOP ITEM
         000014
369 0722 006166           ROL     C1(SP)
         000012
370 0726 006166           ROL     B1(SP)
         000010
371 0732 006166           ROL     A1(SP)  ;SHIFT A1
         000006
372 0736 156604           BISB    A1+1(SP),R4     ;GET E1
         000007
373 0742 001441           BEQ     A1Z$1   ;JUMP IF ZERO
374 0744 106116           ROLB    @SP     ;GET S1
375 0746 006366           ASL     D2(SP)  ;SHIFT OUT SIGN OF SECOND ITEM
         000024
376 0752 006166           ROL     C2(SP)
         000022
377 0756 006166           ROL     B2(SP)
         000020
378 0762 006166           ROL     A2(SP)  ;SHIFT A2
         000016
379 0766 156605           BISB    A2+1(SP),R5     ;GET E2
         000017
380 0772 001030           BNE     A2N$1   ;JUMP IF NOT 0
381 0774 106016           RORB    @SP     ;RECONSTRUCT A1
382 0776 006066           ROR     A1(SP)
         000006
```

```
383 1002 006066          ROR     B1(SP)
         000010
384 1006 006066          ROR     C1(SP)
         000012
385 1012 006066          ROR     D1(SP)
         000014
386 1016 016666          MOV     A1(SP),A2(SP)   ;FIRST ARG TO TOP OF STACK
         000006
         000016
387 1024 016666          MOV     B1(SP),B2(SP)
         000010
         000020
388 1032 016666          MOV     C1(SP),C2(SP)
         000012
         000022
389 1040 016666          MOV     D1(SP),D2(SP)
         000014
         000024
390 1046 005726 A1Z$1:   TST     (SP)+   ;FLUSH SIGNS
391 1050 000167          JMP     OUT$1   ;DONE
         000476
392 1054 106166 A2N$1:   ROLB    SIGNS+1(SP)     ;GET S2
         000001
393 1060 112766          MOVB    #1,A2+1(SP)     ;INSERT NORMAL BIT
         000001
         000017
394 1066 112766          MOVB    #1,A1+1(SP)     ;INSERT NORMAL BIT
         000001
         000007
395 1074 160405          SUB     R4,R5   ;R5=E2-E1, R4=E1
396 1076 003011          BGT     EXA$1   ;JUMP IF E2>E1
397 1100 016600          MOV     A2(SP),R0       ;R0=A2
         000016
398 1104 016601          MOV     B2(SP),R1       ;R1=B2
         000020
399 1110 016602          MOV     C2(SP),R2
         000022
400 1114 016603          MOV     D2(SP),R3
         000024
401 1120 000427          BR      SCK$1   ;GO CHECK SIGNS
402 1122 060504 EXA$1:   ADD     R5,R4   ;R5=E2-E1,R4=E2,E2>E1
403 1124 016600          MOV     A1(SP),R0       ;R0=A1
         000006
404 1130 016601          MOV     B1(SP),R1       ;R1=B1
         000010
405 1134 016602          MOV     C1(SP),R2
         000012
406 1140 016603          MOV     D1(SP),R3
         000014
407 1144 016666          MOV     A2(SP),A1(SP)
         000016
         000006
408 1152 016666          MOV     B2(SP),B1(SP)
         000020
         000010
409 1160 016666          MOV     C2(SP),C1(SP)
         000022
```

```
                000012
410 1166 016666         MOV     D2(SP),D1(SP)
                000024
                000014
411 1174 000316         SWAB    @SP         ;EXCHANGE SIGNS
412 1176 005405         NEG     R5          ;E1-E2
413 1200 126616 SCK$1:  CMPB    SIGNS+1(SP),@SP ;COMPARE SIGNS
                000001
414 1204 001412         BEQ     ECK$1       ;THEY'RE THE SAME, CHECK EXPONENT
415 1206 005403         NEG     R3          ;NEGATE OPERAND
416 1210 005502         ADC     R2
417 1212 005501         ADC     R1
418 1214 005500         ADC     R0
419 1216 005402         NEG     R2
420 1220 005501         ADC     R1
421 1222 005500         ADC     R0
422 1224 005401         NEG     R1
423 1226 005500         ADC     R0
424 1230 005400         NEG     R0
425 1232 005705 ECK$1:  TST     R5          ;CHECK EXPONENTS
426 1234 001467         BEQ     SFD$1       ;JUMP IF E1=E2
427 1236 022705 SFT$1:  CMP     #-57.,R5            ;IS THERE ANY POINT IN SHIFTING?
                177707
428 1242 003411         BLE     SFR$1       ;YES
429 1244 016600         MOV     A1(SP),R0           ;NO, ANSWER IS OPERAND
                000006
430 1250 016601         MOV     B1(SP),R1           ;WITH THE LARGER EXPONENT
                000010
431 1254 016602         MOV     C1(SP),R2
                000012
432 1260 016603         MOV     D1(SP),R3
                000014
433 1264 000504         BR      NOD$1
434 1266 022705 SFR$1:  CMP     #-8.,R5 ;CHECK # OF BITS TO SHIFT
                177770
435 1272 003442         BLE     SR8$1       ;JUMP IF NOT MORE THAN 1/2 WORD
436                     .IFNDF  MULDIV
437 1274 005046         CLR     -(SP)       ;SET UP EXTENSION BITS
438 1276 005700         TST     R0          ;ACCORDING TO HIGH ORDER FRACTION
439 1300 100001         BPL     SF1$1       ;JUMP IF +
440 1302 005116         COM     @SP
441                     .ENDC
442                     .IFDF   MULDIV
443                     TST     R0
444                     .WORD   006746  ;;SEX   -(SP)   ;EXTEND SIGN
445                     .ENDC
446 1304 022705 SF1$1:  CMP     #-16.,R5
                177760
447 1310 002411         BLT     S16$1       ;JUMP IF NOT MORE THAN A WORD TO SHIFT
448 1312 010203         MOV     R2,R3       ;SHIFT A WORD AT A TIME
449 1314 010102         MOV     R1,R2
450 1316 010001         MOV     R0,R1
451 1320 011600         MOV     @SP,R0      ;USE EXTENSION
452 1322 062705         ADD     #16.,R5 ;ADJUST EXPONENT
                000020
453 1326 001366         BNE     SF1$1       ;TRY AGAIN
454 1330 005726         TST     (SP)+       ;POP EXTENSION
```

```
455 1332 000430          BR      SFD$1   ;SHIFT IS ALL DONE
456                      .IFDF   EAE
457              S16$1:  CMP     #-3,R5
458                      BLE     S8A$1   ;JUMP IF NOT MORE THAN 3 TO SHIFT
459                      MOV     R4,@SP  ;SAVE EXP
460                      MOV     #MQ,R4  ;POINT TO MQ
461                      MOV     R3,@R4  ;LOW ORDER PARTS TO AC,MQ
462                      MOV     R2,-(R4)
463                      MOV     R5,@#LSH        ;SHIFT THEM
464                      MOV     (R4)+,R2        ;SAVE PARTIAL R2
465                      MOV     @R4,R3  ;LOWEST ORDER IS DONE
466                      CLR     @R4
467                      MOV     R1,-(R4)        ;SET UP NEXT HIGHER WORD
468                      MOV     R5,@#LSH        ;AND SHIFT IT
469                      TST     (R4)+   ;POINT TO MQ
470                      BIS     @R4,R2  ;FINISH R2
471                      MOV     R1,@R4
472                      MOV     R0,-(R4)        ;DO HIGH ORDER NOW
473                      MOV     R5,@#ASH
474                      MOV     (R4)+,R0        ;HIGH ORDER DONE
475                      MOV     @R4,R1
476                      MOV     (SP)+,R4
477                      BR      SFD$1
478                      .ENDC
479                      .IFNDF  EAE&MULDIV
480 1334 022705 S16$1:  CMP     #-8.,R5
         177770
481 1340 003416          BLE     S8A$1   ;JUMP IF NOT MORE THAN 1/2 WORD TO GO
482 1342 062705          ADD     #16.,R5 ;SHIFT LEFT 16-X
         000020
483 1346 006303 SL8$1:   ASL     R3      ;SHIFT LEFT
484 1350 006102          ROL     R2
485 1352 006101          ROL     R1
486 1354 006100          ROL     R0
487 1356 006116          ROL     @SP
488 1360 005305          DEC     R5      ;COUNT LOOP
489 1362 003371          BGT     SL8$1
490 1364 010203          MOV     R2,R3
491 1366 010102          MOV     R1,R2
492 1370 010001          MOV     R0,R1
493 1372 012600          MOV     (SP)+,R0
494 1374 000407          BR      SFD$1   ;SHIFT DONE
495                      .ENDC
496                      .IFDF   MULDIV
497              S16$1:  CMP     #-3,R5  ;JUMP IF NOT MORE THAN 3 TO SHIFT
498                      BLE     S8A$1
499                      MOV     R4,@SP  ;SAVE EXP AND SHIFT COUNT
500                      MOV     R5,-(SP)
501                      MOV     R1,R4   ;SAVE R1
502                      .WORD   073005  ;;      ASHC    R5,R0   ;SHIFT HIGH ORDE
503                      MOV     R2,R5   ;SAVE R2
504                      .WORD   073416  ;;      ASHC    @SP,R4  ;SHIFT IT
505                      MOV     R2,R4
506                      MOV     R5,R2   ;R2 DONE
507                      MOV     R3,R5   ;SET UP LOW ORDER
508                      .WORD   073426  ;;      ASHC    (SP)+,R4        ;DO LOW
509                      MOV     R5,R3
```

```
510                             MOV     (SP)+,R4        ;RESTORE EXPONENT TO R4
511                             BR      SFD$1
512                             .ENDC
513 1376 005726         S8A$1: TST     (SP)+   ;POP EXTENSION
514 1400 006200         SR8$1: ASR     R0      ;SHIFT RIGHT
515 1402 006001                ROR     R1
516 1404 006002                ROR     R2
517 1406 006003                ROR     R3
518 1410 005205                INC     R5      ;COUNT LOOP
519 1412 002772                BLT     SR8$1
520 1414 066603 SFD$1: ADD     D1(SP),R3       ;FORM THE SUM
         000014
521 1420 005502                ADC     R2
522 1422 005501                ADC     R1
523 1424 005500                ADC     R0
524 1426 066602                ADD     C1(SP),R2
         000012
525 1432 005501                ADC     R1
526 1434 005500                ADC     R0
527 1436 066601                ADD     B1(SP),R1
         000010
528 1442 005500                ADC     R0
529 1444 066600                ADD     A1(SP),R0
         000006
530 1450 126616                CMPB    SIGNS+1(SP),@SP ;CHECK FOR UNEQUAL SIGNS
         000001
531 1454 001065                BNE     SUB$1   ;GO CLEAN UP SUBTRACT
532 1456 030027                BIT     R0,#1000
         001000
533 1462 001405                BEQ     NOD$1   ;JUMP IF NO NORMAL BIT OVERFLOW
534 1464 006200                ASR     R0
535 1466 006001                ROR     R1
536 1470 006002                ROR     R2
537 1472 006003                ROR     R3
538 1474 005204                INC     R4      ;INCREASE EXPONENT
539 1476 000304 NOD$1: SWAB    R4      ;MOVE EXPONENT LEFT
540 1500 001031                BNE     OVF$1   ;JUMP IF OVERFLOW
541 1502 150004 NFL$1: BISB    R0,R4   ;INSERT HIGH ORDER FRACTION
542 1504 006026                ROR     (SP)+   ;INSERT SIGN
543 1506 006004                ROR     R4
544 1510 006001                ROR     R1
545 1512 006002                ROR     R2
546 1514 006003                ROR     R3
547 1516 005503                ADC     R3
548 1520 005502                ADC     R2
549 1522 005501                ADC     R1
550 1524 005504                ADC     R4
551 1526 102417                BVS     OVR$1   ;JUMP IF OVERFLOW ON ROUND
552 1530 103416                BCS     OVR$1
553 1532 010466                MOV     R4,A2+0-2(SP)   ;STORE EXPONENT AND SIGN
         000014
554 1536 010166                MOV     R1,B2+0-2(SP)   ;INSERT LOW ORDER FRACTION
         000016
555 1542 010266                MOV     R2,C2+0-2(SP)
         000020
556 1546 010366                MOV     R3,D2+0-2(SP)
         000022
```

```
557 1552 012605 OUT$1:  MOV     (SP)+,R5
558 1554 012604         MOV     (SP)+,R4
559 1556 062706         ADD     #8.,SP  ;POP SECOND ARGUMENT
         000010
560 1562 000134         JMP     @(R4)+  ;DONE, RETURN
561                 ;
562 1564 005726 OVF$1:  TST     (SP)+   ;POP SIGN
563 1566 004567 OVR$1:  JSR     R5,$ERR ;ERROR 3,1
         020214
564 1572 000767         BR      OUT$1
565 1574     003        .BYTE   3
566 1575     001        .BYTE   1
567 1576 005704 UTS$1:  TST     R4      ;CHECK FOR UNDERFLOW
568 1600 003336         BGT     NOD$1
569 1602 004567 UNF$1:  JSR     R5,$ERR ;ERROR 5,1
         020200
570 1606 000401         BR      UND$1
571 1610     005        .BYTE   5
572 1611     001        .BYTE   1
573 1612 005000 UND$1:  CLR     R0
574 1614 005001         CLR     R1      ;UNDERFLOW, TREAT AS 0
575 1616 005002         CLR     R2
576 1620 005003         CLR     R3
577 1622 005016 ZER$1:  CLR     @SP     ;SET SIGN PLUS
578 1624 005004         CLR     R4
579 1626 000725         BR      NFL$1   ;FINISH OUT NORMALLY
580                 ;
581 1630 005700 SUB$1:  TST     R0      ;CHECK HIGH ORDER RESULT FRACTION
582 1632 003015         BGT     BT9$1   ;IF POSITIVE SIGN IS OK
583 1634 001425         BEQ     ZTS$1   ;CHECK FOR ZERO RESULT
584 1636 005403         NEG     R3      ;GET ABSOLUTE VALUE
585 1640 005502         ADC     R2
586 1642 005501         ADC     R1
587 1644 005500         ADC     R0
588 1646 005402         NEG     R2
589 1650 005501         ADC     R1
590 1652 005500         ADC     R0
591 1654 005401         NEG     R1
592 1656 005500         ADC     R0
593 1660 000316         SWAB    @SP     ;EXCHANGE SIGNS
594 1662 005400         NEG     R0
595 1664 001411         BEQ     ZTS$1;  CHECK FOR ZERO RESULT
596 1666            BT9$1:
597                     .IFDF   EAE
598                     BIT     R0,#740
599                     BNE     B9A$1   ;JUMP IF NOT MORE THAN 4 TO SHIFT
600                     MOV     R4,-(SP)        ;SAVE EXP
601                     MOV     #MQ,R4  ;POINT TO MQ
602                     MOV     R1,@R4          ;LOW  ORDER FRACTION TO MQ
603                     MOV     R0,-2(R4)       ;HIGH ORDER FRACTION TO AC
604                     CLR     @#NOR   ;NORMALIZE
605                     MOV     @#NOR,-(SP)     ;SAVE SCALE
606                     SUB     #6,@SP  ;COMPENSATE FOR NORMAL BIT POSITION
607                     MOV     R1,@R4  ;GET 2 HIGH ORDER PARTS
608                     MOV     R0,-(R4)
609                     MOV     @SP,@#LSH       ;SHIFT THEM
610                     MOV     (R4)+,R0        ;R0 DONE
```

```
                        MOV     @R4,R1  ;SAVE PARTIAL R1
                        MOV     R2,@R4  ;GET NEXT
                        CLR     -(R4)
                        MOV     @SP,@#LSH       ;SHIFT IT
                        BIS     (R4)+,R1        ;FINISH R1
                        MOV     R3,@R4  ;GET NEXT
                        MOV     R2,-(R4)
                        MOV     @SP,@#LSH       ;SHIFT IT
                        MOV     (R4)+,R2        ;FINISH R2
                        MOV     @R4,R3  ;R3 DONE
                        SUB     (SP)+,@SP       ;COMPENSATE EXPONENT
                        MOV     (SP)+,R4        ;RESTORE IT TO R4
                        BGT     NOD$1   ;JUMP IF NO UNDERFLOW
                        BR      UNF$1
                        .ENDC
1666 030027 B9A$1:      BIT     R0,#400 ;CHECK NORMAL BIT
     000400
1672 001341             BNE     UTS$1   ;JUMP IF FOUND
1674 005304             DEC     R4      ;DECREASE EXPONENT
1676 006303             ASL     R3      ;DOUBLE FRACTION
1700 006102             ROL     R2
1702 006101             ROL     R1
1704 006100             ROL     R0
1706 000767             BR      B9A$1   ;TRY AGAIN
1710 162704 ZTS$1:      SUB     #8.,R4  ;REDUCE EXPONENT
     000010
1714 005701             TST     R1
1716 001020             BNE     ZT1$1   ;JUMP IF ONLY R0=0
1720 162704             SUB     #16.,R4
     000020
1724 010201             MOV     R2,R1
1726 001012             BNE     ZT2$1   ;JUMP IF R2 NOT 0
1730 162704             SUB     #16.,R4
     000020
1734 005703             TST     R3
1736 001731             BEQ     ZER$1   ;ANSWER IS 0
1740 150301             BISB    R3,R1   ;MOVE BYTES TO R0,R1
1742 000301             SWAB    R1
1744 000303             SWAB    R3
1746 150300             BISB    R3,R0
1750 005003             CLR     R3      ;MAKE ALL OTHERS 0
1752 000745             BR      BT9$1   ;GO NORMALIZE
1754 010302 ZT2$1:      MOV     R3,R2
1756 005003             CLR     R3
1760 000301 ZT1$1:      SWAB    R1      ;MOVE ALL BYTES LEFT
1762 150100             BISB    R1,R0
1764 105001             CLRB    R1
1766 000302             SWAB    R2
1770 150201             BISB    R2,R1
1772 105002             CLRB    R2
1774 000303             SWAB    R3
1776 150302             BISB    R3,R2
2000 105003             CLRB    R3
2002 000731             BR      BT9$1   ;GO NORMALIZE WHAT'S LEFT
                        .ENDC
                        .ENDC
```

```
                        .TITLE  $ADR04
                        .IFDF   CND$2
                        .GLOBL  $ADR,$SBR,$ERR
                ;       $ADR  ---- THE REAL ADD ROUTINE
                ;       $ADR    V004A
                ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                ;       REPLACE THE TWO ITEMS ON TOP OF THE STACK
                ;       WITH THEIR SUM.
                ;       $SBR  ---- THE REAL SUBTRACT ROUTINE
                ;       SUBTRACT THE TOP STACK ITEM FROM THE SECOND ITEM
                ;       REPLACE THEM BOTH WITH THE DIFFERENCE.
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000007          PC=%7
        000000          SIGNS=0
        000004          A1=4
        000006          B1=6
        000010          A2=8.
        000012          B2=10.
        177302          AC=177302
        177304          MQ=177304
        177312          NOR=177312
        177316          ASH=177316
        000000          F0=%0
02004   062716  $SBR:   ADD     #100000,@SP     ;CHANGE THE SIGN OF TOP ITEM
        100000
                        .IFDF   FPU
                $ADR:   .WORD   170001  ;;SETF
                        .WORD   172426  ;;LDF   (SP)+,F0        ;GET OPERAND
                        .WORD   172026  ;;ADDF  (SP)+,F0        ;ADD
                        .WORD   174046  ;;STF   F0,-(SP)        ;SUM TO STACK
                        JMP     @(R4)+
                        .ENDC
                        .IFNDF  FPU
02010   010446  $ADR:   MOV     R4,-(SP)
02012   005046          CLR     -(SP)   ;CLEAR SIGNS
02014   005002          CLR     R2      ;CLEAR EXPONENTS
02016   005003          CLR     R3
02020   006366          ASL     B1(SP)  ;SHIFT B1
        000006
02024   006166          ROL     A1(SP)  ;SHIFT A1
        000004
02030   156603          BISB    A1+1(SP),R3     ;GET E1
        000005
02034   001574          BEQ     OUT$2   ;JUMP IF ZERO
02036   106116          ROLB    @SP     ;GET S1
02040   006366          ASL     B2(SP)  ;SHIFT B2
        000012
02044   006166          ROL     A2(SP)  ;SHIFT A2
        000010
02050   156602          BISB    A2+1(SP),R2     ;GET E2
        000011
```

```
51 02054 001014          BNE     A2N$2   ;JUMP IF NOT 0
52 02056 106016          RORB    @SP     ;RECONSTRUCT A1,B1
53 02060 006066          ROR     A1(SP)
         000004
54 02064 006066          ROR     B1(SP)
         000006
55 02070 016666          MOV     A1(SP),A2(SP)   ;FIRST ARG TO TOP OF STACK
         000004
         000010
56 02076 016666          MOV     B1(SP),B2(SP)
         000006
         000012
57 02104 000550          BR      OUT$2   ;DONE
58 02106 106166 A2N$2:   ROLB    SIGNS+1(SP)     ;GET S2
         000001
59 02112 112766          MOVB    #1,A2+1(SP)     ;INSERT NORMAL BIT
         000001
         000011
60 02120 112766          MOVB    #1,A1+1(SP)     ;INSERT NORMAL BIT
         000001
         000005
61 02126 160302          SUB     R3,R2   ;R2=E2-E1, R3=E1
62 02130 003005          BGT     EXA$2   ;JUMP IF E2>=E1
63 02132 016600          MOV     A2(SP),R0       ;R0=A2
         000010
64 02136 016601          MOV     B2(SP),R1       ;R1=B2
         000012
65 02142 000415          BR      SCK$2   ;CHECK SIGNS
66 02144 060203 EXA$2:   ADD     R2,R3   ;R2=E2-E1,R3=E2,E2>E1
67 02146 016600          MOV     A1(SP),R0       ;R0=A1
         000004
68 02152 016601          MOV     B1(SP),R1       ;R1=B1
         000006
69 02156 016666          MOV     A2(SP),A1(SP)
         000010
         000004
70 02164 016666          MOV     B2(SP),B1(SP)
         000012
         000006
71 02172 000316          SWAB    @SP     ;EXCHANGE SIGNS
72 02174 005402          NEG     R2      ;E1=E2
73 02176 126616 SCK$2:   CMPB    SIGNS+1(SP),@SP ;SEE IF SIGNS ARE THE SAME
         000001
74 02202 001403          BEQ     ECK$2   ;YES, CHECK EXPONENTS
75 02204 005401          NEG     R1      ;NEGATE FRACTION
76 02206 005500          ADC     R0
77 02210 005400          NEG     R0
78 02212 005702 ECK$2:   TST     R2
79 02214 001450          BEQ     SFD$2   ;JUMP IF E1=E2
80 02216 022702 SFT$2:   CMP     #-25.,R2        ;IS THERE ANY POINT IN SHIFTING?
         177747
81 02222 003405          BLE     SFR$2   ;YES
82 02224 016600          MOV     A1(SP),R0       ;NO, ANSWER IS OPERAND
         000004
83 02230 016601          MOV     B1(SP),R1       ;WITH THE LARGER EXPONENT
         000006
84 02234 000456          BR      NOD$2
```

```
                        .IFDF   EAE
                SFR$2:  MOV     R1,@#MQ ;MOVE FRACTION TO AC,MQ
                        MOV     R0,@#AC
                        MOV     R2,@#ASH        ;SHIFT RIGHT TO EQUALIZE EXPONEN
                        MOV     @#MQ,R1 ;RECOVER SHIFTED FRACTION
                        MOV     @#AC,R0
                        .ENDC
                        .IFDF   MULDIV
                SFR$2:  .WORD   073002  ;;ASHC  R2,R0
                        .ENDC
                        .IFNDF  EAE&MULDIV
02236 022702    SFR$2:  CMP     #-8.,R2 ;CHECK # OF BITS TO SHIFT
      177770
02242 003431            BLE     SF0$2   ;JUMP IF NOT MORE THAN 1/2 WORD
02244 005004            CLR     R4      ;SET UP EXTENSION BITS
02246 005700            TST     R0      ;BASED ON HIGH ORDER FRACTION
2250  100001            BPL     NCP$2   ;JUMP IF +
2252  005104            COM     R4      ;- OTHERWISE
2254  022702    NCP$2:  CMP     #-16.,R2
      177760
2260  002405            BLT     SRL$2   ;JUMP IF LESS THAN ONE WORD TO SHIFT
2262  010001            MOV     R0,R1   ;SHIFT RIGHT A WHOLE WORD
2264  010400            MOV     R4,R0   ;USE EXTENSION BITS
2266  062702            ADD     #16.,R2 ;ACCOUNT FOR SHIFT
      000020
2272  001421            BEQ     SFD$2
2274  022702    SRL$2:  CMP     #-8.,R2
      177770
2300  003412            BLE     SF0$2   ;JUMP IF NOT MORE THAN 1/2 WORD
2302  062702            ADD     #16.,R2 ;SHIFT LEFT 16-X
      000020
2306  006301    SFL$2:  ASL     R1
2310  006100            ROL     R0
2312  006104            ROL     R4
2314  005302            DEC     R2      ;COUNT LOOP
2316  003373            BGT     SFL$2
2320  010001            MOV     R0,R1   ;PUT RESULT IN R0, R1
2322  010400            MOV     R4,R0
2324  000404            BR      SFD$2
2326  006200    SF0$2:  ASR     R0      ;SHIFT A MIN AND B MIN
2330  006001            ROR     R1
2332  005202            INC     R2      ;REDUCE EXPONENT DIFFERENCE
2334  002774            BLT     SF0$2
                        .ENDC
2336  066600    SFD$2:  ADD     A1(SP),R0       ;A1+A2
      000004
2342  066601            ADD     B1(SP),R1       ;B1+B2
      000006
2346  005500            ADC     R0
2350  126616            CMPB    SIGNS+1(SP),@SP
      000001
2354  001034            BNE     SUB$2   ;GO CLEAN UP SUBTRACT
2356  030027            BIT     R0,#1000
      001000
2362  001403            BEQ     NOD$2   ;JUMP IF NO NORMAL BIT OVERFLOW
2364  006200            ASR     R0
2366  006001            ROR     R1
```

```
133 2370 005203           INC     R3      ;INCREASE EXPONENT
134 2372 000303 NOD$2:    SWAB    R3      ;MOVE EXPONENT LEFT
135 2374 001020           BNE     OVR$2   ;JUMP IF OVERFLOW
136 2376 150003           BISB    R0,R3
137 2400 006016           ROR     @SP     ;INSERT SIGN
138 2402 006003           ROR     R3
139 2404 006001           ROR     R1
140 2406 005501           ADC     R1      ;ROUND SUM
141 2410 005503           ADC     R3
142 2412 102411           BVS     OVR$2   ;JUMP IF OVERFLOW ON ROUND
143 2414 103410           BCS     OVR$2
144 2416 010366 STR$2:    MOV     R3,A2(SP)       ;STORE EXPONENT AND SIGN
         000010
145 2422 010166           MOV     R1,B2(SP)       ;INSERT LOW ORDER FRACTION
         000012
146 2426 005726 OUT$2:    TST     (SP)+   ;POP SIGNS
147 2430 012604           MOV     (SP)+,R4
148 2432 022626           CMP     (SP)+,(SP)+     ;POP FIRST ARGUMENT
149 2434 000134           JMP     @(R4)+  ;DONE, RETURN
150                       ;
151 2436 004567 OVR$2:    JSR     R5,$ERR ;ERROR 3,2
         017344
152 2442 000771           BR      OUT$2
153 2444    003           .BYTE   3
154 2445    002           .BYTE   2
155                       ;
156 2446 005700 SUB$2:    TST     R0      ;CHECK HIGH ORDER RESULT FRACTION
157 2450 003005           BGT     BT9$2   ;IF POSITIVE SIGN IS OK
158 2452 001413           BEQ     ZTS$2   ;CHECK FOR ZERO RESULT
159 2454 005400           NEG     R0      ;GET ABSOLUTE VALUE
160 2456 005401           NEG     R1
161 2460 005600           SBC     R0
162 2462 000316           SWAB    @SP     ;EXCHANGE SIGNS
163 2464          BT9$2:
164                       .IFDF   EAE
165                       BIT     R0,#700
166                       BNE     B9A$2   ;JUMP IF NOT MORE THAN 2 TO SHIFT
167                       MOV     R1,@#MQ ;RESULT FRACTION TO AC,MQ
168                       MOV     R0,@#AC
169                       CLR     @#NOR   ;NORMALIZE
170                       SUB     @#NOR,R3        ;ADJUST EXPONENT
171                       MOV     #-6,@#ASH       ;SHIFT TO CORRECT POSITION
172                       ADD     #6,R3   ;COMPENSATE EXPONENT
173                       BLE     UNF$2   ;JUMP IF UNDERFLOW
174                       MOV     @#AC,R0
175                       MOV     @#MQ,R1 ;GET FRACTION BACK
176                       BR      NOD$2
177                       .ENDC
178 2464 030027 B9A$2:    BIT     R0,#400
         000400
179 2470 001014           BNE     UTS$2   ;JUMP IF NORMAL BIT FOUND
180 2472 005303           DEC     R3      ;DECREASE EXPONENT
181 2474 006301           ASL     R1      ;DOUBLE FRACTION
182 2476 006100           ROL     R0
183 2500 000771           BR      B9A$2   ;TRY AGAIN
184 2502 005701 ZTS$2:    TST     R1      ;CHECK LOW ORDER PART
185                       .IFDF   EAE
```

```
186                             BNE     BT9$2
187                             BR      ZER$2
188                             .ENDC
189                             .IFNDF  EAE
190 2504 001415                 BEQ     ZER$2
191 2506 000301                 SWAB    R1          ;SAVE NORMALIZE SOME TIME
192 2510 150100                 BISB    R1,R0       ;MOVE BITS LEFT
193 2512 105001                 CLRB    R1
194 2514 162703                 SUB     #8.,R3      ;TELL EXPONENT ABOUT IT
         000010
195 2520 000761                 BR      BT9$2
196                             .ENDC
197 2522 005703 UTS$2:          TST     R3          ;CHECK FOR UNDERFLOW
198 2524 003322                 BGT     NOD$2       ;JUMP IF NONE
199 2526 004567 UNF$2:          JSR     R5,$ERR     ;ERROR 5,2
         017254
200 2532 000401                 BR      UND$2
201 2534    005                 .BYTE   5
202 2535    002                 .BYTE   2
203 2536 005001 UND$2:          CLR     R1          ;UNDERFLOW, TREAT AS 0
204 2540 005003 ZER$2:          CLR     R3          ;CLEAR EXPONENT
205 2542 000725                 BR      STR$2
206                             .ENDC
207                             .ENDC
```

```
1                               .TITLE  $ALG03
2                               .IFDF   CND$3
3                       ;
4                       ;       ALOG    V03A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  ALOG,ALOG10,$ERR;
9                               .IFNDF  FPU
10                              .GLOBL  $POLSH,$ADR,$SBR,$MLR,$DVR,$IR;
11                              .ENDC
12                      ;       THE FORTRAN ALOG AND ALOG10 FUNCTIONS
13                      ;       CALLING SEQUENCE:
14                      ;       JSR     R5,ALOG (OR ALOG10)
15                      ;       BR      A
16                      ;       .WORD   ARGUMENT ADDRESS
17                      ;A:
18                      ;       RETURNS LN(ARG) (OR LOG10(ARG)) IN R0,R1.
19                      ;
20              000000          R0=%0
21              000001          R1=%1
22              000002          R2=%2
23              000003          R3=%3
24              000004          R4=%4
25              000005          R5=%5
26              000006          SP=%6
27              000007          PC=%7
28              000000          F0=%0
29              000001          F1=%1
30              000002          F2=%2
31              000003          F3=%3
32                              .IFNDF  FPU
33  02544 011746 ALOG10: MOV     @PC,-(SP)       ;GET 0004XX AS A FLAG
34  02546 000401         BR      LOG$3
35  02550 005046 ALOG:   CLR     -(SP)   ;FLAG ALOG
36  02552 016504 LOG$3:  MOV     2(R5),R4        ;GET ARG ADDRESS
            000002
37  02556 012746         MOV     #071030,-(SP)   ;PUSH -1/2*LN(2)
            071030
38  02562 012746         MOV     #137661,-(SP)
            137661
39  02566 024646         CMP     -(SP),-(SP)     ;GET WORK SPACE
40  02570 016446         MOV     2(R4),-(SP)     ;GET ARG
            000002
41  02574 011446         MOV     @R4,-(SP)
42  02576 003034         BLE     ERR$3   ;JUMP IF NOT POSITIVE
43  02600 006316         ASL     @SP
44  02602 116666         MOVB    1(SP),12.(SP)   ;GET EXPONENT
            000001
            000014
45  02610 112766         MOVB    #200,1(SP)      ;TRANSFORM ARG TO (1/2,1)
            000200
            000001
46  02616 006016         ROR     @SP
47  02620 012746         MOV     #002363,-(SP)   ;PUSH 1/2*ROOT2
            002363
48  02624 012746         MOV     #040065,-(SP)
```

```
                040065
49  02630 016646        MOV     6(SP),-(SP)     ;PUSH X
          000006
50  02634 016646        MOV     6(SP),-(SP)
          000006
51  02640 012746        MOV     #002363,-(SP)   ;PUSH 1/2+ROOT2
          002363
52  02644 012746        MOV     #040065,-(SP)
          040065
53  02650 004467        JSR     R4,$POLSH       ;ENTER POLISH MODE
          016770
54  02654 002004'       .WORD   $SBR,UP$3,$ADR,$DVR     ;GET (X-ROOT2)/
    02656 002766'
    02660 002010'
    02662 013256'
55                                                      ;(X+ROOT2)
56  02664 003014'       .WORD   DUP$3,DUP$3     ;GET THREE COPIES
    02666 003014'
57  02670 017162'       .WORD   $MLR,REG$3,STK$3,STK$3,STK$3    ;SET UP POLYNOMI
    02672 002742'
    02674 002754'
    02676 002754'
    02700 002754'
58  02702 017162'       .WORD   $MLR,$ADR,$MLR,$ADR,$MLR,$ADR,$MLR,$ADR
    02704 002010'
    02706 017162'
    02710 002010'
    02712 017162'
    02714 002010'
    02716 017162'
    02720 002010'
59                                      ;EXPAND POLYNOMIAL
60  02722 003000'       .WORD   SCL$3,$IR,PL2$3,$MLR    ;GET LN(EXP)
    02724 016062'
    02726 003026'
    02730 017162'
61  02732 002010'       .WORD   $ADR,EXI$3      ;COMBINE WITH FRACTION
    02734 003040'
62                                                      ;AND CHECK IF DONE
63  02736 017162'       .WORD   $MLR,EXI$3      ;MULTIPLY BY LOG10(E) AND RETURN
    02740 003040'
64                  ;
65  02742 012600 REG$3: MOV     (SP)+,R0        ;POP Y
66  02744 012601        MOV     (SP)+,R1
67  02746 012702        MOV     #CON$3+4,R2     ;POINT TO COEFFICIENTS
          003124'
68  02752 000402        BR      STC$3
69  02754 010146 STK$3: MOV     R1,-(SP)        ;PUSH Y
70  02756 010046        MOV     R0,-(SP)
71  02760 014246 STC$3: MOV     -(R2),-(SP)     ;PUSH COEFFICIENT
72  02762 014246        MOV     -(R2),-(SP)
73  02764 000134        JMP     @(R4)+
74                  ;
75  02766 012666 UP$3:  MOV     (SP)+,10.(SP)   ;MOVE ITEM TO WORK SPACE
          000012
76  02772 012666        MOV     (SP)+,10.(SP)
          000012
```

```
77 02776 000134                 JMP     @(R4)+
78                      ;
79 03000 005046 SCL$3:  CLR     -(SP)
80 03002 156616         BISB    6(SP),@SP       ;GET EXPONENT
         000006
81 03006 162716         SUB     #200,@SP        ;REMOVE EXCESS 128
         000200
82 03012 000134                 JMP     @(R4)+
83                      ;
84 03014 016646 DUP$3:  MOV     2(SP),-(SP)
         000002
85 03020 016646                 MOV     2(SP),-(SP)     ;DUPLICATE STACK ITEM
         000002
86 03024 000134                 JMP     @(R4)+
87                      ;
88 03026 012746 PL2$3:  MOV     #071030,-(SP)   ;PUSH LN(2)
         071030
89 03032 012746                 MOV     #040061,-(SP)
         040061
90 03036 000134                 JMP     @(R4)+
91                      ;
92 03040 105366 EXI$3:  DECB    5(SP)   ;CHECK FOR ALOG10
         000005
93 03044 002405                 BLT     LGT$3   ;NO, DONE
94 03046 012746                 MOV     #055731,-(SP)   ;PUSH LOG10(E)
         055731
95 03052 012746                 MOV     #037736,-(SP)
         037736
96 03056 000134                 JMP     @(R4)+
97 03060 012600 LGT$3:  MOV     (SP)+,R0        ;POP RESULT
98 03062 012601                 MOV     (SP)+,R1
99 03064 005726                 TST     (SP)+   ;FLUSH FLAG
100 3066 000205                 RTS     R5
101 3070 062706 ERR$3:  ADD     #14.,SP
         000016
102 3074 004567                 JSR     R5,$ERR ;ERROR 4,10
         016706
103 3100 000205                 RTS     R5
104 3102    004                 .BYTE   4
105 3103    012                 .BYTE   10.
106                             .ENDC
107                             .IFDF   FPU
108             ALOG10: MOV     @PC,R4;         GET 0004XX AS ALOG10 FLAG
109                     BR      LOG$3;
110             ALOG:   CLR     R4;             GET 0 AS ALOG FLAG
111             LOG$3:  SETF    ;               SINGLE PRECISION FP
112                     SETI    ;               SHORT INTEGERS
113                     MOV     #FCO$3,R0       ;POINTER TO CONSTANTS FOR ROUTIN
114                     LDF     @2(R5),F2;      GET ARGUMENT
115                     CFCC
116                     BLE     ERR$3;          JUMP IF NOT POSITIVE
117                     STEXP   F2,R1;          GET EXPONENT OF ARG
118                     LDCIF   R1,F3;          CONVERT T O FP FORM
119                     MULF    (R0)+,F3;       SCALE FACTOR=EXPONENT*LN(2)
120                     LDEXP   #0,F2;          TRANSFORM ARG TO (1/2,1)
121                     LDF     F2,F1;
122                     SUBF    (R0),F2;        X=1/2*SQRT(2)
```

```
123                             ADDF    (R0)+,F1;       X+1/2*SQRT(2)
124                             DIVF    F1,F2;          W=(X-ROOT2)/(X+ROOT2)
125                             LDF     F2,F1;
126                             MULF    F1,F1;          Y= W**2
127                     ;
128                             MOV     #3,R1;          COUNT OF CONSTS FOR POLYNOMIAL
129                             LDF     (R0)+,F0;       INITIALIZE ACCUMULATOR FOR POLYN
130                     XPD$3:  MULF    F1,F0;
131                             DEC     R1;             COUNT
132                             ADDF    (R0)+,F0;       F0:= Y*F0 + C(I)
133                             BGT     XPD$3;          LOOP
134                     ;
135                             MULF    F2,F0;
136                             ADDF    (R0)+,F0;       F0:= W*F0 - 1/2*LN(2)
137                             ADDF    F3,F0;          ADD SCALE FACTOR FOR EXPONENT
138                             TST     R4;             TEST ALOG10 FLAG
139                             BEQ     LGT$3;
140                             MULF    (R0)+,F0;       ALOG10:= ALOG*LOG10(E)
141                     ;
142                     LGT$3:  STF     F0,-(SP);       MOVE RESULT TO STACK
143                             MOV     (SP)+,R0;
144                             MOV     (SP)+,R1;       AND THENCE TO R0,R1
145                             RTS     R5;
146                     ERR$3:  JSR     R5,$ERR;        ERROR 4,10
147                             RTS     R5;             EXIT-NO STACK CLEANUP NECESSARY
148                             .BYTE   4
149                             .BYTE   10.
150                     ;       ORDER-DEPENDENT CONSTANTS FOR ROUTINE
151                     ;       R0 POINTS AT CURRENT CONSTANT IN FPU VERSION
152                     ;
153                     FCO$3:  .WORD   040061,071030;  LN(2)
154                     ;
155                             .WORD   040065,002363;  1/2*SQRT(2)
156                             .ENDC
157                     ;       CONSTANTS FOR POLYNOMIAL EXPANSION
158                     ;
159 3104 037632                 .WORD   037632,014525   ;.300974506
    3106 014525
160                     ;
161 3110 037714                 .WORD   037714,120036   ;.399659100
    3112 120036
162                     ;
163 3114 040052                 .WORD   040052,125332   ;.666669471
    3116 125332
164                     ;
165 3120 040400 CON$3:  .WORD   040400,000000   ;1.99999999
    3122 000000
166                             .IFDF   FPU
167                     ;       MORE ORDER-DEPENDENT CONSTANTS
168                             .WORD   137661,071030;  -1/2*LN(2)
169                     ;
170                             .WORD   037756,055731;  LOG10(E)
171                             .ENDC
172                             .ENDC
```

```
1                                       .TITLE  $ANT03
2                                       .IFDF   CND$4
3                                       .GLOBL  AINT,$INTR
4                               ;       AINT    V003A
5                               ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYARD, MASS.
6                               ;       AINT    FORTRAN AINT FUNCTION, CALLING SEQUENCE
7                               ;       JSR     R5,AINT
8                               ;       BR      A
9                               ;       .WORD   ADDRESS OF ARGUMENT
10                              ;A:
11                              ;       RETURNS SIGN OF ARG * GREATEST REAL INTEGER <= =
12                              ;       ABS(ARG) IN R0 AND R1.
13                              ;
14                              ;       $INTR   SAME FUNCTION AS AINT, BUT CALLED IN THE
15                              ;       POLISH MODE WITH THE ARGUMENT AND RETURN ON THE STACK.
16              000000                  R0=%0
17              000001                  R1=%1
18              000002                  R2=%2
19              000003                  R3=%3
20              000004                  R4=%4
21              000005                  R5=%5
22              000006                  SP=%6
23              000007                  PC=%7
24              177304                  MQ=177304
25              177314                  LSH=177314
26              000000                  F0=%0
27              000001                  F1=%1
28                                      .IFDF   FPU
29                              AINT:   .WORD   170001  ;;SETF
30                                      .WORD   17245,2         ;;LDF   @2(R5),F0       ;GET ARG
31                                      .WORD   171467,24       ;;MODF  ONE,F0  ;GET INTEGER PAR
32                                      .WORD   174146  ;;STF   F1,-(SP)
33                                      MOV     (SP)+,R0        ;POP TO USER REGS
34                                      MOV     (SP)+,R1
35                                      RTS     R5      ;RETURN
36                              ;
37                              $INTR:  .WORD   170001  ;;SETF
38                                      .WORD   172426  ;;LDF   (SP)+,F0        ;GET ARG
39                                      .WORD   171467,4        ;;MODF  ONE,F0  ;GET INTEGER PAR
40                                      .WORD   174146  ;;STF   F1,-(SP)
41                                      JMP     @(R4)+  ;RETURN
42                              ONE$4:  .WORD   040200,0        ;FLOATING 1.
43                                      .ENDC
44                                      .IFNDF  FPU
45  03124 016504 000002         AINT:   MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
46  03130 011400                        MOV     @R4,R0  ;GET HIGH ORDER ARGUMENT
47  03132 016401 000002                 MOV     2(R4),R1        ;LOW ORDER
48  03136 010702                        MOV     PC,R2   ;MAKE R2 NON 0
49  03140 000403                        BR      AI1$4
50  03142 005002                $INTR:  CLR     R2      ;MAKE R2 0
51  03144 012600                        MOV     (SP)+,R0        ;GET HIGH ORDER ARGUMENT
52  03146 012601                        MOV     (SP)+,R1        ;LOW ORDER
53  03150 010003                AI1$4:  MOV     R0,R3
54  03152 006103                        ROL     R3      ;DUMP SIGN
55  03154 105003                        CLRB    R3
```

```
56 03156 000303          SWAB    R3      ;GET EXPONENT
57 03160 162703          SUB     #230,R3 ;REMOVE EXCESS 200 AND CHECK RANGE
         000230
58 03164 002020          BGE     DNE$4   ;JUMP IF IT IS ALREADY AN INTEGER
59 03166 022703          CMP     #-30,R3
         177750
60 03172 002403          BLT     SHF$4   ;JUMP IF THERE IS WORK TO DO
61 03174 005000          CLR     R0      ;ARG IS < 1, SO RETURN 0
62 03176 005001          CLR     R1
63 03200 000412          BR      DNE$4
64 03202 010346 SHF$4:   MOV     R3,-(SP)        ;PUSH -SHIFT COUNT
65                       .IFNDF  EAE&MULDIV
66 03204 006000 ROR$4:   ROR     R0      ;SHIFT FRACTION
67 03206 006001          ROR     R1
68 03210 005203          INC     R3      ;COUNT LOOP
69 03212 002774          BLT     ROR$4   ;GO AGAIN
70 03214 012603          MOV     (SP)+,R3        ;GET COUNT BACK
71 03216 006301 ASL$4:   ASL     R1      ;SHIFT FRACTION BACK WITH 0'S
72 03220 006100          ROL     R0
73 03222 005203          INC     R3      ;COUNT LOOP AGAIN
74 03224 002774          BLT     ASL$4
75                       .ENDC
76                ;      EAE CODE
77                       .IFDF   EAE
78                       MOV     #MQ,R3  ;POINT TO MQ
79                       MOV     R1,@R3  ;INSERT ARG
80                       MOV     R0,-(R3)
81                       MOV     @SP,@#LSH       ;SHIFT RIGHT
82                       NEG     @SP     ;SET FOR LEFT
83                       MOV     (SP)+,@#LSH     ;SHIFT LEFT
84                       MOV     (R3)+,R0        ;RESULT TO REGS
85                       MOV     @R3,R1
86                       .ENDC
87                ;      MULDIV CODE
88                       .IFDF   MULDIV
89                       .WORD   073016  ;;ASHC  @SP,R0  ;SHIFT OUT FRACTION
90                       NEG     @SP     ;SET FOR LEFT SHIFT
91                       .WORD   073026  ;;ASHC  (SP)+,R0        ;SHIFT INTEGER P
92                       .ENDC
93 03226 005702 DNE$4:   TST     R2      ;CHECK ENTRY FLAG
94 03230 001401          BEQ     DN1$4   ;JUMP IF $INTR
95 03232 000205          RTS     R5      ;RETURN IF $AINT
96 03234 010146 DN1$4:   MOV     R1,-(SP)        ;PUSH RESULT
97 03236 010046          MOV     R0,-(SP)
98 03240 000134          JMP     @(R4)+  ;POLISH RETURN
99                       .ENDC
100                      .ENDC
```

```
1                                       .TITLE  $CMD02
2                                       .IFDF   CND$5
3                                       .GLOBL  $CMD
4                               ;       $CMD    THE DOUBLE COMPARE ROUTINE.
5                               ;
6                               ;       $CMD    V002A
7                               ;
8                               ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
9                               ;       CALLED IN THE POLISH MODE WITH THE TWO
10                              ;       COMPARANDS ON THE STACK:
11                              ;       FIRST IS AT 8(SP), SECOND IS @SP
12                              ;       FLUSH THE TWO COMPARANDS AND RETURN
13                              ;       THE FOLLOWING CONDITION CODES:
14                              ;       FIRST < SECOND  N=1, Z=0
15                              ;       FIRST = SECOND  N=0, Z=1
16                              ;       FIRST > SECOND  N=0, Z=0
17              000000                  R0=%0
18              000001                  R1=%1
19              000002                  R2=%2
20              000004                  R4=%4
21              000006                  SP=%6
22              000007                  PC=%7
23              000000                  F0=%0
24                                      .IFDF   FPU
25                              $CMD:   .WORD   170011  ;;SETD
26                                      .WORD   172426  ;;LDD   (SP)+,F0        ;GET SECOND ARG
27                                      .WORD   173426  ;;CMPD  (SP)+,F0        ;COMPARE
28                                      .WORD   170000  ;;CFCC  ;GET CONDITION CODES
29                                      JMP     @(R4)+
30                                      .ENDC
31                                      .IFNDF  FPU
32  03242  011700               $CMD:   MOV     @PC,R0  ;GET 00XXXXX   XXXX01 IN R0
33  03244  016601                       MOV     8.(SP),R1       ;GET HIGH ORDER FIRST ARG
           000010
34  03250  002004                       BGE     FPS$5   ;JUMP IF FIRST ARG +
35  03252  006300                       ASL     R0      ;FLAG FIRST ARG -
36  03254  012602                       MOV     (SP)+,R2        ;GET HIGH SECOND ARG
37  03256  002403                       BLT     SME$5   ;JUMP IF BOTH SIGNS -
38  03260  000422                       BR      NEG$5   ;JUMP IF FIRST - AND SECOND +
39  03262  012602               FPS$5:  MOV     (SP)+,R2
40  03264  002421                       BLT     PLS$5   ;JUMP IF FIRST + AND SECOND -
41  03266  020102               SME$5:  CMP     R1,R2   ;COMPARE MAGNITUDES
42  03270  001014                       BNE     OUT$5   ;JUMP IF DIFFERENT
43  03272  026616                       CMP     8.(SP),@SP
           000010
44  03276  001011                       BNE     OUT$5
45  03300  026666                       CMP     10.(SP),2(SP)
           000012
           000002
46  03306  001005                       BNE     OUT$5
47  03310  026666                       CMP     12.(SP),4(SP)
           000014
           000004
48  03316  001001                       BNE     OUT$5
49  03320  005000                       CLR     R0      ;FLAG =
50  03322  006000               OUT$5:  ROR     R0      ;SAVE C BIT AND TEST SECOND ARG -
51  03324  103401                       BCS     PLS$5   ;JUMP IF SECOND ARG +
```

```
52 03326 005400 NEG$5:  NEG     R0        ;REVERSE C BIT
53 03330 062706 PLS$5:  ADD     #14.,SP   ;POP ARGS
         000016
54 03334 005700         TST     R0        ;SET Z AND N BITS CORRECTLY
55 03336 000134         JMP     @(R4)+    ;RETURN TO CALLER
56                      .ENDC
57                      .ENDC
```

```
1                               .TITLE  $CMR02
2                               .IFDF   CND$6
3                               .GLOBL  $CMR
4                       ;       $CMR    THE REAL COMPARE ROUTINE.
5                       ;
6                       ;       $CMR    V002A
7                       ;
8                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYARD, MASS.
9                       ;       CALLED IN THE POLISH MODE WITH THE TWO
10                      ;       COMPARANDS ON THE STACK:
11                      ;       FIRST IS AT 4(SP), SECOND IS @SP
12                      ;       FLUSH THE TWO COMPARANDS AND RETURN
13                      ;       THE FOLLOWING CONDITION CODES:
14                      ;       FIRST < SECOND  N=1, Z=0
15                      ;       FIRST = SECOND  N=0, Z=1
16                      ;       FIRST > SECOND  N=0, Z=0
17              000000          R0=%0
18              000001          R1=%1
19              000002          R2=%2
20              000004          R4=%4
21              000006          SP=%6
22              000007          PC=%7
23              000000          F0=%0
24                              .IFDF   FPU
25                      $CMR:   .WORD   170001  ;;SETF
26                              .WORD   172426  ;;LDF    (SP)+,F0        ;GET SECOND ARG
27                              .WORD   173426  ;;CMPF   (SP)+,F0        ;COMPARE
28                              .WORD   170000  ;;CFCC   ;GET CONDITION CODES
29                              JMP     @(R4)+
30                              .ENDC
31                              .IFNDF  FPU
32  03340  011700        $CMR:   MOV     @PC,R0  ;GET 00XXXXX   XXXX01 IN R0
33  03342  016601                MOV     4(SP),R1        ;GET HIGH ORDER FIRST ARG
           000004
34  03346  002004                BGE     FPS$6   ;JUMP IF FIRST ARG +
35  03350  006300                ASL     R0      ;FLAG FIRST ARG =
36  03352  012602                MOV     (SP)+,R2        ;GET HIGH SECOND ARG
37  03354  002403                BLT     SME$6   ;JUMP IF BOTH SIGNS =
38  03356  000412                BR      NEG$6   ;JUMP IF FIRST = AND SECOND +
39  03360  012602        FPS$6:  MOV     (SP)+,R2
40  03362  002411                BLT     PLS$6   ;JUMP IF FIRST + AND SECOND =
41  03364  020102        SME$6:  CMP     R1,R2   ;COMPARE MAGNITUDES
42  03366  001004                BNE     OUT$6   ;JUMP IF DIFFERENT
43  03370  026616                CMP     4(SP),@SP       ;COMPARE LOW ORDER
           000004
44  03374  001001                BNE     OUT$6   ;JUMP IF DIFFERENT
45  03376  005000                CLR     R0      ;FLAG =
46  03400  006000        OUT$6:  ROR     R0      ;SAVE C BIT AND TEST SECOND ARG =
47  03402  103401                BCS     PLS$6   ;JUMP IF SECOND ARG +
48  03404  005400        NEG$6:  NEG     R0      ;REVERSE C BIT
49  03406  062706        PLS$6:  ADD     #6,SP   ;POP ARGS
           000006
50  03412  005700                TST     R0      ;SET Z AND N BITS CORRECTLY
51  03414  000134                JMP     @(R4)+  ;RETURN TO CALLER
52                              .ENDC
53                              .ENDC
```

```
1                               .TITLE  $DBL02
2                               .IFDF   CND$7
3                       ;
4                       ;       DBLE    V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8
9                               .GLOBL  DBLE
10                      ;       THE FORTRAN DBLE FUNCTION
11                      ;       CALLING SEQUENCE:
12                      ;       JSR     R5,DBLE
13                      ;       BR      A
14                      ;       .WORD   ARGUMENT ADDRESS
15                      ;A:
16                      ;       RETURNS THE DOUBLE PRESICION EQUIVALENT
17                      ;       OF THE REAL ARGUMENT IN R0 - R3.
18                      ;
19              000000          R0=%0
20              000001          R1=%1
21              000002          R2=%2
22              000003          R3=%3
23              000005          R5=%5
24  03416 016502 DBLE:  MOV     2(R5),R2        ;GET ARG ADDRESS
                000002
25  03422 012200        MOV     (R2)+,R0        ;GET HIGH ORDER
26  03424 011201        MOV     @R2,R1  ;GET LOW ORDER
27  03426 005002        CLR     R2      ;CLEAR LOWEST ORDER
28  03430 005003        CLR     R3
29  03432 000205        RTS     R5      ;RETURN TO CALLER
30                              .ENDC
```

```
1                               .TITLE  $DCI01
2                               .IFDF   CND$8
3                       ;
4                       ;       $DCI    V001A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  $DCI,$RCI
9                       ;       $DCI --- ASCII TO DOUBLE CONVERSION.
10                      ;       $RCI --- ASCII TO REAL CONVERSION.
11                      ;       CALLING SEQUENCE:
12                      ;       PUSH ADDRESS OF START OF FIELD
13                      ;       PUSH LENGTH OF FIELD
14                      ;       PUSH FORMAT SCALE D FROM W.D
15                      ;       PUSH P FORMAT SCALE
16                      ;       JSR     PC,$DCI (OR $RCI)
17
18              000000          R0=%0
19              000001          R1=%1
20              000002          R2=%2
21              000003          R3=%3
22              000004          R4=%4
23              000005          R5=%5
24              000006          SP=%6
25              000007          PC=%7
26              000000  NUMEND=0
27              000002  POINTL=2
28              000004  DIGITS=4
29              000006  BEXP=6
30              000010  ESIGN=8.
31              000012  SIGN=10.
32              000014  EEXP=12.
33              000036  P=30.
34              000040  D=32.
35              000052  ERR=26.
36              000042  LENGTH=34.
37              000042  TEMP=LENGTH
38              000036  RESULT=P
39              000044  START=36.
40              000044  END=START
41  03434 005046 $RCI:  CLR     -(SP)   ;CLEAR ERROR FLAG
42  03436 005216        INC     @SP     ;SET REAL CONVERSION FLAG
43  03440 000401        BR      CNV$8
44  03442 005046 $DCI:  CLR     -(SP)   ;CLEAR ERROR FLAG AND SET FOR DOUBLE
45  03444 010046 CNV$8: MOV     R0,-(SP)
46  03446 010146        MOV     R1,-(SP)
47  03450 010246        MOV     R2,-(SP)
48  03452 010346        MOV     R3,-(SP)
49  03454 010446        MOV     R4,-(SP)
50  03456 010546        MOV     R5,-(SP)
51  03460 005046        CLR     -(SP)   ;CLEAR EXP
52  03462 005046        CLR     -(SP)   ;CLEAR SIGN
53  03464 005046        CLR     -(SP)   ;CLEAR ESIGN
54  03466 012746        MOV     #65.,-(SP)      ;INITIALIZE BEXP
          000101
55  03472 012746        MOV     #18.,-(SP)      ;INITIALIZE MAX DIGITS
          000022
```

```
56 03476 005046          CLR     -(SP)    ;CLEAR POINTL
57 03500 005046          CLR     -(SP)    ;CLEAR NUMEND
58 03502 016605          MOV     START(SP),R5     ;GET FIELD START ADDRESS
         000044
59 03506 066566          ADD     LENGTH(SP),END(SP)       ;POINT TO END +1
         000042
         000044
60 03514 005000          CLR     R0       ;CLEAR NUMERIC WORK SPACE
61 03516 005001          CLR     R1
62 03520 005002          CLR     R2
63 03522 005003          CLR     R3
64 03524 112504 SCN$8:   MOVB    (R5)+,R4         ;GET NEXT INPUT CHARACTER
65 03526 042704          BIC     #177600,R4
         177600
66 03532 120427          CMPB    R4,#'    ;TEST FOR BLANK
         000040
67 03536 001005          BNE     SGS$8    ;IF NOT BLANK LOOK FOR + OR -
68 03540 020566          CMP     R5,START(SP)     ;CHECK END OF FIELD
         000044
69 03544 002767          BLT     SCN$8    ;IF NOT DONE GO GET NEXT
70 03546 000167          JMP     ZER$8    ;ENTIRE FIELD IS BLANK
         000326
71 03552 120427 SGS$8:   CMPB    R4,#'+   ;CHECK FOR + SIGN
         000053
72 03556 001455          BEQ     FLD$8    ;IF FOUND IGNORE IT
73 03560 120427          CMPB    R4,#'-   ;CHECK FOR - SIGN
         000055
74 03564 001013          BNE     NCK$8    ;IF NOT FOUND CHECK NUMERICS
75 03566 005266          INC     SIGN(SP)         ;SET - SIGN FLAG
         000012
76 03572 000447          BR      FLD$8
77 03574 112504 NXT$8:   MOVB    (R5)+,R4         ;GET NEXT INPUT CHARACTER
78 03576 042704          BIC     #177600,R4
         177600
79 03602 120427          CMPB    R4,#'    ;CHECK FOR BLANKS
         000040
80 03606 001002          BNE     NCK$8
81 03610 012704          MOV     #'0,R4   ;TREAT BLANK AS 0
         000060
82 03614 120427 NCK$8:   CMPB    R4,#'0   ;CHECK FOR LEGAL CHARACTER
         000060
83 03620 002514          BLT     PCK$8    ;CHECK FOR DECIMAL POINT
84 03622 001010          BNE     NNZ$8    ;JUMP IF NOT 0
85 03624 005700          TST     R0       ;CHECK TO SEE IF ANY NON-ZERO DIGITS FOU
86 03626 001006          BNE     NNZ$8
87 03630 005701          TST     R1
88 03632 001004          BNE     NNZ$8
89 03634 005702          TST     R2
90 03636 001002          BNE     NNZ$8
91 03640 005703          TST     R3
92 03642 001423          BEQ     FLD$8
93 03644 120427 NNZ$8:   CMPB    R4,#'9
         000071
94 03650 003121          BGT     EXC$8    ;CHECK FOR EXPONENT
95 03652 005366          DEC     DIGITS(SP)       ;COUNT AS A SIGNIFICANT DIGIT
         000004
96 03656 002003          BGE     A2I$8    ;JUMP IF WE CAN USE THIS DIGIT
```

```
97 03660 005266          INC     EEXP(SP)        ;COMPENSATE FOR SKIPPED DIGIT
         000014
98 03664 000412          BR      FLD$8
99 03666 162704 A2I$8:   SUB     #60,R4  ;CONVERT ASCII TO INTEGER
         000060
100 3672 004767          JSR     PC,ML5$8        ;MULTIPLY BY 5
         001044
101 3676 004767          JSR     PC,LFT$8        ;DOUBLE RESULT FOR 10
         001106
102 3702 060403          ADD     R4,R3   ;ADD IN CURRENT DIGIT
103 3704 005502          ADC     R2
104 3706 005501          ADC     R1
105 3710 005500          ADC     R0      ;END OF CONVERT FOR THIS DIGIT
106 3712 020566 FLD$8:   CMP     R5,END(SP)      ;CHECK FOR END OF FIELD
         000044
107 3716 002726          BLT     NXT$8
108 3720 010516          MOV     R5,@SP  ;POINTER TO LAST NUMERIC TO NUMEND
109 3722 005700 SCL$8:   TST     R0
110 3724 001006          BNE     SC1$8   ;JUMP IF NUMBER NOT 0
111 3726 005701          TST     R1
112 3730 001004          BNE     SC1$8
113 3732 005702          TST     R2
114 3734 001002          BNE     SC1$8
115 3736 005703          TST     R3
116 3740 001457          BEQ     ZER$8   ;INPUT NUMBER IS 0
117 3742 021503 SC1$8:   CMP     @SP,R5  ;CHECK NUMEND
118 3744 001003          BNE     NOP$8   ;JUMP IF THERE WAS AN EXPONENT FIELD
119 3746 166666          SUB     P(SP),EEXP(SP)  ;USE THE FORMAT P SCALE
         000036
         000014
120 3754 005766 NOP$8:   TST     POINTL(SP)
         000002
121 3760 001002          BNE     PNT$8   ;JUMP IF THERE WAS A DECIMAL POINT
122 3762 016616          MOV     D(SP),@SP       ;USE THE D SCALE
         000040
123 3766 166616 PNT$8:   SUB     POINTL(SP),@SP
         000002
124 3772 161666          SUB     @SP,EEXP(SP)    ;FORM COMPLETE DECIMAL EXPONENT
         000014
125 3776 003003          BGT     MUL$8   ;MULTIPLY BY 10**EXP
126 4000 002543          BLT     DIV$8   ;JUMP IF DECIMAL EXPONENT IS NEG
127 4002 000167          JMP     FLT$8   ;JUMP IF EXP IS 0
         000446
128 4006 020027 MUL$8:   CMP     R0,#31462
         031462
129 4012 101011          BHI     MDV$8   ;JUMP IF FRACT TOO BIG TO MULT BY 5
130 4014 004767          JSR     PC,ML5$8        ;FRACT=5*FRACT
         000722
131 4020 005266          INC     BEXP(SP)        ;TIMES 2
         000006
132 4024 005366 D10$8:   DEC     EEXP(SP)        ;OVER 10
         000014
133 4030 003366          BGT     MUL$8   ;JUMP IF MORE DECIMAL EXPONENT
134 4032 000167          JMP     FLT$8   ;DECIMAL EXPONENT GONE
         000416
135 4036 004767 MDV$8:   JSR     PC,M54$8        ;MULTIPLY BY 5/4
         000632
```

```
136 4042 062766          ADD     #3,BEXP(SP)     ;TIMES 8
         000003
         000006
137 4050 000765          BR      D10$8   ;GO DIVIDE BY 10
138 4052 120427 PCK$8:   CMPB    R4,#'.
         000056
139 4056 001006          BNE     ERR$8   ;JUMP IF NOT A DECIMAL POINT
140 4060 005766 PTF$8:   TST     POINTL(SP)
         000002
141 4064 001003          BNE     ERR$8   ;JUMP IF A . ALREADY ENCOUNTERED
142 4066 010566          MOV     R5,POINTL(SP)   ;SAVE A POINTER TO THE . +1
         000002
143 4072 000707          BR      FLD$8   ;GO FOR NEXT CHARACTER
144 4074 105166 ERR$8:   COMB    ERF+1(SP)       ;FLAG ERROR
         000033
145 4100 005000 ZER$8:   CLR     R0      ;RESULT IS 0
146 4102 005001          CLR     R1
147 4104 005002          CLR     R2
148 4106 005003          CLR     R3
149 4110 000167          JMP     STR$8   ;GO PUSH RESULT AND RETURN
         000450
150 4114 120427 EXC$8:   CMPB    R4,#'E
         000105
151 4120 001403          BEQ     EXT$8   ;JUMP IF E
152 4122 120427          CMPB    R4,#'D
         000104
153 4126 001352          BNE     ERR$8   ;IF NOT E OR D THEN ERROR
154 4130 010516 EXT$8:   MOV     R5,@SP  ;SAVE POINTER TO END OF NUM +1
155 4132 005316          DEC     @SP     ;DECREMENT NUMEND
156 4134 010366          MOV     R3,TEMP(SP)
         000042
157 4140 005003          CLR     R3
158 4142 020566          CMP     R5,END(SP)
         000044
159 4146 002352          BGE     ERR$8   ;JUMP IF NO ROOM FOR EXP
160 4150 112504          MOVB    (R5)+,R4
161 4152 042704          BIC     #177600,R4
         177600
162 4156 120427          CMPB    R4,#'+  ;CHECK FOR +EXP
         000053
163 4162 001405          BEQ     EF1$8
164 4164 120427          CMPB    R4,#'-  ;CHECK FOR -EXP
         000055
165 4170 001010          BNE     ENM$8   ;GO CHECK FOR NUMERIC
166 4172 005266          INC     ESIGN(SP)       ;FLAG EXPONENT NEGATIVE
         000010
167 4176 020566 EF1$8:   CMP     R5,END(SP)
         000044
168 4202 002334          BGE     ERR$8
169 4204 112504 EF2$8:   MOVB    (R5)+,R4        ;GET NEXT CHAR
170 4206 042704          BIC     #177600,R4
         177600
171 4212 120427 ENM$8:   CMPB    R4,#'   ;CHECK FOR BLANK
         000040
172 4216 001002          BNE     EN1$8
173 4220 012704          MOV     #'0,R4  ;TREAT BLANK AS 0
         000060
```

```
174 4224 120427 EN1$8:  CMPB    R4,#'0
         000060
175 4230 002721         BLT     ERR$8
176 4232 120427         CMPB    R4,#'9
         000071
177 4236 003315         BGT     ERR$8   ;NOT A VALID CHAR
178 4240 162704         SUB     #60,R4  ;CONVERT ASCII TO INTEGER
         000060
179 4244 006303         ASL     R3      ;X=10*X+D
180 4246 060304         ADD     R3,R4
181 4250 006303         ASL     R3
182 4252 006303         ASL     R3
183 4254 060403         ADD     R4,R3   ;END OF ABOVE COMMENT
184 4256 020566         CMP     R5,END(SP)
         000044
185 4262 002750         BLT     EF2$8   ;JUMP IF MORE FIELD TO GO
186 4264 005766         TST     ESIGN(SP)       ;CHECK EXPONENT SIGN
         000010
187 4270 001401         BEQ     EN2$8   ;JUMP IF IT IS +
188 4272 005403         NEG     R3      ;MAKE USER EXPONENT -
189 4274 060366 EN2$8:  ADD     R3,EEXP(SP)     ;GET COMPLETE DECIMAL EXPONENT
         000014
190 4300 016603         MOV     TEMP(SP),R3
         000042
191 4304 000167         JMP     SCL$8   ;GO SCALE THE NUMERIC PART
         177412
192 4310 005700 DIV$8:  TST     R0
193 4312 002405         BLT     DV1$8   ;JUMP IF FRACT LEFT JUSTIFIED
194 4314 005366 DV2$8:  DEC     BEXP(SP)        ;LEFT JUSTIFY NUMERIC BITS
         000006
195 4320 004767         JSR     PC,LFT$8
         000464
196 4324 100373         BPL     DV2$8
197 4326 012704 DV1$8:  MOV     #16.,R4 ;SET FOR SIXTEEN ITERATIONS
         000020
198 4332 004767         JSR     PC,RIT$8
         000464
199 4336 010346         MOV     R3,-(SP)
200 4340 010246         MOV     R2,-(SP)
201 4342 010146         MOV     R1,-(SP)        ;INITIALIZE QUOTIENT
202 4344 010046         MOV     R0,-(SP)
203 4346 004767 DV3$8:  JSR     PC,RIT$8
         000450
204 4352 000241         CLC
205 4354 004767         JSR     PC,RIT$8
         000442
206 4360 012705         MOV     #2,R5
         000002
207 4364 000241         CLC
208 4366 004767 DV4$8:  JSR     PC,RIT$8
         000430
209 4372 066603         ADD     6(SP),R3
         000006
210 4376 005502         ADC     R2
211 4400 005501         ADC     R1
212 4402 005500         ADC     R0
213 4404 066602         ADD     4(SP),R2
```

```
                000004
214 4410 005501         ADC     R1
215 4412 005500         ADC     R0
216 4414 066601         ADD     2(SP),R1
                000002
217 4420 005500         ADC     R0
218 4422 061600         ADD     @SP,R0
219 4424 005305         DEC     R5      ;COUNT TWICE
220 4426 003357         BGT     DV4$8
221 4430 005304         DEC     R4
222 4432 003345         BGT     DV3$8
223 4434 062706         ADD     #8.,SP  ;POP DIVIDEND
                000010
224 4440 162766         SUB     #3,BEXP(SP)
                000003
                000006
225 4446 005266         INC     EEXP(SP)        ;BUMP DECIMAL EXPONENT
                000014
226 4452 002716         BLT     DIV$8   ;JUMP IF MORE TO DO
227 4454 005366 FLT$8:  DEC     BEXP(SP)        ;POST NORMALIZE THE RESULT
                000006
228 4460 004767         JSR     PC,LFT$8
                000324
229 4464 103373         BCC     FLT$8
230 4466 062766         ADD     #200,BEXP(SP)   ;SET EXCESS 128
                000200
                000006
231 4474 003475         BLE     UND$8   ;NUMBER TOO SMALL TO REPRESENT
232 4476 026627         CMP     BEXP(SP),#377
                000006
                000377
233 4504 003071         BGT     OVR$8   ;JUMP IF NUMBER TOO BIG
234 4506 105003         CLRB    R3
235 4510 150203         BISB    R2,R3
236 4512 000303         SWAB    R3
237 4514 105002         CLRB    R2
238 4516 150102         BISB    R1,R2
239 4520 000302         SWAB    R2
240 4522 105001         CLRB    R1
241 4524 150001         BISB    R0,R1   ;MOVE OUT LOWEST ORDER BITS
242 4526 000301         SWAB    R1
243 4530 105000         CLRB    R0
244 4532 156600         BISB    BEXP(SP),R0     ;INSERT THE BINARY EXPONENT
                000006
245 4536 000300         SWAB    R0      ;PUT IN THE RIGHT ORDER
246 4540 006066         ROR     SIGN(SP)        ;TEST THE ARITHMETIC SIGN
                000012
247 4544 004767         JSR     PC,RLT$8        ;INSERT IN RESULT
                000252
248 4550 005503         ADC     R3
249 4552 005502         ADC     R2
250 4554 005501         ADC     R1      ;FINAL ROUND
251 4556 005500         ADC     R0
252 4560 102443         BVS     OVR$8   ;JUMP IF OVERFLOW
253 4562 103442         BCS     OVR$8
254 4564 105766 STR$8:  TSTB    ERF(SP) ;TEST REAL/DOUBLE FLAG
                000032
```

```
101 5264 004767 ML1$9:  JSR     PC,M45$9        ;GET 4/5 FRACTION
         000646
102 5270 005266         INC     EEXP(SP)        ;MULTIPLY BY 10
         000006
103 5274 162766         SUB     #3,BEXP(SP)     ;AND DIVIDE BY 8
         000003
         000004
104 5302 003356         BGT     MUL$9   ;JUMP IF BINARY EXPONENT STILL POS.
105 5304 001424         BEQ     NOM$9   ;JUMP IF EXPONENT GONE NOW
106 5306 020127 DIV$9:  CMP     R1,#146314      ;BINARY EXPONENT IS NEGATIVE
         146314
107 5312 103014         BHIS    DV1$9   ;JUMP IF NO ROOM FOR 5/4 FRACTION
108 5314 026627         CMP     BEXP(SP),#-3
         000004
         177775
109 5322 003010         BGT     DV1$9   ;JUMP IF NOT ENOUGH BINARY EXP LEFT
110 5324 004767         JSR     PC,M54$9        ;MULTIPLY FRACTION BY 5/4
         000520
111 5330 005366         DEC     EEXP(SP)        ;DIVIDE BY 10
         000006
112 5334 062766         ADD     #2,BEXP(SP)     ;MULTIPLY BY 4
         000002
         000004
113 5342 000402         BR      DV2$9
114 5344 004767 DV1$9:  JSR     PC,RIT$9        ;DIVIDE BY 2
         001264
115 5350 005266 DV2$9:  INC     BEXP(SP)        ;MULTIPLY BY 2
         000004
116 5354 001354         BNE     DIV$9   ;HIT IT AGAIN IF BIN.EXP. NOT GONE
117                     ;       AT THIS POINT THE BINARY EXPONENT IS 0
118                     ;       AND THE FRACTION IS IN R1, R2, R3 AND R4.
119 5356 005000 NOM$9:  CLR     R0      ;CLEAR OVERFLOW ACCUMULATOR
120 5360 004767 NO1$9:  JSR     PC,M54$9        ;MULTIPLY FRACTION BY 5/4
         000464
121 5364 004767         JSR     PC,ML8$9        ;AND NOW BY 8
         000656
122 5370 005700         TST     R0
123 5372 001003         BNE     NOD$9   ;JUMP IF AN INTEGER PART RESULTS
124 5374 005366         DEC     EEXP(SP)        ;DECREMENT EXPONENT
         000006
125 5400 000767         BR      NO1$9   ;GO AGAIN TO GET AN INTEGER PART
126                     ;       AT THIS POINT THE MOST SIGNIFICANT NON ZERO DIGIT IS IN
127 5402 105766 NOD$9:  TSTB    TYPE(SP)        ;TEST CONVERSON TYPE
         000014
128 5406 001424         BEQ     FFT$9   ;JUMP IF F FORMAT
129 5410 106066         RORB    TYPE(SP)
         000014
130 5414 103114         BCC     EFT$9   ;JUMP IF E FORMAT OR D FORMAT
131 5416 005766         TST     EEXP(SP)        ;G FORMAT
         000006
132 5422 002511         BLT     EFT$9   ;JUMP IF RESULT <.1
133 5424 026666         CMP     EEXP(SP),D(SP)
         000006
         000022
134 5432 003105         BGT     EFT$9   ;JUMP IF RESULT >10**D
135 5434 105066         CLRB    TYPE(SP)        ;MAKE TYPE F INSTEAD OF G
         000014
```

```
136 5440 162766          SUB     #4,L(SP)          ;LEAVE ROOM FOR BLANKS ON RIGHT
         000004
         000024
137 5446 166666          SUB     EEXP(SP),D(SP)    ;DECREASE D BY # OF DIGITS LEFT
         000006
         000022
138 5454 005066          CLR     P(SP)    ;SUSPEND P SCALE
         000020
139 5460 016605 FFT$9:   MOV     EEXP(SP),R5       ;F FORMAT
         000006
140 5464 066605 FFE$9:   ADD     D(SP),R5
         000022
141 5470 066605          ADD     P(SP),R5
         000020
142 5474 004767          JSR     PC,RUD$9          ;ROUND BY ADDING 5*10***-P-D-E
         000640
143 5500 016605          MOV     L(SP),R5
         000024
144 5504 166605          SUB     D(SP),R5
         000022
145 5510 105766          TSTB    TYPE(SP)
         000014
146 5514 001013          BNE     FF5$9    ;JUMP IF NOT F CONVERSION
147 5516 066666          ADD     EEXP(SP),P(SP)    ;COMBINE P AND EXP
         000006
         000020
148 5524 003407          BLE     FF5$9    ;JUMP IF THERE IS NO INTEGER PART IN RES
149 5526 166605          SUB     P(SP),R5
         000020
150 5532 162705          SUB     #2,R5    ;SIGN SLOT IS S+L-D-E-P-2
         000002
151 5536 004767          JSR     PC,ISN$9          ;INSERT SIGN AND CHECK WIDTH
         000744
152 5542 000416          BR      FF3$9    ;JUMP TO INSERT IDGITS
153 5544 162705 FF5$9:   SUB     #3,R5    ;SIGN SLOT IS S+L-D-3
         000003
154 5550 004767          JSR     PC,ISN$9          ;GO INSERT SIGN AND CHECK WIDTH
         000732
155 5554 112725          MOVB    #'0,(R5)+         ;INSERT LEADING 0
         000060
156 5560 112725          MOVB    #'.,(R5)+         ;INSERT .
         000056
157 5564 020566 FF4$9:   CMP     R5,L(SP)          ;CHECK FIELD END
         000024
158 5570 103003          BHIS    FF3$9    ;JUMP IF FIELD FULL
159 5572 112725          MOVB    #'0,(R5)+         ;PUT IN ANOTHER LEADING ZERO
         000060
160 5576 000772          BR      FF4$9
161 5600 016605 FF3$9:   MOV     L(SP),R5
         000024
162 5604 166605          SUB     D(SP),R5
         000022
163 5610 005305          DEC     R5       ;LOCATION FOR .
164 5612 010566          MOV     R5,PUINT(SP)      ;REMEMBER ITS LOCATION
         000002
165 5616 005766          TST     P(SP)
         000020
```

```
166 5622 003001          BGT     FF6$9
167 5624 005205          INC     R5      ;POINT TO SLOT FOR FIRST NON-ZERO DIGIT
168 5626 166605 FF6$9:   SUB     P(SP),R5
         000020
169 5632 004767          JSR     PC,DGS$9        ;GO INSERT ALL DIGITS
         000714
170 5636 105766          TSTB    TYPE(SP)
         000014
171 5642 001467          BEQ     DNE$9   ;ALL THROUGH IF F FORMAT
172 5644 000433          BR      EFE$9   ;GO FINISH E FORMAT
173 5646 162766 EFT$9:   SUB     #4,L(SP)        ;MAKE ROOM FOR E FIELD
         000004
         000024
174 5654 005005          CLR     R5
175 5656 005766          TST     P(SP)
         000020
176 5662 003700          BLE     FFE$9   ;PROCESS AS F FMT & RETURN TO EFMTE
177 5664 016605          MOV     D(SP),R5        ;GET ROUNDING FACTOR
         000022
178 5670 066605          ADD     P(SP),R5;       ALLOW FOR P SCALE
         000020
179 5674 004767          JSR     PC,RUD$9        ;GO USE IT
         000440
180 5700 016605          MOV     L(SP),R5        ;POINT TO SIGN SLOT
         000024
181 5704 166605          SUB     D(SP),R5
         000022
182 5710 005305          DEC     R5;             POINT SLOT = L-D-1
183 5712 010566          MOV     R5,POINT(SP);   SAVE LOCATION FOR .
         000002
184 5716 166605          SUB     P(SP),R5;
         000020
185 5722 005305          DEC     R5;             SIGN SLOT = L-D-P-2
186 5724 004767          JSR     PC,ISN$9        ;GO CHECK WIDTH AND INSERT SIGN
         000556
187 5730 004767          JSR     PC,DGS$9        ;GO PROCESS ALL DIGITS
         000616
188 5734 166666 EFE$9:   SUB     P(SP),EEXP(SP)  ;CORRECT EXPONENT FOR P
         000020
         000006
189 5742 016603          MOV     L(SP),R3
         000024
190 5746 116623          MOVB    TYPE+1(SP),(R3)+        ;MOVE OUT E OR D
         000015
191 5752 016604          MOV     EEXP(SP),R4
         000006
192 5756 002004          BGE     EXP$9   ;JUMP IF EXPONENT POSITIVE
193 5760 005404          NEG     R4      ;GET ABSOLUTE VALUE
194 5762 112723          MOVB    #'-,(R3)+       ;INSERT -
         000055
195 5766 000402          BR      EX1$9
196 5770 112723 EXP$9:   MOVB    #' ,(R3)+       ;INSERT BLANK FOR +
         000040
197 5774 112713 EX1$9:   MOVB    #'0,@R3 ;CLEAR TENS DIGIT
         000060
198 6000 162704 EX3$9:   SUB     #10.,R4 ;TEST FOR TENS
         000012
```

```
199 6004 002402          BLT     EX2$9
200 6006 105213          INCB    @R3     ;ACCUMULATE TENS
201 6010 000773          BR      EX3$9
202 6012 062704 EX2$9:   ADD     #72,R4  ;GET POSITIVE UNITS
         000072
203 6016 110463          MOVB    R4,1(R3)        ;MOVE UNITS OUT
         000001
204 6022 062706 DNE$9:   ADD     #8.,SP
         000010
205 6026 012605          MOV     (SP)+,R5
206 6030 012604          MOV     (SP)+,R4
207 6032 012666          MOV     (SP)+,6(SP)     ;MOVE FLAG AND RETURN UP
         000006
208 6036 012666          MOV     (SP)+,6(SP)
         000006
209 6042 022626          CMP     (SP)+,(SP)+     ;FLUSH JUNK
210 6044 006126          ROL     (SP)+   ;SET C BIT IF ERROR
211 6046 000207          RTS     PC      ;RETURN TO CALLER
212
213              ;       MULTIPLY CONTENTS OF R1 ... R4 BY 5/4,
214              ;       ANY OVERFLOW GOES INTO R0.
215 6050 010146 M54$9:   MOV     R1,-(SP)        ;5/4X=X+X/4
216 6052 010246          MOV     R2,-(SP)
217 6054 010346          MOV     R3,-(SP)
218 6056 010446          MOV     R4,-(SP)
219 6060 004767          JSR     PC,RIT$9        ;X/2
         000550
220 6064 004767          JSR     PC,RIT$9        ;X/4
         000544
221 6070 005504          ADC     R4      ;ROUND
222 6072 005503          ADC     R3
223 6074 005502          ADC     R2
224 6076 005501          ADC     R1
225 6100 062604          ADD     (SP)+,R4
226 6102 005503          ADC     R3
227 6104 005502          ADC     R2
228 6106 005501          ADC     R1
229 6110 005500          ADC     R0
230 6112 062603          ADD     (SP)+,R3
231 6114 005502          ADC     R2
232 6116 005501          ADC     R1
233 6120 005500          ADC     R0
234 6122 062602          ADD     (SP)+,R2
235 6124 005501          ADC     R1
236 6126 005500          ADC     R0
237 6130 062601          ADD     (SP)+,R1
238 6132 005500          ADC     R0
239 6134 000207          RTS     PC      ;RETURN TO CALLER
240              ;
241              ;
242 6136 012705 M45$9:   MOV     #16.,R5 ;MULTIPLY R1...R4 BY 4/5
         000020
243 6142 004767          JSR     PC,RIT$9
         000466
244 6146 010446          MOV     R4,-(SP)
245 6150 010346          MOV     R3,-(SP)
246 6152 010246          MOV     R2,-(SP)
```

```
247 6154 010146          MOV     R1,-(SP)
248 6156 004767 M51$9:   JSR     PC,RIT$9
         000452
249 6162 004767          JSR     PC,RIT$9
         000446
250 6166 012700          MOV     #2,R0
         000002
251 6172 004767 M52$9:   JSR     PC,RIT$9
         000436
252 6176 066604          ADD     6(SP),R4
         000006
253 6202 005503          ADC     R3
254 6204 005502          ADC     R2
255 6206 005501          ADC     R1
256 6210 066603          ADD     4(SP),R3
         000004
257 6214 005502          ADC     R2
258 6216 005501          ADC     R1
259 6220 066602          ADD     2(SP),R2
         000002
260 6224 005501          ADC     R1
261 6226 061601          ADD     @SP,R1
262 6230 005300          DEC     R0
263 6232 003357          BGT     M52$9
264 6234 005305          DEC     R5
265 6236 003347          BGT     M51$9
266 6240 062706          ADD     #8.,SP  ;FLUSH MULTIPLIER
         000010
267 6244 000207          RTS     PC
268                ;
269                ;       MULTIPLY THE CONTENTS OF R0 ... R4 BY 8.
270                ;       NO OVERFLOW IS ANTICIPATED
271 6246 010546 ML8$9:   MOV     R5,-(SP)
272 6250 012705          MOV     #3,R5
         000003
273 6254 006304 M81$9:   ASL     R4
274 6256 006103          ROL     R3
275 6260 006102          ROL     R2
276 6262 006101          ROL     R1
277 6264 006100          ROL     R0
278 6266 005305          DEC     R5
279 6270 003371          BGT     M81$9
280 6272 012605          MOV     (SP)+,R5
281 6274 000207          RTS     PC
282 6276 005726 ERR$9:   TST     (SP)+   ;POP RETURN
283 6300 016603          MOV     S(SP),R3        ;POINT TO FIELD BEGIN
         000026
284 6304 016604          MOV     L(SP),R4        ;GET FIELD END +1
         000024
285 6310 105766          TSTB    TYPE(SP)        ;CHECK IF END MODIFIED
         000014
286 6314 001402          BEQ     STS$9   ;NO, THIS IS F FORMAT
287 6316 062704          ADD     #4,R4   ;PUT BACK EXPONENT SPACE
         000004
288 6322 112723 STS$9:   MOVB    #'*,(R3)+       ;FILL FIELD WITH *
         000052
289 6326 020304          CMP     R3,R4
```

```
290 6330 103774          BLO     STS$9   ;JUMP IF MORE TO GO
291 6332 005166          COM     TYPE(SP)        ;FLAG ERROR
         000014
292 6336 000631          BR      DNE$9
293                      ;
294                      ;       ROUND THE CONTENTS OF R0 ... R4 TO THE PRECISION
295                      ;       SPECIFIED BY R5.
296                      ;       THIS ROUTINE IS SHORTER THAN THE TABLE THAT
297                      ;       OTHERWISE WOULD BE NEEDED.
298 6340 020527 RUD$9:   CMP     R5,#20.
         000024
299 6344 003054          BGT     RU1$9   ;JUMP IF NOT WORTH ROUNDING
300 6346 010566          MOV     R5,BEXP+0+2(SP) ;SAVE ROUNDING PRECISION IN TEMP
         000006
301 6352 001452          BEQ     RU3$9   ;JUMP IF ROUND IS TO LEADING DIGIT
302 6354 002450          BLT     RU1$9   ;JUMP IF NO ROUNDING TO DO
303 6356 010046          MOV     R0,-(SP)
304 6360 010146          MOV     R1,-(SP)
305 6362 010246          MOV     R2,-(SP)
306 6364 010346          MOV     R3,-(SP)
307 6366 010446          MOV     R4,-(SP)
308 6370 012701          MOV     #100000,R1      ;INSERT .5
         100000
309 6374 005002          CLR     R2
310 6376 005003          CLR     R3
311 6400 005004          CLR     R4
312 6402 005366 RDF$9:   DEC     BEXP+0+2+10.(SP)        ;COUNT PRECISION
         000020
313 6406 001411          BEQ     RDD$9   ;JUMP IF DONE
314 6410 004767          JSR     PC,M45$9        ;MULTIPLY BY 4/5
         177522
315 6414 004767          JSR     PC,RIT$9
         000214
316 6420 004767          JSR     PC,RIT$9
         000210
317 6424 004767          JSR     PC,RIT$9        ;DIVIDE BY 8
         000204
318 6430 000764          BR      RDF$9   ;GO CHECK IF DONE WITH FACTOR
319 6432 005000 RDD$9:   CLR     R0
320 6434 062604          ADD     (SP)+,R4        ;ADD FRACTION TO RND FACTOR
321 6436 005503          ADC     R3
322 6440 005502          ADC     R2
323 6442 005501          ADC     R1
324 6444 062603          ADD     (SP)+,R3
325 6446 005502          ADC     R2
326 6450 005501          ADC     R1
327 6452 062602          ADD     (SP)+,R2
328 6454 005501          ADC     R1
329 6456 062601          ADD     (SP)+,R1
330 6460 005500          ADC     R0
331 6462 062600          ADD     (SP)+,R0
332 6464 022700 RU2$9:   CMP     #10.,R0
         000012
333 6470 003002          BGT     RU1$9   ;JUMP IF NO OVERFLOW
334 6472 005266          INC     EEXP+2(SP)      ;BUMP DECIMAL EXPONENT
         000010
335 6476 000207 RU1$9:   RTS     PC      ;RETURN TO CALLER
```

```
1                               .TITLE  $DLG03
2                               .IFDF   CND$10
3                       ;
4                       ;       DLOG    V003A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  DLOG,DLOG10,$ERR;
9                               .IFNDF  FPU
10                              .GLOBL  $POLSH,$ADD,$SBD,$MLD,$DVD,$ID,$POPR4;
11                              .ENDC
12                      ;       THE FORTRAN DLOG AND DLOG10 FUNCTIONS
13                      ;       CALLING SEQUENCE:
14                      ;       JSR     R5,DLOG (OR DLOG10)
15                      ;       BR      A
16                      ;       .WORD   ARGUMENT ADDRESS
17                      ;A:
18                      ;       RETURNS LN(ARG) (OR LOG10(ARG)) IN R0 - R3.
19                      ;
20              000000          R0=%0
21              000001          R1=%1
22              000002          R2=%2
23              000003          R3=%3
24              000004          R4=%4
25              000005          R5=%5
26              000006          SP=%6
27              000007          PC=%7
28              000000          F0=%0
29              000001          F1=%1
30              000002          F2=%2
31              000003          F3=%3
32                              .IFNDF  FPU
33  06650 011746 DLOG10: MOV     @PC,-(SP)       ;GET 0004XX AS A FLAG
34  06652 000401         BR      LOG$10
35  06654 005046 DLOG:   CLR     -(SP)   ;FLAG DLOG
36  06656 010546 LOG$10: MOV     R5,-(SP)        ;SAVE RETURN POINTER
37  06660 016504         MOV     2(R5),R4        ;GET ARG ADDRESS
          000002
38  06664 062704         ADD     #8.,R4  ;POINT TO LEAST SIGNIFICANT PART
          000010
39  06670 012746         MOV     #147572,-(SP)
          147572
40  06674 012746         MOV     #173721,-(SP)
          173721
41  06700 012746         MOV     #071027,-(SP)   ;PUSH -1/2*LN(2)
          071027
42  06704 012746         MOV     #137061,-(SP)
          137061
43  06710 162706         SUB     #8.,SP  ;GET WORK SPACE
          000010
44  06714 014446         MOV     -(R4),-(SP)     ;GET ARG
45  06716 014446         MOV     -(R4),-(SP)
46  06720 014446         MOV     -(R4),-(SP)
47  06722 014446         MOV     -(R4),-(SP)
48  06724 003501         BLE     ERR$10  ;JUMP IF NOT POSITIVE
49  06726 006316         ASL     @SP
50  06730 116666         MOVB    1(SP),26.(SP)   ;GET EXPONENT
```

```
                000001
                000032
51  06736 112766        MOVB    #200,1(SP)      ;TRANSFORM ARG TO (1/2,1)
          000200
          000001
52  06744 006016        ROR     @SP
53  06746 012746        MOV     #157145,-(SP)
          157145
54  06752 012746        MOV     #031771,-(SP)
          031771
55  06756 012746        MOV     #002363,-(SP)   ;PUSH 1/2*ROOT2
          002363
56  06762 012746        MOV     #040065,-(SP)
          040065
57  06766 016646        MOV     14.(SP),-(SP)   ;PUSH X
          000016
58  06772 016646        MOV     14.(SP),-(SP)
          000016
59  06776 016646        MOV     14.(SP),-(SP)
          000016
60  07002 016646        MOV     14.(SP),-(SP)
          000016
61  07006 012746        MOV     #157145,-(SP)
          157145
62  07012 012746        MOV     #031771,-(SP)
          031771
63  07016 012746        MOV     #002363,-(SP)   ;PUSH 1/2*ROOT2
          002363
64  07022 012746        MOV     #040065,-(SP)
          040065
65  07026 004467        JSR     R4,$POLSH       ;ENTER POLISH MODE
          012612
66  07032 000700!       .WORD   $SBD,UP$10,$ADD,$DVD    ;GET (X-ROOT2)/
    07034 007210!
    07036 000704!
    07040 012210!
67                                                      ;(X+ROOT2)
68  07042 007246!       .WORD   DUP$10,DUP$10   ;GET THREE COPIES
    07044 007246!
69  07046 016146!       .WORD   $MLD    ;SET UP POLYNOMIAL
70  07050 017576!       .WORD   $POPR4          ;POP Y
71  07052 007144!       .WORD   REG$10
72  07054 016146!XPD$10:.WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD,$MLD,$ADD
    07056 000704!
    07060 016146!
    07062 000704!
    07064 016146!
    07066 000704!
    07070 016146!
    07072 000704!
73  07074 016146!       .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD
    07076 000704!
    07100 016146!
    07102 000704!
    07104 016146!
    07106 000704!
74                                  ;EXPAND POLYNOMIAL
```

```
75 07110 007232!          .WORD   SCL$10,$ID,PL2$10,$MLD  ;GET LN(EXP)
   07112 016046!
   07114 007270!
   07116 016146!
76 07120 000704!          .WORD   $ADD,EXI$10      ;COMBINE WITH FRACTION
   07122 007312!
77                                                        ;AND CHECK IF DONE
78 07124 016146!          .WORD   $MLD,EXI$10      ;MULTIPLY BY LOG10(E) AND RETURN
   07126 007312!
79 07130 062706 ERR$10: ADD       #24.,SP ;FLUSH JUNK
         000030
80 07134 004567           JSR     R5,$ERR
         012646
81 07140 000504           BR      ERO$10
82 07142     004          .BYTE   4
83 07143     003          .BYTE   3
84                ;
85 07144 012704 REG$10: MOV       #CON$10+8.,R4   ;POINT TO COEFFICIENTS
         007450!
86 07150 012705           MOV     #7,R5    ;SEVEN CONSTANTS
         000007
87 07154 000404           BR      STC$10
88 07156 010346 STK$10: MOV       R3,-(SP)
89 07160 010246           MOV     R2,-(SP)
90 07162 010146           MOV     R1,-(SP)         ;PUSH Y
91 07164 010046           MOV     R0,-(SP)
92 07166 014446 STC$10: MOV       -(R4),-(SP)      ;PUSH COEFFICIENT
93 07170 014446           MOV     -(R4),-(SP)
94 07172 014446           MOV     -(R4),-(SP)
95 07174 014446           MOV     -(R4),-(SP)
96 07176 005305           DEC     R5       ;COUNT CONSTANTS
97 07200 003366           BGT     STK$10
98 07202 012704           MOV     #XPD$10,R4       ;SET UP RETURN TO LIST
         007054!
99 07206 000134           JMP     @(R4)+
100               ;
101 7210 012666 UP$10:  MOV       (SP)+,22.(SP)   ;MOVE ITEM TO WORK SPACE
         000026
102 7214 012666           MOV     (SP)+,22.(SP)
         000026
103 7220 012666           MOV     (SP)+,22.(SP)
         000026
104 7224 012666           MOV     (SP)+,22.(SP)
         000026
105 7230 000134           JMP     @(R4)+
106               ;
107 7232 005046 SCL$10: CLR       -(SP)
108 7234 156616           BISB    12.(SP),@SP      ;GET EXPONENT
         000014
109 7240 162716           SUB     #200,@SP         ;REMOVE EXCESS 128
         000200
110 7244 000134           JMP     @(R4)+
111               ;
112 7246 016646 DUP$10: MOV       6(SP),-(SP)
         000006
113 7252 016646           MOV     6(SP),-(SP)      ;DUPLICATE STACK ITEM
         000006
```

```
114 7256 016646 000006          MOV     6(SP),-(SP)
115 7262 016646 000006          MOV     6(SP),-(SP)
116 7266 000134                 JMP     @(R4)+
117                     ;
118 7270 012746 147572  PL2$10: MOV     #147572,-(SP)
119 7274 012746 173721          MOV     #173721,-(SP)
120 7300 012746 071027          MOV     #071027,-(SP)   ;PUSH LN(2)
121 7304 012746 040061          MOV     #040061,-(SP)
122 7310 000134                 JMP     @(R4)+
123                     ;
124 7312 105366 000013  EXI$10: DECB    11.(SP) ;CHECK FOR ALOG10
125 7316 002411                 BLT     LGT$10  ;NO, DONE
126 7320 012746 024162          MOV     #024162,-(SP)
127 7324 012746 124467          MOV     #124467,-(SP)
128 7330 012746 055730          MOV     #055730,-(SP)   ;PUSH LOG10(E)
129 7334 012746 037736          MOV     #037736,-(SP)
130 7340 000134                 JMP     @(R4)+
131 7342 012600 LGT$10: MOV     (SP)+,R0        ;POP RESULT
132 7344 012601                 MOV     (SP)+,R1
133 7346 012602                 MOV     (SP)+,R2
134 7350 012603                 MOV     (SP)+,R3
135 7352 012605 ERO$10: MOV     (SP)+,R5        ;RESTORE RETURN
136 7354 005726                 TST     (SP)+   ;FLUSH FLAG
137 7356 000205                 RTS     R5
138                             .ENDC
139                     ;
140                             .IFDF   FPU
141                     DLOG10: MOV     @PC,R4;         GET 0004XX AS DLOG10 FLAG
142                             BR      LOG$10;
143                     DLOG:   CLR     R4;             GET 0 AS DLOG FLAG
144                     LOG$10: SETD    ;               DOUBLE PRECISION FP
145                             SETI    ;               SHORT INTEGERS
146                             MOV     #FCO*10,R0;     POINTER TO CONSTANTS
147                             LDD     @2(R5),F2;              GET ARG
148                             CFCC
149                             BLE     ERR$10;         JUMP IF NOT POSITIVE
150                             STEXP   F2,R1;          GET EXPONENT OF ARGUMENT
151                             LDCID   R1,F3;          CONVERT TO FP FORM
152                             MULD    (R0)+,F3;       SCALE FACTOR=EXPONENT*LN(2)
153                             LDEXP   #0,F2;          TRANSFORM ARG TO(1/2,1)
154                     ;
155                             LDD     F2,F1;
156                             SUBD    (R0),F2;        X=1/2*SQRT(2)
157                             ADDD    (R0)+,F1;       X+1/2*SQRT(2)
158                             DIVD    F1,F2;          W=(X-ROOT2)/(X+ROOT2)
159                             LDD     F2,F1;
```

```
160                             MULD    F1,F1;          Y= W**2
161                     ;
162                             MOV     #6,R1;          COUNT CONSTANTS FOR POLYNOMIAL
163                             LDD     (R0)+,F0;               INITIALIZE ACCUMULATOR
164                     XPD$10: MULD    F1,F0;
165                             DEC     R1;             COUNT
166                             ADDD    (R0)+,F0;       F0:= Y*F0 + C(I)
167                             BGT     XPD$10;         LOOP
168                             MULD    F2,F0;
169                             ADDD    (R0)+,F0;       F0:= W*F0 + 1/2*LN(2)
170                             ADDD    F3,F0;          ADD SCALE FACTOR FOR EXPONENT
171                             TST     R4;             TEST DLOG10 FLAG
172                             BEQ     LGT$10;
173                             MULD    (R0),F0;        DLOG10 = DLOG*LOG10(E)
174                     ;
175                     LGT$10: STD     F0,-(SP);       MOVE RESULT TO STACK
176                             MOV     (SP)+,R0;       AND TENCE TO R0,..R3
177                             MOV     (SP)+,R1;
178                             MOV     (SP)+,R2;
179                             MOV     (SP)+,R3;
180                             RTS     R5;             EXIT
181                     ;
182                     ERR$10: JSR     R5,$ERR;        ERROR 4,3
183                             RTS     R5;             EXIT - NO STACK CLEANUP REQUIRED
184                             .BYTE   4
185                             .BYTE   3
186                     ;
187                     ;       ORDER-DEPENDENT CONSTANTS FOR ROUTINE
188                     ;       R0 POINTS AT CURRENT CONSTANT IN FPU VERSION
189                     ;
190                     FCO$10: .WORD   040061,071027;          LN(2)
191                             .WORD   173721,147572;
192                     ;
193                             .WORD   040065,002363;  1/2*SQRT(2)
194                             .WORD   031771,157145;
195                             .ENDC
196                     ;
197 7360 037455                 .WORD   037455,106270   ;.16948212488
    7362 106270
198 7364 157166                 .WORD   157166,174770
    7366 174770
199                     ;
200 7370 037471                 .WORD   037471,072731   ;.1811136267967
    7372 072731
201 7374 137716                 .WORD   137716,117115
    7376 117115
202                     ;
203 7400 037543                 .WORD   037543,111153   ;.2222382332791
    7402 111153
204 7404 060101                 .WORD   060101,135465
    7406 135465
205                     ;
206 7410 037622                 .WORD   037622,044436   ;.2857140915904889
    7412 044436
207 7414 007306                 .WORD   007306,063062
    7416 063062
208                     ;
```

```
209 7420 037714          .WORD   037714,146314   ;.4000000001206045365
    7422 146314
210 7424 153450          .WORD   153450,165773
    7426 165773
211                      ;
212 7430 040052          .WORD   040052,125252   ;.6666666666633660894
    7432 125252
213 7434 125247          .WORD   125247,004643
    7436 004643
214                      ;
215 7440 040400 CON$10:  .WORD   040400,000000   ;2.0000000000000261
    7442 000000
216 7444 000000          .WORD   000000,000057
    7446 000057
217                      .IFDF   FPU
218                      ;       MORE ORDER-DEPENDENT CONSTANTS
219                      ;
220                      .WORD   137661,071027;  -1/2*LN(2)
221                      .WORD   173721,147572;
222                      ;
223                      .WORD   037736,055730;  LOG10(E)
224                      .WORD   124467,024162;
225                      .ENDC
226                      .ENDC
```

```
1                                       .TITLE  $DNT02
2                                       .IFDF   CNO$11
3                               ;
4                               ;       $DINT   V002A
5                               ;
6                                       .GLOBL  $DINT
7                               ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
8                               ;       OTS INTERNAL FUNCTION TO FIND THE INTEGER
9                               ;       PART OF A DOUBLE PRECISION NUMBER.
10                              ;       CALLED IN THE POLISH MODE.
11              000000                  R0=%0
12              000001                  R1=%1
13              000002                  R2=%2
14              000003                  R3=%3
15              000004                  R4=%4
16              000005                  R5=%5
17              000006                  SP=%6
18              177304                  MQ=177304
19              177316                  ASH=177316
20              000000                  F0=%0
21              000001                  F1=%1
22                                      .IFDF   FPU
23                              $DINT:  .WORD   170011  ;;SETD
24                                      .WORD   172426  ;;LDD    (SP)+,F0         ;LOAD ARG
25                                      .WORD   171467,4        ;;MODD  ONE,F0   ;GET INTEGER PAR
26                                      .WORD   174146  ;;STD   F1,-(SP)         ;PUSH INTEGER
27                                      JMP     @(R4)+  ;RETURN TO CALLER
28                                      .WORD   040200,0,0,0    ;FLOATING 1.
29                                      .ENDC
30                                      .IFNDF  FPU
31  07450  012600  $DINT:   MOV     (SP)+,R0        ;POP DOUBLE ARG
32  07452  012601           MOV     (SP)+,R1
33  07454  012602           MOV     (SP)+,R2
34  07456  012603           MOV     (SP)+,R3
35  07460  010446           MOV     R4,-(SP)
36  07462  010546           MOV     R5,-(SP)
37  07464  010004           MOV     R0,R4   ;GET EXPONENT
38  07466  006104           ROL     R4
39  07470  105004           CLRB    R4
40  07472  000304           SWAB    R4
41  07474  162704           SUB     #270,R4 ;CONVERT TO -SHIFT COUNT
               000270
42  07500  002041           BGE     DNE$11  ;JUMP IF ARG MUST BE INTEGER ALREADY
43  07502  022704           CMP     #-70,R4
               177710
44  07506  002405           BLT     SHF$11  ;JUMP TO GET INTEGER PART
45  07510  005000           CLR     R0      ;ANSWER IS 0
46  07512  005001           CLR     R1
47  07514  005002  C2$$11:  CLR     R2
48  07516  005003           CLR     R3
49  07520  000431           BR      DNE$11
50  07522                  SHF$11:
51                                      .IFNDF  EAE&MULDIV
52  07522  010405           MOV     R4,R5   ;SAVE A COPY OF SHIFT COUNT
53  07524  022704           CMP     #-32.,R4        ;CHECK LOW OR HIGH TRUNCATION
               177740
54  07530  002415           BLT     RUR$11
```

```
                        BEQ     C23$11  ;GO CLEAR LOW ORDER HALF
                        ADD     #32.,R4 ;DO HIGH ORDER
                        MOV     R4,R5
RR1$11: ROR     R0      ;SHIFT OUT FRACTION BITS
                        ROR     R1
                        INC     R4
                        BLT     RR1$11
AS1$11: ASL     R1      ;SHIFT IN 0'S
                        ROL     R0
                        INC     R5
                        BLT     AS1$11
                        BR      C23$11  ;GO CLEAR LOW ORDER
ROR$11: ROR     R2      ;MOVE OUT FRACTION BITS
                        ROR     R3
                        INC     R4      ;COUNT LOOP
                        BLT     ROR$11
ASL$11: ASL     R3
                        ROL     R2
                        INC     R5      ;COUNT LOOP
                        BLT     ASL$11
                        .ENDC
                        .IFDF   EAE
                        MOV     #MQ,R5  ;POINT TO MQ
                        .ENDC
                        .IFDF   MULDIV!EAE
                        CMP     #-32.,R4        ;CHECK FOR HIGH OR LOW ORDER TRU
                        BLT     R23$11  ;LOW
                        BEQ     C23$11  ;CLEAR LOW ORDER
R01$11: ADD     #32.,R4 ;HIGH ORDER PARTS
                        .IFDF   MULDIV
                        .WORD   073004  ;;ASHC  R4,R0   ;SHIFT OUT FRACTION
                        NEG     R4      ;SET TO SHIFT LEFT
                        .WORD   073004  ;;ASHC  R4,R0   ;BRING IN THE 0'S
                        .ENDC
                        .IFDF   EAE
                        MOV     R1,@R5  ;HIGH ORDER TO AC,MQ
                        MOV     R0,-(R5)
                        MOV     R4,@#ASH        ;SHIFT RIGHT
                        NEG     R4
                        MOV     R4,@#ASH        ;SHIFT LEFT
                        MOV     (R5)+,R0        ;RESULT TO REGS
                        MOV     @R5,R1
                        .ENDC
                        BR      C23$11  ;GO CLEAR LOW ORDER
                        .IFDF   MULDIV
R23$11: .WORD   073204  ;;ASHC  R4,R2
                        NEG     R4
                        .WORD   073204  ;;ASHC  R4,R2   ;SHIFT IN 0'S
                        .ENDC
                        .IFDF   EAE
R23$11: MOV     R3,@R5  ;LOW ORDER TO AC,MQ
                        MOV     R2,-(R5)
                        MOV     R4,@#ASH        ;DUMP BITS
                        NEG     R4
                        MOV     R4,@#ASH        ;BRING IN 0'S
                        MOV     (R5)+,R2        ;RESULT TO REGS
```

```
111                             MOV     @R5,R3
112                             .ENDC
113                             .ENDC
114 7604 012605 DNE$11: MOV     (SP)+,R5
115 7606 012604         MOV     (SP)+,R4
116 7610 010346         MOV     R3,-(SP)        ;PUSH RESULT
117 7612 010246         MOV     R2,-(SP)
118 7614 010146         MOV     R1,-(SP)
119 7616 010046         MOV     R0,-(SP)
120 7620 000134         JMP     @(R4)+  ;RETURN
121                             .ENDC
122                             .ENDC
```

```
1                               .TITLE  $DR02
2                               .IFDF   CND$12
3                               .GLOBL  $DR,$ERR
4                       ;       $DR     THE DOUBLE PRECISION TO REAL CONVERTER
5                       ;
6                       ;       $DR     V002A
7                       ;
8                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
9                       ;       ROUND THE TOP STACK ITEM TO REAL FORMAT.
10              000004          R4=%4
11              000005          R5=%5
12              000006          SP=%6
13              000000          F0=%0
14                              .IFDF   FPU
15                      $DR:    .WORD   170001  ;;SETF
16                              .WORD   177426  ;;LDCDF (SP)+,F0        ;CONVERT ARG
17                              .WORD   170000  ;;CFCC  ;GET CONDITION CODES
18                              BVS     OV1$12  ;JUMP IF OVERFLOW ON ROUND
19                              .WORD   174046  ;;STF   F0,-(SP)
20                              JMP     @(R4)+  ;           !!!
21                              .ENDC
22                              .IFNDF  FPU
23  07622  006160  000004 $DR:  ROL     4(SP)   ;ROUND LOW ORDER PART
24  07626  005560  000002       ADC     2(SP)
25  07632  005510               ADC     @SP
26  07634  103406               BCS     OVR$12  ;JUMP IF OVERFLOW
27  07636  102405               BVS     OVR$12
28  07640  012660  000002 DR1$12: MOV     (SP)+,2(SP)     ;MOVE HIGHEST ORDER PART
29  07644  012660  000002       MOV     (SP)+,2(SP)     ;MOVE LOW ORDER REAL
30  07650  000134               JMP     @(R4)+  ;RETURN
31  07652  022626  OVR$12: CMP     (SP)+,(SP)+     ;FLUSH ARG
32  07654  022626               CMP     (SP)+,(SP)+
33                              .ENDC
34  07656  004567  012124 OV1$12: JSR     R5,$ERR ;ERROR 3,23
35  07662  000401               BR      DR2$12
36  07664      003              .BYTE   3
37  07665      027              .BYTE   23.
38  07666  005046  DR2$12: CLR     -(SP)   ;RETURN 0.
39  07670  005046               CLR     -(SP)
40  07672  000134               JMP     @(R4)+
41                              .ENDC
```

```
1                               .TITLE  $DSN04
2                               .IFDF   CND$13
3                       ;
4                       ;       DSINCS  V004A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8
9                               .GLOBL  DSIN,DCOS;
10                              .IFNDF  FPU
11                              .GLOBL  $ADD,$SBD,$MLD,$DVD,$DINT,$POLSH,$POPR4;
12                              .ENDC
13                      ;       DSIN    DCOS    THE DOUBLE PRECISION SIN AND COS
14                      ;       FUNCTIONS.
15                      ;       CALLING SEQUENCE:
16                      ;       JSR     R5,DSIN (OR DCOS)
17                      ;       BR      A
18                      ;       .WORD   ARG ADDRESS
19                      ;A:
20                      ;       RETURNS SIN OR COS OF ARG IN R0 - R3.
21              000000          R0=%0
22              000001          R1=%1
23              000002          R2=%2
24              000003          R3=%3
25              000004          R4=%4
26              000005          R5=%5
27              000006          SP=%6
28              000007          PC=%7
29              000000          F0=%0
30              000001          F1=%1
31              000002          F2=%2
32              000003          F3=%3
33                              .IFNDF  FPU
34  07674 010546 DCOS:          MOV     R5,-(SP)        ;SAVE RETURN POINTER
35  07676 016504                MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
            000002
36  07702 005046                CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
37  07704 016446                MOV     6(R4),-(SP)
            000006
38  07710 016446                MOV     4(R4),-(SP)
            000004
39  07714 016446                MOV     2(R4),-(SP)     ;PUSH ARGUMENT
            000002
40  07720 011446                MOV     @R4,-(SP)
41  07722 012746                MOV     #064302,-(SP)
            064302
42  07726 012746                MOV     #121041,-(SP)
            121041
43  07732 012746                MOV     #007732,-(SP)   ;PUSH PI/2
            007732
44  07736 012746                MOV     #040311,-(SP)
            040311
45  07742 004467                JSR     R4,$POLSH       ;ENTER POLISH MODE
            011675
46  07746 000704!               .WORD   $ADD,SNC$13     ;COS(X)=SIN(X+PI/2)
    07750 010000!
47  07752 010546 DSIN:          MOV     R5,-(SP)        ;SAVE RETURN
```

```
48 07754 016504          MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
         000002
49 07760 005046          CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
50 07762 016446          MOV     6(R4),-(SP)
         000006
51 07766 016446          MOV     4(R4),-(SP)
         000004
52 07772 016446          MOV     2(R4),-(SP)
         000002
53 07776 011446          MOV     @R4,-(SP)       ;PUSH ARGUMENT
54 10000 006316 SNG$13:  ASL     @SP     ;CLEAR SIGN AND SAVE IT
55 10002 006066          ROR     8.(SP)  ;IN QUADRANT FLAG
         000010
56 10006 006016          ROR     @SP
57 10010 012746          MOV     #064302,-(SP)
         064302
58 10014 012746          MOV     #121041,-(SP)
         121041
59 10020 012746          MOV     #007732,-(SP)   ;PUSH 2*PI
         007732
60 10024 012746          MOV     #040711,-(SP)
         040711
61 10030 004467          JSR     R4,$POLSH       ;ENTER POLISH MODE
         011610
62 10034 012210!         .WORD   $DVD    ;X/2PI
63 10036 010154!         .WORD   DUP$13  ;2 COPIES
64 10040 007450!         .WORD   $DINT   ;INT(X/2PI)
65 10042 000700!         .WORD   $SBD    ;FRACT(X/2PI)
66 10044 010176!         .WORD   X4$13   ;4*FRACT(X/2PI)
67 10046 010154!         .WORD   DUP$13  ;2 COPIES
68 10050 007450!         .WORD   $DINT   ;INT(4*FRACT(X/2PI))
69 10052 010216!         .WORD   QUD$13  ;SAVE INT(.....)
70 10054 000700!         .WORD   $SBD    ;Y=FRACT(4*FRACT(X/2PI))
71 10056 010224!         .WORD   QST$13  ;REDUCE Y TO (-1,1)
72 10060 010154!QSE$13:  .WORD   DUP$13  ;2 COPIES
73 10062 010154!         .WORD   DUP$13  ;3 COPIES
74 10064 016146!         .WORD   $MLD    ;Y*Y
75 10066 017576!         .WORD   $POPR4  ;SAVE Y*Y
76 10070 010300!         .WORD   PLY$13  ;PUSH COEFFICIENTS
77 10072 016146!XPD$13:  .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD,$MLD,$ADD
   10074 000704!
   10076 016146!
   10100 000704!
   10102 016146!
   10104 000704!
   10106 016146!
   10110 000704!
78 10112 016146!         .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD,$MLD,$ADD
   10114 000704!
   10116 016146!
   10120 000704!
   10122 016146!
   10124 000704!
   10126 016146!
   10130 000704!
79 10132 016146!         .WORD   $MLD    ;Y*P(Y*Y)
80 10134 017576!PR4$13:  .WORD   $POPR4  ;POP HIGH ORDER RESULT
```

```
81 10136 010140'          .WORD   RTN$13
82 10140 005726 RTN$13:   TST     (SP)+    ;POP QUADRANT FLAG
83 10142 002002           BGE     RT1$13   ;JUMP IF ARGUMENT WAS +
84 10144 062700           ADD     #100000,R0       ;SIN(-X)=-SIN(X)
         100000
85 10150 012605 RT1$13:   MOV     (SP)+,R5
86 10152 000205           RTS     R5       ;BACK TO CALLER
87                        ;
88 10154 016646 DUP$13:   MOV     6(SP),-(SP)      ;DUPLICATE STACK ITEM
         000006
89 10160 016646           MOV     6(SP),-(SP)
         000006
90 10164 016646           MOV     6(SP),-(SP)
         000006
91 10170 016646           MOV     6(SP),-(SP)
         000006
92 10174 000134           JMP     @(R4)+
93                        ;
94 10176 005715 X4$13:    TST     @SP      ;CHECK FOR 0 FRACTION
95 10200 001403           BEQ     ZER$13;  QUIT NOW
96 10202 105266           INCB    1(SP)    ;QUADRUPLE STACK ITEM
         000001
97 10206 000134           JMP     @(R4)+
98 10210 012704 ZER$13:   MOV     #PR4$13,R4;      RETURN ZERO RESULT
         010134'
99 10214 000134           JMP     @(R4)+;          USE POLISH
100                       ;
101 0216 051666 QUD$13:   BIS     @SP,16.(SP)      ;SAVE QUADRANT NUMBER
         000020
102 0222 000134           JMP     @(R4)+
103                       ;
104 0224 105766 QST$13:   TSTB    8.(SP)   ;TEST QUADRANT
         000010
105 0230 001415           BEQ     Q13$13   ;JUMP IF FIRST OR THIRD QUAD
106 0232 062716           ADD     #100000,@SP      ;NEGATE STACK ITEM
         100000
107 0236 005046           CLR     -(SP)
108 0240 005046           CLR     -(SP)
109 0242 005046           CLR     -(SP)    ;PUSH A FLOATING 1.
110 0244 012746           MOV     #40200,-(SP)
         040200
111 0250 004467           JSR     R4,$POLSH        ;ENTER POLISH
         011370
112 0254 000704'          .WORD   $ADD,QSR$13      ;X=1.-X
    0256 010260'
113 0260 012704 QSR$13:   MOV     #QSE$13,R4       ;POINT BACK INTO LIST
         010060'
114 0264 106266 Q13$13:   ASRB    9.(SP)   ;TEST QUADRANT
         000011
115                       ;
116 0270 103002           BCC     QUT$13   ;JUMP IF FIRST OR SECOND
117 0272 062716           ADD     #100000,@SP      ;NEGATE STACK ITEM
         100000
118 0276 000134 QUT$13:   JMP     @(R4)+
119                       ;
120 0300 012704 PLY$13:   MOV     #CON$13+8.,R4    ;POINT TO LIST OF COEFFICIENTS
         010454'
```

```
121 0304 012705          MOV     #9.,R5  ;NINE CONSTANTS
         000011
122 0310 000404          BR      PY1$13
123 0312 010346 PY2$13:  MOV     R3,-(SP)
124 0314 010246          MOV     R2,-(SP)
125 0316 010146          MOV     R1,-(SP)        ;PUSH Y*Y
126 0320 010046          MOV     R0,-(SP)
127 0322 014446 PY1$13:  MOV     -(R4),-(SP)     ;PUSH CONSTANT
128 0324 014446          MOV     -(R4),-(SP)
129 0326 014446          MOV     -(R4),-(SP)
130 0330 014446          MOV     -(R4),-(SP)
131 0332 005305          DEC     R5      ;COUNT COEFFICIENTS
132 0334 003366          BGT     PY2$13
133 0336 012704          MOV     #XPD$13,R4
         010072'
134 0342 000134          JMP     @(R4)+
135                      .ENDC
136              ;
137                      .IFDF   FPU
138              DCOS:   SETD     ;              DOUBLE PRECISION FP
139                      LDD     @2(R5),F0;      GET ARGUMENT
140                      ADDD    PI2$13,F0;      COS(X)=SIN(X+PI/2)
141                      BR      SNC$13;
142              DSIN:   SETD     ;              DOUBLE PRECISION FP
143                      LDD     @2(R5),F0;      GET ARGUMENT
144              SNC$13: SETI     ;              SHORT INTEGERS
145                      MOV     #FCO$13,R0;     POINTER TO CONSTANTS
146                      CLR     R4;             SIGN FLAG: + ARG
147                      CFCC     ;              GET SIGN OF ARG
148                      BGE     POS$13;
149                      INC     R4;             SIGN FLAG: - ARG
150                      ABSD    F0;             REMOVE ARGUMENT SIGN
151              POS$13: DIVD    (R0)+,F0;       X/2PI
152                      MODD    #1.0,F0;        F0= FRACT(X/2PI)
153                      CFCC
154                      BEQ     RTN$13;         EXIT ON 0 FRACTION
155                      MODD    #4.0,F0;                F0= FRACT(4*FRACT(X/2PI)
156                      STCDI   F1,R1;          QUAD= INT(4*FRACT(X/2PI))
157                      ROR     R1;
158                      BCC     Q13$13;         JUMP IF FIRST OR THIR QUAD
159                      NEGD    F0;
160                      ADDD    #1.0,F0;        Y=1.0-X
161              Q13$13: ROR     R1;
162                      BCC     Q12$13;         JUMP IF FIRST OR 2ND QUAD
163                      NEGD    F0;             Y = -Y
164              ;
165              Q12$13: LDD     F0,F2;
166                      MULD    F2,F2;          Z=Y**2
167                      MOV     #8.,R1;         COUNT OF CONSTANTS FOR POLYNOMIA
168                      LDD     (R0)+,F1;       INITIALIZE ACCUMULATOR
169              XPD$13: MULD    F2,F1;
170                      DEC     R1;             COUNT
171                      ADDD    (R0)+,F1;       F1:= Z:F1 + C(I)
172                      BGT     XPD$13;         LOOP
173              ;
174                      MULD    F1,F0;          F0:= Y*F1
175                      TST     R4;             TEST SIGN FLAG
```

```
176                             BEQ     RTN$13;
177                             NEGD    F0;             SIN(-X) = -SIN(X)
178                     RTN$13: STD     F0,-(SP);       MOVE RESULT TO STACK
179                             MOV     (SP)+,R0;               AND THENCE TO R0...R3
180                             MOV     (SP)+,R1;
181                             MOV     (SP)+,R2;
182                             MOV     (SP)+,R3;
183                             RTS     R5;             EXIT
184                     ;
185                     PI2$13: .WORD   040311,007732;  PI/2
186                             .WORD   121041,064302;
187                     ;
188                     ;       ORDER-DEPENDENT CONSTANTS
189                     ;
190                     FCU$13: .WORD   040711,007732;  2*PI
191                             .WORD   121041,064302;
192                             .ENDC
193 0344 026716                .WORD   026716,106703   ;.587061098171E-11
    0346 106703
194 0350 045277                 .WORD   045277,146362
    0352 146362
195                     ;
196 0354 130467                 .WORD   130467,136273   ;-.66843217206396E-9
    0356 136273
197 0360 103054                 .WORD   103054,123153
    0362 123153
198                     ;
199 0364 032164                 .WORD   032164,074657   ;.5692134872719023E-7
    0366 074657
200 0370 047254                 .WORD   047254,154742
    0372 154742
201                     ;
202 0374 133561                 .WORD   133561,101646   ;-.3598843007208693E-5
    0376 101646
203 0400 167216                 .WORD   167216,134016
    0402 134016
204                     ;
205 0404 035050                 .WORD   035050,036032   ;.1604411847068221E-3
    0406 036032
206 0410 041214                 .WORD   041214,103131
    0412 103131
207                     ;
208 0414 136231                 .WORD   136231,064546   ;-.4681754135302643E-2
    0416 064546
209 0420 071423                 .WORD   071423,125024
    0422 125024
210                     ;
211 0424 037243                 .WORD   037243,032743   ;.7969262624616544E-1
    0426 032743
212 0430 035655                 .WORD   035655,051557
    0432 051557
213                     ;
214 0434 140045                 .WORD   140045,056747   ;-.6459640975062462
    0436 056747
215 0440 030455                 .WORD   030455,171222
    0442 171222
216                     ;
```

```
217 0444 040311 CON$131: .WORD   040311,007732   ;1.570796326794897
    0446 007732
218 0450 121041          .WORD   121041,064302
    0452 064302
219                 ;
220                      .ENDC
```

```
1                               .TITLE  $DSQ03
2                               .IFOF   CND$14
3                       ;
4                       ;       DSQRT   V003A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8
9                               .GLOBL  DSQRT,$ERR;
10                              .IFNDF  FPU
11                              .GLOBL  $ADD,$DVD,$POLSH;
12                              .ENDC
13                      ;       $DSQRT  THE DOUBLE PRECISION SQUARE ROOT FUNCTION
14                      ;       CALLING SEQUENCE:
15                      ;       JSR     R5,$DSQRT
16                      ;       BR      A
17                      ;       #ARG
18                      ;A:
19                      ;       RETURNS DSQRT IN R0 - R3.
20                      ;
21              000000          R0=%0
22              000001          R1=%1
23              000002          R2=%2
24              000003          R3=%3
25              000004          R4=%4
26              000005          R5=%5
27              000000          F0=%0
28              000001          F1=%1
29              000002          F2=%2
30              000006          SP=%6
31                              .IFNDF  FPU
32  10454 010546 DSQRT:         MOV     R5,-(SP)
33  10456 016505                MOV     2(R5),R5        ;GET ARGUMENT ADDRESS
            000002
34  10462 011501                MOV     @R5,R1  ;GET HIGH ORDER ARGUMENT
35  10464 100467                BMI     ERR$14  ;ERROR IF ARGUMENT NEGATIVE
36  10466 001472                BEQ     ZER$14  ;FAST EXIT IF ZERO
37  10470 016502                MOV     2(R5),R2
            000002
38  10474 012746                MOV     #4,-(SP)        ;PUSH ITERATION COUNT
            000004
39  10500 006201                ASR     R1      ;FORM INITIAL ESTIMATE
40  10502 006002                ROR     R2
41  10504 062701                ADD     #20100,R1
            020100
42  10510 005046                CLR     -(SP)
43  10512 005046                CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
44  10514 010246                MOV     R2,-(SP)
45  10516 010146                MOV     R1,-(SP)        ;'CAUSE ADD AND DIVIDE ARE
46  10520 005046                CLR     -(SP)   ;FASTER THAT WAY
47  10522 005046                CLR     -(SP)
48  10524 016546                MOV     2(R5),-(SP)
            000002
49  10530 011546                MOV     @R5,-(SP)
50  10532 005046                CLR     -(SP)
51  10534 005046                CLR     -(SP)
52  10536 010246                MOV     R2,-(SP)
```

```
53 10540 010146          MOV     R1,-(SP)
54 10542 004467 LUP$14: JSR     R4,$POLSH       ;ENTER POLISH MODE
         011076
55 10546 012210!         .WORD   $DVD,$ADD,UPL$14        ;(X/E+E)
   10550 000704!
   10552 010554!
56 10554 162716 UPL$14: SUB     #200,@SP        ;(X/E+E)/2
         000200
57 10560 005366          DEC     8.(SP)  ;COUNT LOOP
         000010
58 10564 001420          BEQ     OUT$14
59 10566 016546          MOV     6(R5),-(SP)
         000006
60 10572 016546          MOV     4(R5),-(SP)
         000004
61 10576 016546          MOV     2(R5),-(SP)     ;USE LOW ORDER PARTS
         000002
62 10602 011546          MOV     @R5,-(SP)       ;TOO FROM NOW ON
63 10604 016646          MOV     14.(SP),-(SP)
         000016
64 10610 016646          MOV     14.(SP),-(SP)
         000016
65 10614 016646          MOV     14.(SP),-(SP)
         000016
66 10620 016646          MOV     14.(SP),-(SP)
         000016
67 10624 000746          BR      LUP$14  ;GO FOR ANOTHER ITERATION
68 10626 012600 OUT$14: MOV     (SP)+,R0        ;GET RESULT INTO R0-R3
69 10630 012601          MOV     (SP)+,R1
70 10632 012602          MOV     (SP)+,R2
71 10634 012603          MOV     (SP)+,R3
72 10636 005726          TST     (SP)+   ;POP ITERATION COUNTER
73 10640 012605 RTN$14: MOV     (SP)+,R5
74 10642 000205          RTS     R5      ;RETURN TO CALLER
75 10644 004567 ERR$14: JSR     R5,$ERR ;ERROR 4,4
         011136
76 10650 000773          BR      RTN$14
77 10652     004         .BYTE   4
78 10653     004         .BYTE   4
79 10654 005000 ZER$14: CLR     R0
80 10656 005001          CLR     R1
81 10660 005002          CLR     R2
82 10662 005003          CLR     R3
83 10664 000765          BR      RTN$14
84                 ;
85                       .ENDC
86                 ;
87                       .IFDF   FPU
88                 DSQRT: MOV     2(R5),R4;       GET ARGUMENT ADDRESS
89                       MOV     @R4,R1;         GET HIGH ORDER ARGUMENT
90                       BMI     ERR$14  ;ERROR IF ARGUMENT NEGATIVE
91                       BEQ     ZER$14  ;FAST EXIT IF ZERO
92                       MOV     2(R4),R2;
93                       ASR     R1      ;FORM INITIAL ESTIMATE
94                       ROR     R2
95                       ADD     #20100,R1
96                       CLR     -(SP)
```

```
97                      CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
98                      MOV     R2,-(SP)
99                      MOV     R1,-(SP)        ;'CAUSE ADD AND DIVIDE ARE
100                     MOV     #4,R0;          ITERATION COUNT
101                     SETD    ;               DOUBLE PRECISION FP
102                     LDD     (SP)+,F0;       GET INITIAL ESTIMATE
103                     LDD     @R4,F2;         GET X
104             ;
105             LUP$14: LDD     F0,F1;          E=E'
106                     LDD     F2,F0;          X
107                     DIVD    F1,F0;          X/E
108                     ADDD    F1,F0;          X/E+E
109                     DEC     R0;             COUNT
110                     DIVD    #2.0,F0;        E'=(X/E+E)/2
111                     BGT     LUP$14;         LOOP
112             ;
113                     STD     F0,-(SP);       MOVE RESULT TO STACK
114                     MOV     (SP)+,R0;
115                     MOV     (SP)+,R1;
116                     MOV     (SP)+,R2;       AND THENCE TO R0...R3
117                     MOV     (SP)+,R3;
118                     RTS     R5;
119             ;
120             ERR$14: JSR     R5,$ERR;        ERROR 4,4
121                     RTS     R5;
122                     .BYTE   4
123                     .BYTE   4
124             ZER$14: CLR     R0;
125                     CLR     R1;
126                     CLR     R2;
127                     CLR     R3;
128                     RTS     R5;
129                     .ENDC
130                     .ENDC
```

```
                        .TITLE  $DTN03
                        .IFDF   CND$15
                ;
                ;       DATAN   V003A
                ;
                ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                ;
                        .GLOBL  DATAN,DATAN2;
                        .IFNDF  FPU
                        .GLOBL  $ADD,$SBD,$MLD,$DVD,$POLSH,$POPR4;
                        .ENDC
                ;       THE FORTRAN DATAN AND DATAN2 FUNCTIONS
                ;       CALLING SEQUENCE FOR DATAN:
                ;       JSR     R5,DATAN
                ;       BR      A
                ;       .WORD   ARGUMENT ADDRESS
                ;A:
                ;       RETURNS ARCTAN(ARG) IN R0 AND R1.
                ;
                ;       CALLING SEQUENCE FOR DATAN2:
                ;       JSR     R5,DATAN2
                ;       BR      A
                ;       .WORD   ARGUMENT 1 ADDRESS
                ;       .WORD   ARGUMENT 2 ADDRESS
                ;A:
                ;       RETURNS ACRTAN(ARG1/ARG2) IN R0 AND R1.
                ;       IF ABS(ARG1/ARG2) > 2**24, THE RESULT IS
                ;       SIGN(ARG1)*PI/2.
                ;       IF ARG2 <0 THE RESULT IS ARCTAN(ARG1/ARG2) +
                ;       SIGN(ARG1)*PI.
                ;
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000000          F0=%0
        000001          F1=%1
        000002          F2=%2
        000003          F3=%3
        000004          F4=%4
        000005          F5=%5
                        .IFNDF  FPU
10666 010546 DATAN2:    MOV     R5,-(SP)
10670 005046            CLR     -(SP)   ;CLEAR SIGN FLAG
10672 005046            CLR     -(SP)   ;CLEAR DATAN2 BIAS
10674 005046            CLR     -(SP)
10676 005046            CLR     -(SP)
10700 005046            CLR     -(SP)
10702 005046            CLR     -(SP)   ;CLEAR QUADRANT BIAS
10704 005046            CLR     -(SP)
10706 005046            CLR     -(SP)
10710 005046            CLR     -(SP)
10712 016504            MOV     2(R5),R4        ;GET FIRST ARG ADDRESS
```

```
            000002
58 10716 016446          MOV     6(R4),-(SP)
            000006
59 10722 016446          MOV     4(R4),-(SP)
            000004
60 10726 016446          MOV     2(R4),-(SP)     ;GET FIRST ARG
            000002
61 10732 011446          MOV     @R4,-(SP)
62 10734 011600          MOV     @SP,R0  ;ARG1 TO R0
63 10736 016504          MOV     4(R5),R4        ;GET SECOND ARG ADDRESS
            000004
64 10742 016446          MOV     6(R4),-(SP)
            000006
65 10746 016446          MOV     4(R4),-(SP)
            000004
66 10752 016446          MOV     2(R4),-(SP)     ;GET SECOND ARG
            000002
67 10756 011446          MOV     @R4,-(SP)
68 10760 011601          MOV     @SP,R1  ;ARG2 TO R1
69 10762 001445          BEQ     INF$15  ;JUMP IF DENOMINATOR IS 0
70 10764 006300          ASL     R0      ;GET ABS VAL ARG1
71 10766 105000          CLRB    R0      ;GET EXPONENT
72 10770 000300          SWAB    R0
73 10772 006301          ASL     R1
74 10774 105001          CLRB    R1      ;GET EXPONENT ARG2
75 10776 000301          SWAB    R1
76 11000 160100          SUB     R1,R0   ;GET EXPONENT DIFFERENCE
77 11002 022700          CMP     #58.,R0 ;CHECK MAGNITUDE
            000072
78 11006 002433          BLT     INF$15  ;TREAT AS INFINITY
79 11010 004467 DIV$15:  JSR     R4,$POLSH
            010630
80 11014 012210'         .WORD   $DVD,UPL$15     ;GET ARG1/ARG2
   11016 011020'
81 11020 005775 UPL$15:  TST     @4(R5)  ;IF ARG2 >0, BIAS =0
            000004
82 11024 002022          BGE     ATE$15  ;IF ARG2<0, BIAS=SIGN(ARG1)*PI
83 11026 012766          MOV     #040511,16.(SP) ;PI
            040511
            000020
84 11034 012766          MOV     #007732,18.(SP)
            007732
            000022
85 11042 012766          MOV     #121041,20.(SP)
            121041
            000024
86 11050 012766          MOV     #064301,22.(SP)
            064301
            000026
87 11056 005775          TST     @2(R5)  ;TEST ARG1
            000002
88 11062 002003          BGE     ATE$15
89 11064 062766          ADD     #100000,16.(SP) ;-PI
            100000
            000020
90 11072 005716 ATE$15:  TST     @SP     ;SET CODES
91 11074 000443          BR      AT1$15  ;JOIN MAIN ROUTINE
```

```
92 11076 062706 INF$15: ADD     #36.,SP ;FLUSH STACK
         000044
93 11102 012700         MOV     #040311,R0      ;ANS = SIGN(ARG1)*PI/2
         040311
94 11106 012701         MOV     #007732,R1
         007732
95 11112 012702         MOV     #121041,R2
         121041
96 11116 012703         MOV     #064301,R3
         064301
97 11122 005775         TST     @2(R5)  ;TEST ARG1
         000002
98 11126 002002         BGE     INR$15  ;JUMP IF +PI/2
99 11130 062700         ADD     #100000,R0      ;-PI/2
         100000
100 1134 000205 INR$15: RTS     R5      ;RETURN TO USER
101              ;
102 1136 010546 DATAN:  MOV     R5,-(SP)
103 1140 005046         CLR     -(SP)   ;CLEAR SIGN FLAG
104 1142 005046         CLR     -(SP)   ;CLEAR ATAN2 BIAS
105 1144 005046         CLR     -(SP)
106 1146 005046         CLR     -(SP)
107 1150 005046         CLR     -(SP)
108 1152 005046         CLR     -(SP)   ;CLEAR QUADRANT BIAS
109 1154 005046         CLR     -(SP)
110 1156 005046         CLR     -(SP)
111 1160 005046         CLR     -(SP)
112 1162 016504         MOV     2(R5),R4        ;GET ARG ADDRESS
         000002
113 1166 016446         MOV     6(R4),-(SP)
         000006
114 1172 016446         MOV     4(R4),-(SP)
         000004
115 1176 016446         MOV     2(R4),-(SP)     ;GET LOW ORDER ARG
         000002
116 1202 011446         MOV     @R4,-(SP)       ;GET HIGH ORDER
117 1204 002004 AT1$15: BGE     PLU$15  ;JUMP IF QUADRANT 1 OR 3
118 1206 062716         ADD     #100000,@SP     ;GET ABS VALUE
         100000
119 1212 005266         INC     24.(SP) ;FLAG =
         000030
120 1216 021627 PLU$15: CMP     @SP,#40200      ;CHECK IF <1.
         040200
121 1222 103455         BLO     LE1$15  ;JUMP IF <1.
122 1224 003011         BGT     GT1$15  ;>1.
123 1226 005766         TST     2(SP)   ;CHECK LOW ORDER
         000002
124 1232 001006         BNE     GT1$15
125 1234 005766         TST     4(SP)
         000004
126 1240 001003         BNE     GT1$15
127 1242 005766         TST     6(SP)
         000006
128 1246 001443         BEQ     LE1$15  ;=1.
129 1250 012766 GT1$15: MOV     #140311,8.(SP)  ;-PI/2
         140311
         000010
```

```
130 1256 012766          MOV     #007732,10.(SP) ;ATAN(X)=PI/2-ATAN(1/X)
         007732
         000012
131 1264 012766          MOV     #121041,12.(SP)
         121041
         000014
132 1272 012766          MOV     #064301,14.(SP)
         064301
         000016
133 1300 005366          DEC     24.(SP) ;ADJUST SIGN
         000030
134 1304 016646          MOV     6(SP),-(SP)     ;MOVE ARG DOWN
         000006
135 1310 016646          MOV     6(SP),-(SP)
         000006
136 1314 016646          MOV     6(SP),-(SP)
         000006
137 1320 016646          MOV     6(SP),-(SP)
         000006
138 1324 012766          MOV     #40200,8.(SP)   ;INSERT 1.
         040200
         000010
139 1332 005066          CLR     10.(SP)
         000012
140 1336 005066          CLR     12.(SP)
         000014
141 1342 005066          CLR     14.(SP)
         000016
142 1346 004467          JSR     R4,$FOLSH       ;COMPUTE 1./X
         010272
143 1352 012210'         .WORD   $DVD,LE1$15
    1354 011356'
144 1356 016646 LE1$15:  MOV     6(SP),-(SP)     ;MOVE ARG DOWN
         000006
145 1362 016646          MOV     6(SP),-(SP)
         000006
146 1366 016646          MOV     6(SP),-(SP)
         000006
147 1372 016646          MOV     6(SP),-(SP)
         000006
148 1376 005066          CLR     8.(SP)  ;INSERT A 0.
         000010
149 1402 005066          CLR     10.(SP)
         000012
150 1406 005066          CLR     12.(SP)
         000014
151 1412 005066          CLR     14.(SP)
         000016
152 1416 021627          CMP     @SP,#037611     ;TAN(15)
         037611
153 1422 103307          BLO     L15$15  ;JUMP IF LESS THAN TAN(15)
154 1424 101016          BHI     TNS$15  ;JUMP IF >
155 1426 026627          CMP     2(SP),#030242
         000002
         030242
156 1434 101012          BHI     TNS$15
157 1436 103501          BLO     L15$15
```

```
158 1440 026627          CMP     4(SP),#172366
         000004
         172366
159 1446 101005          BHI     TNS$15
160 1450 103474          BLO     L15$15
161 1452 026627          CMP     6(SP),#065261
         000006
         065261
162 1460 101470          BLOS    L15$15
163 1462 012766 TNS$15:  MOV     #040006,8.(SP)  ;INSERT PI/6
         040006
         000010
164 1470 012766          MOV     #005221,10.(SP)
         005221
         000012
165 1476 012766          MOV     #140553,12.(SP)
         140553
         000014
166 1504 012766          MOV     #115454,14.(SP)
         115454
         000016
167 1512 011600          MOV     @SP,R0  ;ARG TO REGS
168 1514 016601          MOV     2(SP),R1
         000002
169 1520 016602          MOV     4(SP),R2
         000004
170 1524 016603          MOV     6(SP),R3
         000006
171 1530 012746          MOV     #062524,-(SP)
         062524
172 1534 012746          MOV     #041302,-(SP)
         041302
173 1540 012746          MOV     #131727,-(SP)   ;PUSH -ROOT 3
         131727
174 1544 012746          MOV     #140335,-(SP)
         140335
175 1550 010346          MOV     R3,-(SP)
176 1552 010246          MOV     R2,-(SP)
177 1554 010146          MOV     R1,-(SP)
178 1556 010046          MOV     R0,-(SP)        ;PUSH ARG
179 1560 005046          CLR     -(SP)
180 1562 005046          CLR     -(SP)
181 1564 005046          CLR     -(SP)   ;PUSH 1.
182 1566 012746          MOV     #40200,-(SP)
         040200
183 1572 012746          MOV     #062524,-(SP)
         062524
184 1576 012746          MOV     #041302,-(SP)
         041302
185 1602 012746          MOV     #131727,-(SP)   ;PUSH ROOT3
         131727
186 1606 012746          MOV     #040335,-(SP)
         040335
187 1612 010346          MOV     R3,-(SP)
188 1614 010246          MOV     R2,-(SP)
189 1616 010146          MOV     R1,-(SP)        ;PUSH ARG
190 1620 010046          MOV     R0,-(SP)
```

```
191 1622 004467          JSR     R4,$POLSH       ;TRANSFORM ARG
         010016
192                      ;                       (ROOT3*X-1)/(ROOT3 +X)
193 1626 016146'         .WORD   $MLD,$SBD,UP$15,$SBD,$DVD,L15$15
    1630 000700'
    1632 011776'
    1634 000700'
    1636 012210'
    1640 011642'
194 1642 011600 L15$15:  MOV     @SP,R0  ;GET ARG
195 1644 016601          MOV     2(SP),R1
         000002
196 1650 016602          MOV     4(SP),R2
         000004
197 1654 016603          MOV     6(SP),R3
         000006
198 1660 010346          MOV     R3,-(SP)
199 1662 010246          MOV     R2,-(SP)
200 1664 010146          MOV     R1,-(SP)        ;GET THREE COPIES
201 1666 010046          MOV     R0,-(SP)
202 1670 010346          MOV     R3,-(SP)
203 1672 010246          MOV     R2,-(SP)
204 1674 010146          MOV     R1,-(SP)
205 1676 010046          MOV     R0,-(SP)
206 1700 004467          JSR     R4,$POLSH
         007740
207 1704 016146'         .WORD   $MLD    ;GET ARG**2
208 1706 017576'         .WORD   $POPR4,PLY$15   ;SET UP COEFFICIENTS
    1710 012020'
209 1712 016146'XPU$15:  .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD
    1714 000704'
    1716 016146'
    1720 000704'
    1722 016146'
    1724 000704'
210 1726 016146'         .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD
    1730 000704'
    1732 016146'
    1734 000704'
    1736 016146'
    1740 000704'
211 1742 016146'         .WORD   $MLD,$ADD,$MLD,$ADD,$MLD,$ADD
    1744 000704'
    1746 016146'
    1750 000704'
    1752 016146'
    1754 000704'
212 1756 000704'         .WORD   $ADD    ;P(X)+0 IF X<=1, P(X)=PI/2 IF X>1
213 1760 012064'         .WORD   SGN$15  ;ADJUST SIGN
214 1762 000704'         .WORD   $ADD    ;ADD ATAN2 BIAS
215 1764 017576'         .WORD   $POPR4          ;POP RESULT TO REGS
216 1766 011770'         .WORD   EXI$15
217 1770 005726 EXI$15:  TST     (SP)+   ;POP SIGN FLAG
218 1772 012605          MOV     (SP)+,R5
219 1774 000205          RTS     R5      ;RETURN TO USER
220                      ;
221 1776 012666 UP$15:   MOV     (SP)+,22.(SP)   ;MOVE STACK ITEM UP
```

```
          000026
222 2002  012666          MOV     (SP)+,22.(SP)
          000026
223 2006  012666          MOV     (SP)+,22.(SP)
          000026
224 2012  012666          MOV     (SP)+,22.(SP)
          000026
225 2016  000134          JMP     @(R4)+
226                 ;
227 2020  012704 PLY$15:  MOV     #CON$15+8.,R4   ;POINT TO COEFFICIENT TABLE
          012210'
228 2024  012705          MOV     #9.,R5  ;GET # OF CONSTANTS
          000011
229 2030  000404          BR      PY1$15
230 2032  010346 PY2$15:  MOV     R3,-(SP)
231 2034  010246          MOV     R2,-(SP)
232 2036  010146          MOV     R1,-(SP)        ;PUSH ARG
233 2040  010046          MOV     R0,-(SP)
234 2042  014446 PY1$15:  MOV     -(R4),-(SP)     ;PUSH CONSTANT
235 2044  014446          MOV     -(R4),-(SP)
236 2046  014446          MOV     -(R4),-(SP)
237 2050  014446          MOV     -(R4),-(SP)
238 2052  005305          DEC     R5      ;COUNT
239 2054  003366          BGT     PY2$15
240 2056  012704          MOV     #XPD$15,R4
          011712'
241 2062  000134          JMP     @(R4)+
242                 ;
243 2064  005766 SGN$15:  TST     16.(SP) ;CHECK SIGN FLAG
          000020
244 2070  001402          BEQ     SG1$15
245 2072  062716          ADD     #100000,@SP     ;NEGATE RESULT FOR (-1,0) & (1,I
          100000
246 2076  000134 SG1$15:  JMP     @(R4)+
247                       .ENDC
248                 ;
249                       .IFDF   FPU
250                 DATAN2: SETD    ;               SET OP MODE FOR FPU
251                       MOV     2(R5),R3;       ADDRESS OF ARG1
252                       MOV     4(R5),R4;       ADDRESS OF ARG2
253                       MOV     @R3,R0;         HIGH ORDER ARG1
254                       MOV     @R4,R1;         HIGH ORDER ARG2
255                       BEQ     INF$15;         JUMP IF DENOMINATOR 0
256                       ASL     R0;
257                       CLRB    R0;
258                       SWAB    R0;             EXPONENT OF ARG1
259                       ASL     R1;
260                       CLRB    R1;
261                       SWAB    R1;             EXPONENT OF ARG2
262                       SUB     R1,R0;          GET EXPONENT DIFFERENCE
263                       CMP     #58.,R0;                CHECK MAGNITUDE
264                       BLT     INF$15;         TREAT AS INFINITE
265                       LDD     PI$15,F3;               INITIALIZE BIAS=PI
266                       LDD     @R3,F0;         GET ARG1
267                       CFCC
268                       BGE     A1P$15;         JUMP IF ARG1>0
269                       NEGD    F3;             BIAS=SIGN(ARG1)*PI
```

```
270             A1P$15: LDD     @R4,F1;         GET ARG2
271                     CFCC
272                     BLT     A2M$15;
273                     CLRD    F3;             IF ARG2>0, BIAS=0
274             A2M$15: DIVD    F1,F0;          ARG1/ARG2, SET FLOAT CC
275                     BR      AT1$15;         JOIN MAIN ROUTINE
276             ;
277             INF$15: LDD     PI2$15,F1;              RESULT=SIGN(ARG1)*PI/2
278                     TST     @R3;            TEST ARG1
279                     BGE     EXI$15;         +PI/2
280                     NEGD    F1;             -PI/2
281                     BR      EXI$15;
282             ;
283             DATAN:  SETD    ;               SET DP MODE FOR FPU
284                     CLRD    F3;             CLEAR ATAN2 BIAS
285                     LDD     @2(R5),F0;      GET ARGUMENT
286             AT1$15: CLR     R4;             CLEAR SIGN FLAG
287                     CFCC    ;               GET SIGN OF ARGUMENT
288                     STD     F3,F5;          F5=ATAN2 BIAS
289                     CLRD    F3;             CLEAR QUADRANT BIAS
290                     BGE     PLU$15;         JUMP IF QUADRANT 1 OR 3
291                     ABSD    F0;             ABS(X)
292                     INC     R4;             FLAG -
293             PLU$15: LDD     #1.0,F1;        1.0
294                     CMPD    F0,F1;          CHECK IF X<=1.0
295                     CFCC
296                     BLE     LE1$15;
297             GT1$15: DEC     R4;             X>1.0, ADJUST SIGN FLAG
298                     DIVD    F0,F1;          1.0/X
299                     LDD     F1,F0;          ATAN(X)=PI/2-ATAN(1/X)
300                     LDD     PI2$15,F3;              QUADRANT BIAS=PI/2
301             ;
302             LE1$15: STD     F3,F4;          F4=QUADRANT BIAS
303                     CLRD    F3;             F3=0.0
304                     CMPD    T15$15,F0;      COMPARE TAN(15) : X
305                     CFCC
306                     BGE     L15$15;         X<= TAN(15)
307                     LDD     PI6$15,F3;              F3=PI/6
308                     LDD     F0,F1;
309                     MULD    RT3$15,F0;
310                     SUBD    #1.0,F0;        X*ROOT3-1.0
311                     ADDD    RT3$15,F1;      X+ROOT3
312                     DIVD    F1,F0;          (X*ROOT3-1.0)/(X+ROOT3)
313             ;
314             L15$15: LDD     F0,F2;          X
315                     MULD    F0,F0;          X**2
316                     MOV     #FCO$15,R0;     POINTER TO POLYNOMIAL CONSTANTS
317                     MOV     #8.,R1;         COUNT OF COEFFICIENTS
318                     LDD     (R0)+,F1;       INITIALIZE ACCUMULATOR
319             XPD$15: MULD    F0,F1;
320                     DEC     R1;             COUNT
321                     ADDD    (R0)+,F1;       F1:= F1* X**2 + C(I)
322                     BGT     XPD$15; LOOP
323                     MULD    F2,F1;          F1:= F1*X
324                     ADDD    F3,F1;          PI/6 OR 0.0
325                     SUBD    F4,F1;          P(X)-QUAD BIAS
326                     TST     R4;             TEST SIGN FLAG
```

```
                                BEQ     SG1$15;         NO ADJUSTMENT
                                NEGD    F1;             NEGATE RESULT FOR (-1,0)&(1,INF)
                        SG1$15: ADDD    F5,F1;          ATAN2 BIAS
                                ;
                        EXI$15: STD     F1,-(SP);       MOVE RESULT TO STACK
                                MOV     (SP)+,R0;       AND THEN TO REGISTERS
                                MOV     (SP)+,R1;
                                MOV     (SP)+,R2;
                                MOV     (SP)+,R3;
                                RTS     R5;             EXIT
                                ;
                                ;
                        PI$15:  .WORD   040511,007732;  PI
                                .WORD   121041,054301;
                                ;
                        PI2$15: .WORD   040311,007732;  PI/2
                                .WORD   121041,054301;
                                ;
                        T15$15: .WORD   037611,030242;  TAN(15)
                                .WORD   172366,065261;
                                ;
                        PI6$15: .WORD   040006,005221;  PI/6
                                .WORD   140553,115454;
                                ;
                        RT3$15: .WORD   040335,131727;
                                .WORD   041302,062524;
                                .ENDC
2100 037065             FCU$15: .WORD   037065,150707   ;.0443895157187
2102 150707
2104 162300                     .WORD   162300,163030
2106 163030
                                ;
2110 137204                     .WORD   137204,143233   ;-.06483193510303
2112 143233
2114 004010                     .WORD   004010,000413
2116 000413
                                ;
2120 037235                     .WORD   037235,043002   ;.0767936896066
2122 043002
2124 027154                     .WORD   027154,142446
2126 142446
                                ;
2130 137272                     .WORD   137272,025671   ;-.0909037114191074
2132 025671
2134 116412                     .WORD   116412,065630
2136 065630
                                ;
2140 037343                     .WORD   037343,107047   ;.1111097898051048
2142 107047
2144 023625                     .WORD   023625,025401
2146 025401
                                ;
2150 137422                     .WORD   137422,044444   ;-.1428571402825545
2152 044444
2154 071335                     .WORD   071335,116151
2156 116151
                                ;
```

```
372 2160 037514          .WORD   037514,146314   ;.19999999998729448
    2162 146314
373 2164 146224          .WORD   146224,165650
    2166 165650
374                      ;
375 2170 137652          .WORD   137652,125252   ;-.33333333333329930
    2172 125252
376 2174 125252          .WORD   125252,113602
    2176 113602
377                      ;
378 2200 040200 CONS15:  .WORD   040200,000000   ;.999999999999999
    2202 000000
379 2204 000000          .WORD   000000,000000
    2206 000000
380                      ;
381                      .ENDC
```

```
                            .TITLE  $DVD05
                            .IFDF   CND$16
                            .GLOBL  $DVD,$ERRA
                    ;       $DVD --- THE DOUBLE DIVIDE ROUTINE
                    ;
                    ;       $DVD    V005A
                    ;
                    ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
                    ;       CALLED IN THE POLISH MODE
                    ;       THE NUMERATOR IS THE SECOND ITEM ON THE STACK
                    ;       AND THE DENOMINATOR IS ON TOP.
                    ;       TAKES THE QUOTIENT AND PUTS IT ON TOP
                    ;       OF THE STACK IN THEIR PLACE
            000000          R0=%0
            000001          R1=%1
            000002          R2=%2
            000003          R3=%3
            000004          R4=%4
            000005          R5=%5
            000006          SP=%6
            000007          PC=%7
            000000          F0=%0
            000001          F1=%1
            000010          D=8.
            000020          N=16.
            000020          Q=16.
                            .IFDF   FPU
                    $DVD:   .WORD   170011  ;;SETD
                            .WORD   172526  ;;LDD   (SP)+,F1        ;GET DIVISOR
                            .WORD   172426  ;;LDD   (SP)+,F0        ;GET DIVIDEND
                            .WORD   174401  ;;DIVD  F1,F0   ;GET QUOTIENT
                            .WORD   174046  ;;STD   F0,-(SP)        ;TO STACK
                            JMP     @(R4)+
                            .ENDC
                            .IFNDF  FPU
12210 010446        $DVD:   MOV     R4,-(SP)
12212 010546            MOV     R5,-(SP)
12214 005000            CLR     R0
12216 005001            CLR     R1
12220 005002            CLR     R2
12222 005003            CLR     R3
12224 005046            CLR     -(SP)
12226 006366            ASL     N+0-2(SP)       ;SHIFT NUMERATOR
      000016
12232 006116            ROL     @SP     ;GET NUMERATOR SIGN
12234 005046            CLR     -(SP)
12236 005766            TST     D(SP);          CHECK FOR 0.0 DENOMINATOR
      000010
12242 001521            BEQ     DCH$16;         JUMP TO ERROR EXIT
12244 156616            BISB    N+1(SP),@SP     ;GET NUMERATOR EXPONENT
      000021
12250 001526            BEQ     ZER$16  ;JUMP IF NUMERATOR IS ZERO
12252 156600            BISB    N(SP),R0
      000020
12256 000300            SWAB    R0      ;LEFT JUSTIFY NUMERATOR FRACTION
12260 000261            SEC             ;INSERT NORMAL BIT
12262 006000            ROR     R0
```

```
54 12264 156600          BISB    N+3(SP),R0
         000023
55 12270 156601          BISB    N+2(SP),R1
         000022
56 12274 000301          SWAB    R1
57 12276 156601          BISB    N+5(SP),R1
         000025
58 12302 156602          BISB    N+4(SP),R2
         000024
59 12306 000302          SWAB    R2
60 12310 156602          BISB    N+7(SP),R2
         000027
61 12314 156603          BISB    N+6(SP),R3
         000026
62 12320 000303          SWAB    R3
63 12322 006366          ASL     D(SP)   ;SHIFT DENOMINATOR
         000010
64 12326 005366          ADC     2(SP)   ;GET RESULT SIGN
         000002
65 12332 005004          CLR     R4
66 12334 156604          BISB    D+1(SP),R4      ;GET DIVISOR EXPONENT
         000011
67 12340 160416          SUB     R4,@SP  ;SUBTRACT EXPONENTS
68 12342 000366          SWAB    D(SP)   ;LEFT JUSTIFY DENOMINATOR
         000010
69 12346 000261          SEC             ;INSERT NORMAL BIT
70 12350 006066          ROR     D(SP)
         000010
71 12354 116066          MOVB    D+3(SP),D(SP)
         000013
         000010
72 12362 116066          MOVB    D+2(SP),D+3(SP)
         000012
         000013
73 12370 116066          MOVB    D+5(SP),D+2(SP)
         000015
         000012
74 12376 116066          MOVB    D+4(SP),D+5(SP)
         000014
         000015
75 12404 116066          MOVB    D+7(SP),D+4(SP)
         000017
         000014
76 12412 116066          MOVB    D+6(SP),D+7(SP)
         000016
         000017
77 12420 105066          CLRB    D+6(SP)
         000016
78 12424 005066          CLR     Q(SP)   ;CLEAR QUOTIENT
         000020
79 12430 005066          CLR     Q+2(SP)
         000022
80 12434 005066          CLR     Q+4(SP)
         000024
81 12440 020066          CMP     R0,D(SP)        ;COMPARE HIGH NUM. AND DEN.
         000010
82 12444 101042          BHI     DLW$16  ;JUMP IF DENOMINATOR LOW
```

```
83 12446 103446          BLO     DHI$16  ;JUMP IF DENOMINATOR HIGH
84 12450 020166          CMP     R1,D+2(SP)      ;COMPARE LOW ORDER PARTS
         000012
85 12454 101036          BHI     DLW$16
86 12456 103442          BLO     DHI$16
87 12460 020266          CMP     R2,D+4(SP)
         000014
88 12464 101032          BHI     DLW$16
89 12466 103436          BLO     DHI$16
90 12470 020366          CMP     R3,D+6(SP)
         000016
91 12474 101026          BHI     DLW$16
92 12476 001032          BNE     DHI$16
93 12500 005216          INC     @SP     ;BUMP EXPONENT
94 12502 005004          CLR     R4
95 12504 000465          BR      FLT$16
96 12506 012700 DCM$16:  MOV     #1403,R0        ;ERROR 3,3
         001403
97 12512 000403          BR      EC1$16
98 12514 012700 UND$16:  MOV     #4005,R0        ;ERROR 5,8
         004005
99 12520 005746 ECL$16:  TST     -(SP)   ;FAKE SIGN
100 2522 004567 EC1$16:  JSR     R5,$ERRA
         007270
101 2526 022626 ZER$16:  CMP     (SP)+,(SP)+     ;FLUSH EXP AND SIGN
102 2530 005066          CLR     Q+0-4(SP)
         000014
103 2534 005066          CLR     Q+2-4(SP)
         000016
104 2540 005066          CLR     Q+4-4(SP)
         000020
105 2544 005066          CLR     Q+6-4(SP)
         000022
106 2550 000477          BR      RTN$16
107 2552 006000 DLW$16:  ROR     R0      ;HALVE DENOMINATOR  (C=0)
108 2554 006001          ROR     R1      ;TO ENSURE THAT N<D
109 2556 006002          ROR     R2
110 2560 006003          ROR     R3
111 2562 005216          INC     @SP     ;COMPENSATE EXPONENT
112 2564 012705 DHI$16:  MOV     #9.,R5  ;GO DO FIRST 9 QUOTIENT BITS
         000011
113 2570 004767          JSR     PC,DV1$16
         000176
114 2574 110466          MOVB    R4,Q(SP)        ;SAVE ALL HIGH ORDER Q FRACTION
         000020
115                                              ;EXCEPT NORMAL BIT
116 2600 005705          TST     R5      ;SEE IF DONE
117 2602 001025          BNE     FL1$16  ;YES, REST OF NUMERATOR IS 0
118 2604 012705          MOV     #16.,R5 ;GO DO 16 MORE BITS
         000020
119 2610 004767          JSR     PC,DV1$16
         000156
120 2614 010466          MOV     R4,Q+2(SP)
         000022
121 2620 005705          TST     R5
122 2622 001015          BNE     FL1$16
123 2624 012705          MOV     #16.,R5
```

```
                000020
124 2630 004767         JSR     PC,DV1$16
         000136
125 2634 010466         MOV     R4,Q+4(SP)
         000024
126 2640 005705         TST     R5
127 2642 001005         BNE     FL1$16
128 2644 012705         MOV     #16.,R5
         000020
129 2650 004767         JSR     PC,DV1$16
         000116
130 2654 000401         BR      FLT$16
131 2656 005004 FL1$16: CLR     R4      ;CLEAR LOWEST ORDER QUOTIENT
132 2660 012605 FLT$16: MOV     (SP)+,R5        ;PUSH UP EXPONENT
133 2662 062705         ADD     #200,R5 ;ADD IN EXCESS 200
         000200
134 2666 003712         BLE     UND$16  ;UNDERFLOW
135 2670 022705         CMP     #377,R5
         000377
136 2674 002433         BLT     OVR$16  ;OVERFLOW
137 2676 110566         MOVB    R5,Q+1-2(SP)    ;INSERT EXPONENT IN RESLT
         000017
138 2702 006026 SGN$16: ROR     (SP)+   ;INSERT QUOTIENT SIGN
139 2704 006066         ROR     Q+0-4(SP)
         000014
140 2710 006066         ROR     Q+2-4(SP)
         000016
141 2714 006066         ROR     Q+4-4(SP)
         000020
142 2720 006004         ROR     R4
143 2722 005504         ADC     R4      ;ROUND
144 2724 005566         ADC     Q+4-4(SP)
         000020
145 2730 005566         ADC     Q+2-4(SP)
         000016
146 2734 005566         ADC     Q+0-4(SP)
         000014
147 2740 010466         MOV     R4,Q+6-4(SP)    ;INSERT LOW ORDER FRACTION
         000022
148 2744 103406         BCS     OV1$16
149 2746 102405         BVS     OV1$16
150 2750 012605 RTN$16: MOV     (SP)+,R5
151 2752 012604         MOV     (SP)+,R4
152 2754 062706         ADD     #8.,SP  ;FLUSH FIRST ARGUMENT
         000010
153 2760 000134         JMP     @(R4)+
154 2762 005746 OV1$16: TST     -(SP)   ;FAKE EXP
155 2764 012700 OVR$16: MOV     #2003,R0        ;ERROR 3,4
         002003
156 2770 000653         BR      ECL$16
157 2772 006304 DV1$16: ASL     R4      ;SHIFT QUOTIENT
158 2774 006303         ASL     R3      ;SHIFT NUMERATOR
159 2776 006102         ROL     R2
160 3000 006101         ROL     R1
161 3002 006100         ROL     R0
162 3004 103420         BCS     GO$16   ;GUARANTEED TO GO
163 3006 026600         CMP     D+0+2(SP),R0    ;COMPARE HIGH DIVISOR AND DIVIDE
```

85

```
                000012
164 3012 101034         BHI     NGO$16  ;JUMP IF DIVISOR BIGGER
165 3014 103414         BLO     GO$16   ;JUMP IF DIVISOR SMALLER
166 3016 026601         CMP     D+2+2(SP),R1    ;CHECK THE LOW ORDERS
                000014
167 3022 101030         BHI     NGO$16
168 3024 103410         BLO     GO$16
169 3026 026602         CMP     D+4+2(SP),R2
                000016
170 3032 101024         BHI     NGO$16
171 3034 103404         BLO     GO$16
172 3036 026603         CMP     D+6+2(SP),R3
                000020
173 3042 101020         BHI     NGO$16
174 3044 001422         BEQ     NGO$16  ;JUMP IF NUMERATOR =DENOMINATOR
175 3046 166603 GO$16:  SUB     D+6+2(SP),R3    ;N=N-D
                000020
176 3052 005602         SBC     R2
177 3054 005601         SBC     R1
178 3056 005600         SBC     R0
179 3060 166602         SUB     D+4+2(SP),R2
                000016
180 3064 005601         SBC     R1
181 3066 005600         SBC     R0
182 3070 166601         SUB     D+2+2(SP),R1
                000014
183 3074 005600         SBC     R0
184 3076 166600         SUB     D+0+2(SP),R0
                000012
185 3102 005204         INC     R4      ;INSERT QUOTIENT BIT
186 3104 005305 NGO$16: DEC     R5      ;COUNT LOOP
187 3106 003331         BGT     DV1$16
188 3110 000207         RTS     PC
189 3112 005204 NGO$16: INC     R4      ;INSERT LAST 1 BIT IN QUOTIENT
190 3114 000401         BR      EQ1$16
191 3116 006304 EQ2$16: ASL     R4      ;FINISH OUT QUOTIENT WITH 0'S
192 3120 005305 EQ1$16: DEC     R5
193 3122 003375         BGT     EQ2$16
194 3124 005205         INC     R5      ;FLAG NO MORE NUMERATOR
195 3126 000207 RTS$16: RTS     PC      ;RETURN TO CALLER
196                     .ENDC
197                     .ENDC
```

```
1                               .TITLE  $DVI03
2                               .IFDF   CND$17
3                               .GLOBL  $DVI,$ERR
4                       ;       $DVI  ------THE INTEGER DIVIDE ROUTINE
5                       ;
6                       ;       $DVI    V003A
7                       ;
8                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP. MAYNARD, MASS.
9                       ;       CALLED IN THE POLISH MODE WITH THE NUMERATOR AT 2(SP)
10                      ;       AND THE DENOMINATOR @SP.
11                      ;       RETURNS THE INTEGER QUOTIENT @SP.
12          000000              R0=%0
13          000001              R1=%1
14          000002              R2=%2
15          000003              R3=%3
16          000004              R4=%4
17          000005              R5=%5
18          000006              SP=%6
19          177304              MQ=177304
20                              .IFNDF  EAE&MULDIV
21  13130 005000 $DVI:  CLR     R0          ;CLEAR RESULT SIGN
22  13132 012601        MOV     (SP)+,R1        ;GET DENOMINATOR
23  13134 003003        BGT     P1$17   ;JUMP IF DENOMINATOR PLUS
24  13136 001443        BEQ     CHK$17  ;CAN'T DIVIDE BY ZERO
25  13140 005200        INC     R0          ;NOTE -
26  13142 005401        NEG     R1
27  13144 011603 P1$17: MOV     @SP,R3  ;GET NUMERATOR
28  13146 003003        BGT     P2$17   ;JMP IF NUMERATOR PLUS
29  13150 001434        BEQ     ZER$17  ;JUMP IF IT IS ZERO
30  13152 005200        INC     R0          ;SET RESULT SIGN
31  13154 005403        NEG     R3
32  13156 010446 P2$17: MOV     R4,-(SP)
33  13160 012704        MOV     #8.,R4  ;SET FOR 8 ITERATIONS
          000010
34  13164 005002        CLR     R2          ;CLEAR HIGH ORDER DIVIDEND
35  13166 000303        SWAB    R3          ;TEST HIGH ORDER NUMERATOR
36  13170 001402        BEQ     DIV$17  ;JUMP IF HIGH ORDER QUOTIENT IS 0
37  13172 006304        ASL     R4          ;WE NEED ALL 16 ITERATIONS
38  13174 000303        SWAB    R3          ;UNDO THE ABOVE SWAB
39  13176 006303 DIV$17: ASL    R3          ;DOUBLE DIVIDEND
40  13200 006102        ROL     R2
41  13202 001405        BEQ     LUP$17  ;JUMP IF NO CHANCE THIS TIME
42  13204 005203        INC     R3          ;ASSUME IT WILL GO. INSERT QUOTIENT BIT
43  13206 160102        SUB     R1,R2   ;TRIAL STEP
44  13210 103002        BHIS    LUP$17  ;OK
45  13212 060102        ADD     R1,R2   ;DIVIDEND NOT BIG ENOUGH YET
46  13214 005303        DEC     R3          ;TAKE OUT QUOTIENT BIT
47  13216 005304 LUP$17: DEC    R4
48  13220 003366        BGT     DIV$17  ;GO AGAIN
49  13222 012604        MOV     (SP)+,R4
50  13224 005403        NEG     R3          ;TEST FOR NEGMAX
51  13226 006200        ASR     R0          ;GET RESULT SIGN
52  13230 103402        BCS     P3$17   ;JUMP IF -
53  13232 005403        NEG     R3          ;ANSWER IS POSITIVE
54  13234 102404        BVS     CHK$17  ;JUMP IF ANSWER IS -NEGMAX
55  13236 010316 P3$17: MOV     R3,@SP  ;OUTPUT RESULT
56  13240 000134        JMP     @(R4)+  ;RETURN
```

```
13242 005016 ZER$17: CLR     @SP     ;RESULT IS 0
13244 000134         JMP     @(R4)+
                     .ENDC
                ;    $DVI FOR THE EAE
                     .IFDF   EAE
                $DVI: MOV    #MQ,R0  ;POINT TO MQ
                     MOV     (SP)+,R1        ;GET DIVISOR
                     BEQ     CHK$17  ;JUMP IF DIVISION BY 0
                     MOV     (SP)+,@R0       ;DIVIDEND TO MQ
                     TST     -(R0)   ;SKIP AC
                     MOV     R1,-(R0)        ;DIVISOR TO DIV
                     CMP     (R0)+,(R0)+     ;POINT TO MQ
                     MOV     @R0,-(SP)       ;GET QUOTIENT
                     JMP     @(R4)+  ;RETURN TO USER
                     .ENDC
                ;    $DVI FOR MUL/DIV
                     .IFDF   MULDIV
                $DVI: MOV    2(SP),R1        ;GET LOW ORDER DIVIDEND
                     .WORD   006700  ;;SEX   R0      ;EXTEND SIGN
                     .WORD   071026  ;;DIV   (SP)+,R0        ;DIVIDE
                     MOV     R0,@SP  ;PUSH QUOTIENT
                     BCS     CHK$17  ;JUMP IF ERROR
                     JMP     @(R4)+
                     .ENDC
13246 004567 CHK$17: JSR     R5,$ERR ;ERROR 3,5
      006534
13252 000134         JMP     @(R4)+
13254     003        .BYTE   3
13255     005        .BYTE   5
                     .ENDC
```

```
                        .TITLE  $DVR08
                        .IFDF   CND$18
                        .GLOBL  $DVR,$ERRA
                ;       $DVR --- THE REAL DIVIDE ROUTINE
                ;
                ;
                ;       $DVR    V008A
                ;
                ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
                ;
                ;       CALLED IN THE POLISH MODE
                ;       THE NUMERATOR IS THE SECOND ITEM ON THE STACK
                ;       AND THE DENOMINATOR IS ON TOP.
                ;       TAKES THE QUOTIENT AND PUTS IT ON TOP
                ;       OF THE STACK IN THEIR PLACE
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000007          PC=%7
        177304          MQ=177304
        177312          NOR=177312
        177314          LSH=177314
        177316          ASH=177316
        000000          F0=%0
        000001          F1=%1
        000010          D=8.
        000014          N=12.
        000014          Q=12.
                        .IFDF   FPU
                $DVR:   .WORD   170001  ;;SETF
                        .WORD   172526  ;;LDF   (SP)+,F1        ;GET DIVISOR
                        .WORD   172426  ;;LDF   (SP)+,F0        ;GET DIVIDEND
                        .WORD   174401  ;;DIVF  F1,F0   ;DIVIDE
                        .WORD   174046  ;;STF   F0,-(SP)        ;QUOTIENT TO STC
                        JMP     @(R4)+
                        .ENDC
                        .IFNDF  FPU
13256 010446    $DVR:   MOV     R4,-(SP)
13260 010546            MOV     R5,-(SP)
13262 005000            CLR     R0
13264 005001            CLR     R1
13266 005046            CLR     -(SP)
13270 006366            ASL     N+0-2(SP)       ;SHIFT NUMERATOR
      000012
13274 006116            ROL     @SP     ;GET NUMERATOR SIGN
13276 005046            CLR     -(SP)
13300 005766            TST     D(SP);          CHECK FOR 0,0 DENOMINATOR
      000010
13304 001456            BEQ     DCH$18;
13306 156616            BISB    N+1(SP),@SP     ;GET NUMERATOR EXPONENT
      000015
13312 001451            BEQ     ZER$18  ;JUMP IF NUMERATOR IS ZERO
13314 156600            BISB    N(SP),R0
```

```
                000014
55  13320 000300        SWAB    R0          ;LEFT JUSTIFY NUMERATOR FRACTION
56  13322 000261        SEC                 ;INSERT NORMAL BIT
57  13324 006000        ROR     R0
58  13326 156600        BISB    N+3(SP),R0
                000017
59  13332 156601        BISB    N+2(SP),R1
                000016
60  13336 000301        SWAB    R1
61  13340 005002        CLR     R2
62  13342 005003        CLR     R3
63  13344 006366        ASL     D(SP)       ;SHIFT DENOMINATOR
                000010
64  13350 005566        ADC     2(SP)       ;GET RESULT SIGN
                000002
65  13354 156602        BISB    D+1(SP),R2      ;GET DIVISOR EXPONENT
                000011
66  13360 160216        SUB     R2,@SP      ;SUBTRACT EXPONENTS
67  13362 005002        CLR     R2
68  13364 156602        BISB    D(SP),R2        ;GET HIGH ORDER FRACTION
                000010
69  13370 000302        SWAB    R2
70  13372 000261        SEC                 ;INSERT NORMAL BIT
71  13374 006002        ROR     R2
72  13376 156602        BISB    D+3(SP),R2
                000013
73  13402 156603        BISB    D+2(SP),R3
                000012
74  13406 000303        SWAB    R3
75                      .IFDF   EAE!MULDIV
76                      CLC                 ;ENSURE NUM. AND DENOM. +
77                      ROR     R0;
78                      ROR     R1          ;LOW ORDER R1 AND R3 ARE 0
79                      ROR     R2
80                      ROR     R3
81                      .ENDC
82  13410 020002        CMP     R0,R2       ;COMPARE HIGH NUMERATOR AND DENOMINATOR
83  13412 103440        BLO     DHI$18      ;JUMP IF DENOMINATOR HIGH
84                      .IFNDF  EAE&MULDIV
85  13414 101034        BHI     DLW$18      ;JUMP IF DENOMINATOR LOW
86  13416 020103        CMP     R1,R3       ;COMPARE LOW ORDER PARTS
87  13420 101032        BHI     DLW$18
88  13422 001034        BNE     DHI$18
89  13424 005066        CLR     Q(SP)       ;QUOTIENT FRACTION IS 1
                000014
90  13430 005216        INC     @SP         ;BUMP EXPONENT
91  13432 005005        CLR     R5
92  13434 000445        BR      FLT$18
93                      .ENDC
94                      .IFDF   EAE!MULDIV
95                      BHIS    DLW$18      ;JUMP IF DENOMINATOR LOW OR SAME
96                      .ENDC
97  13436 022626 ZER$18: CMP    (SP)+,(SP)+     ;FLUSH EXP AND SIGN
98  13440 000415        BR      EC1$18
99  13442 005726 DCM$18: TST    (SP)+       ;FLUSH EXP
100 3444  012700        MOV     #4003,R0        ;ERROR 3,8
                004003
```

```
101 3450 000406          BR      ECL$18
102 3452 005746 OV1$18:  TST     -(SP)   ;FAKE SIGN
103 3454 012700 OVR$18:  MOV     #3003,R0        ;ERROR 3,6
         003003
104 3460 000402          BR      ECL$18
105 3462 012700 UND$18:  MOV     #1405,R0        ;ERROR 5,3
         001405
106 3466 005726 ECL$18:  TST     (SP)+   ;FLUSH SIGN
107 3470 004567          JSR     R5,$ERRA
         006322
108 3474 005066 EC1$18:  CLR     Q+0-4(SP)       ;RETURN 0
         000010
109 3500 005066          CLR     Q+2-4(SP)
         000012
110 3504 000445          BR      RTN$18
111 3506 006000 DLW$18:  ROR     R0      ;HALVE NUMERATOR  (C=0)
112 3510 006001          ROR     R1      ;TO ENSURE THAT N<D
113 3512 005216          INC     @SP     ;COMPENSATE EXPONENT
114                      .IFNDF  EAE&MULDIV
115 3514 012704 DHI$18:  MOV     #9.,R4  ;GO DO FIRST 9 QUOTIENT BITS
         000011
116 3520 004767          JSR     PC,DV1$18
         000104
117 3524 110566          MOVB    R5,Q(SP)        ;SAVE ALL HIGH ORDER Q FRACTION
         000014
118                                              ;EXCEPT NORMAL BIT
119 3530 005704          TST     R4      ;SEE IF DONE
120 3532 001402          BEQ     NT0$18  ;NO, NUMERATOR NOT 0
121 3534 005005          CLR     R5      ;ALL THE REST OF THE QUOTIENT IS ZERO
122 3536 000404          BR      FLT$18
123 3540 012704 NT0$18:  MOV     #16.,R4 ;GO DO 16 MORE BITS
         000020
124 3544 004767          JSR     PC,DV1$18
         000060
125                      .ENDC
126                      .IFDF   EAE!MULDIV
127              DHI$18: CLC
128                      ROR     R3      ;ENSURE LOW HALF DENOM. +
129                      ROR     R0      ;SCALE NUMERATOR FOR FIXED PT. DIVIDE
130                      ROR     R1
131                      .ENDC
132                      .IFDF   EAE
133                      MOV     #MQ,R5  ;POINT TO MQ
134                      MOV     R1,@R5  ;NUMERATOR TO AC,MQ
135                      MOV     R0,-(R5)
136                      MOV     R2,-(R5)        ;(A+S*B)/C
137                      TST     (R5)+   ;POINT TO AC
138                      MOV     (R5)+,R1        ;KEEP REMAINDER
139                      MOV     (R5)+,R4        ;KEEP QUOTIENT
140                      MOV     R3,@R5  ;GET Q*D
141                      TST     -(R5)   ;POINT TO MQ
142                      ASR     R1      ;SCALE R
143                      SUB     R1,-(R5)        ;Q*D-R
144                      DEC     @#ASH
145                      MOV     R2,-(R5)        ;(Q*D-R)/C
146                      CMP     (R5)+,(R5)+     ;MQ
147                      NEG     @R5
```

91

```
                        MOV     #2,@#ASH        ;MULT BY 4
                        ADD     R4,-(R5)        ;Q+(Q*D-R)*S/C
                        CLR     @#NOR   ;NORMALIZE
                        SUB     @#NOR,@SP       ;APPLY TO EXPONENT
                        MOV     #-6,@#LSH       ;POSITION NORMAL BIT
                        MOV     (R5)+,Q(SP)     ;STORE QUOTIENT
                        MOV     @R5,R5
                        .ENDC
                        .IFDF   MULDIV
                        MOV     R0,R4   ;NUMERATOR TO DIVIDEND
                        MOV     R1,R5
                        .WORD   071402  ;;      DIV     R2,R4   ;(A+S*B)/C
                        MOV     R5,R1   ;SAVE REMAINDER
                        MOV     R4,R0   ;SAVE QUOTIENT
                        .WORD   070403  ;;      MUL     R3,R4   ;GET Q*D
                        ASR     R1      ;SCALE R
                        SUB     R1,R4   ;Q*D-R
                        .WORD   073427,-1       ;;      ASHC    #-1,R4  ;SCALE
                        .WORD   071402  ;;      DIV     R2,R4   ;GET (Q*D-R)/C
                        NEG     R4      ;(R-Q*D)/C
                        .WORD   073427,-14.     ;;      ASHC    #14.,R4 ;UNSCALE
                        ADD     R0,R4   ;Q+(R-Q*D)*S/C
NBT$18:                 .WORD   073427,1        ;;      ASHC    #1,R4   ;SHIFT
                        BMI     NBI$18  ;CHECK FOR NORMAL BIT
                        DEC     @SP     ;COMPENSATE EXPONENT
                        BR      NBT$18  ;GO AGAIN
NBI$18:                 .WORD   073427,-7       ;;ASHC  #-8,R4  ;ALIGN FRACTION
                        MOV     R4,Q(SP)        ;STORE HIGH ORDER
                        .ENDC
3550 012604 FLT$18:     MOV     (SP)+,R4        ;PUSH UP EXPONENT
3552 062704             ADD     #200,R4 ;ADD IN EXCESS 200
     000200
3556 003741             BLE     UND$18  ;UNDERFLOW
3560 022704             CMP     #377,R4
     000377
3564 002733             BLT     OVR$18  ;OVERFLOW
3566 110466             MOVB    R4,Q+1-2(SP)    ;INSERT EXPONENT IN RESULT
     000013
3572 006026 SGN$18:     ROR     (SP)+   ;INSERT QUOTIENT SIGN
3574 006066             ROR     Q+0-4(SP)
     000010
3600 006005             ROR     R5
3602 005505             ADC     R5      ;ROUND
3604 005566             ADC     Q+0-4(SP)
     000010
3610 010566             MOV     R5,Q+2-4(SP)    ;INSERT LOW ORDER FRACTION
     000012
3614 103716             BCS     OV1$18
3616 102715             BVS     OV1$18
3620 012605 RTN$18:     MOV     (SP)+,R5
3622 012604             MOV     (SP)+,R4
3624 022626             CMP     (SP)+,(SP)+     ;FLUSH FIRST ARGUMENT
3626 000134             JMP     @(R4)+
                        .IFNDF  EAE&MULDIV
3630 006305 DV1$18:     ASL     R5      ;SHIFT QUOTIENT
3632 006301             ASL     R1      ;SHIFT NUMERATOR
3634 006100             ROL     R0
```

```
199 3636 103406          BCS     GO$18   ;GUARANTEED TO GO
200 3640 020200          CMP     R2,R0   ;COMPARE HIGH DIVISOR AND DIVIDEND
201 3642 101010          BHI     NGO$18  ;JUMP IF DIVISOR BIGGER
202 3644 103403          BLO     GO$18   ;JUMP IF DIVISOR SMALLER
203 3646 020301          CMP     R3,R1   ;CHECK THE LOW ORDERS
204 3650 101005          BHI     NGO$18
205 3652 001407          BEQ     NOD$18  ;JUMP IF NUMERATOR =DENOMINATOR
206 3654 160301 GO$18:   SUB     R3,R1   ;N=N-D
207 3656 005600          SBC     R0
208 3660 160200          SUB     R2,R0
209 3662 005205          INC     R5      ;INSERT QUOTIENT BIT
210 3664 005304 NGO$18:  DEC     R4      ;COUNT LOOP
211 3666 003360          BGT     DV1$18
212 3670 000207          RTS     PC
213 3672 005205 NOD$18:  INC     R5      ;INSERT LAST 1 BIT IN QUOTIENT
214 3674 000401          BR      EQ1$18
215 3676 006305 EQ2$18:  ASL     R5      ;FINISH OUT QUOTIENT WITH 0'S
216 3700 005304 EQ1$18:  DEC     R4
217 3702 003375          BGT     EQ2$18
218 3704 005204          INC     R4      ;FLAG NO MORE NUMERATOR
219 3706 000207 RTS$18:  RTS     PC      ;RETURN TO CALLER
220                      .ENDC
221                      .ENDC
222                      .ENDC
```

```
                        .TITLE  $DXP05
                        .IFDF   CND$19
                ;
                ;       DEXP    V005A
                ;
                ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                ;
                        .GLOBL  DEXP,$ERRA;
                        .IFNDF  FPU
                        .GLOBL  $ADD,$SBD,$MLD,$DVD,$ID,$DI,$POLSH,$POPR4;
                        .ENDC
                ;       THE FORTRAN DEXP FUNCTION
                ;       CALLING SEQUENCE:
                ;       JSR     R5,DEXP
                ;       BR      A
                ;       .WORD   ARGUMENT ADDRESS
                ;A:
                ;       RETURNS E**ARG IN R0 - R3.
                ;
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000000          F0=%0
        000001          F1=%1
        000002          F2=%2
        000003          F3=%3
                        .IFDF   FPU
                DEXP:   MOV     @2(R5),R0;      GET HIGH ORDER ARG
                        .ENDC
                        .IFNDF  FPU
13710 010546    DEXP:   MOV     R5,-(SP)        ;SAVE RETURN
13712 016504            MOV     2(R5),R4        ;GET ARG POINTER
      000002
13716 011400            MOV     @R4,R0  ;GET HIGH ORDER ARG
                        .ENDC
13720 003004            BGT     POS$19  ;JUMP IF +
13722 020027            CMP     R0,#141662      ;ARG IS -
      141662
13726 101062            BHI     ZER$19  ;JUMP IF ARG <88.7
13730 000403            BR      SMT$19  ;JUMP TO TEST SMALL MAGNITUDE ARG
13732 020027    POS$19: CMP     R0,#41660
      041660
13736 101053            BHI     OVR$19  ;JUMP IF ARG >87
13740 006300    SMT$19: ASL     R0      ;DUMP SIGN
13742 020027            CMP     R0,#43000
      043000
13746 103444            BLO     ONE$19  ;JUMP IF ARG MAGNITUDE <2**-60
                        .IFNDF  FPU
13750 162706            SUB     #20.,SP ;GET WORK SPACE
      000024
13754 062704            ADD     #8.,R4  ;POINT TO LOW ORDER ARG
      000010
13760 014446            MOV     -(R4),-(SP)     ;PUSH ARG
```

```
52 13762 014446          MOV     -(R4),-(SP)
53 13764 014446          MOV     -(R4),-(SP)
54 13766 014446          MOV     -(R4),-(SP)
55 13770 012746          MOV     #013761,-(SP)   ;PUSH LOG2(E)
         013761
56 13774 012746          MOV     #024534,-(SP)
         024534
57 14000 012746          MOV     #125073,-(SP)
         125073
58 14004 012746          MOV     #40270,-(SP)
         040270
59 14010 004467          JSR     R4,$POLSH       ;ENTER POLISH MODE
         005630
60 14014 016146!         .WORD   $MLD    ;Y=X*LOG2(E)
61 14016 014454!         .WORD   DUP$19
62 14020 017640!         .WORD   $DI     ;INT(X*LOG2(E))
63 14022 014344!         .WORD   ADJ$19
64 14024 016046!         .WORD   $ID     ;Z=INT(X*LOG2(E)),Y>=0; Z=Z-1,Y<0
65 14026 000700!         .WORD   $SBD
66 14030 014362!         .WORD   M16$19  ;D=16*(X*LOG2(E)-FLOAT(Z))
67 14032 014454!         .WORD   DUP$19  ;2 COPIES
68 14034 017640!         .WORD   $DI
69 14036 014402!         .WORD   DSV$19  ;SAVE INTEGER PART OF 2**Y
70 14040 016046!         .WORD   $ID     ;E=D-INT(D)
71 14042 000700!         .WORD   $SBD,D16$19     ;E/16
   14044 014370!
72 14046 014454!         .WORD   DUP$19,DUP$19   ;GET 3 COPIES
   14050 014454!
73 14052 016146!         .WORD   $MLD    ;E*E
74 14054 017576!         .WORD   $POPR4          ;POP E*E TO REGS
75 14056 014116!         .WORD   UPL$19
76 14060 012700 ONE$19:  MOV     #40200,R0      ;RESULT IS 1.
         040200
77 14064 000410          BR      Z1$19
78 14066 012700 OVR$19:  MOV     #1004,R0       ;ERROR 4,2
         001004
79 14072 000402          BR      ECL$19
80 14074 012700 ZER$19:  MOV     #2005,R0       ;ERROR 5,4
         002005
81 14100 004567 ECL$19:  JSR     R5,$ERRA
         005712
82 14104 005000          CLR     R0      ;RESULT IS 0
83 14106 005001 Z1$19:   CLR     R1
84 14110 005002          CLR     R2
85 14112 005003          CLR     R3
86 14114 000511          BR      OUT$19
87 14116 012746 UPL$19:  MOV     #033343,-(SP)   ;PUSH P0=7.2135034108448190B3
         033343
88 14122 012746          MOV     #015345,-(SP)
         015345
89 14126 012746          MOV     #152405,-(SP)
         152405
90 14132 012746          MOV     #040746,-(SP)
         040746
91                       ;
92 14136 010346          MOV     R3,-(SP)
93 14140 010246          MOV     R2,-(SP)
```

95

```
                                ;
14142 010146                    MOV     R1,-(SP)
14144 010046                    MOV     R0,-(SP)
                                ;
14146 012746                    MOV     #153703,-(SP)   ;PUSH P1=.057761135831801928
      153703
14152 012746                    MOV     #153011,-(SP)
      153011
14156 012746                    MOV     #113360,-(SP)
      113360
4162 012746                     MOV     #037154,-(SP)
      037154
                                ;
4166 010346                     MOV     R3,-(SP)
4170 010246                     MOV     R2,-(SP)
4172 010146                     MOV     R1,-(SP)
4174 010046                     MOV     R0,-(SP)
                                ;
4176 012746                     MOV     #171042,-(SP)   ;PUSH Q0=20.8137711965230362973
      171042
4202 012746                     MOV     #074433,-(SP)
      074433
4206 012746                     MOV     #101232,-(SP)
      101232
4212 012746                     MOV     #041246,-(SP)
      041246
                                ;
4216 004467                     JSR     R4,$POLSH
      005422
4222 000704!                    .WORD   $ADD,AUP$19     ;A=E*E+Q0 TO WORK SPACE
     4224 014410!
4226 016146!                    .WORD   $MLD,$ADD,$MLD  ;B=E*(E*E+P1+P0)
     4230 000704!
     4232 016146!
4234 014476!                    .WORD   TWC$19  ;DUPLICATE A AND B
4236 000704!                    .WORD   $ADD,ABP$19     ;A+B TO WORD SPACE
     4240 014432!
4242 000700!                    .WORD   $SBD,$DVD       ;(A+B)/(A-B)
     4244 012210!
4246 014250!                    .WORD   SCL$19  ;APPLY SCALE FACTORS
4250 012705 SCL$19: MOV         #RT2$19+8.,R5   ;POINT TO POWERS OF 2
      014554!
4254 006266 ASR$19: ASR         8.(SP)  ;SHIFT D
      000010
4260 103010                     BCC     NML$19  ;JUMP IF BIT IS OFF
4262 014546                     MOV     -(R5),-(SP)     ;PUSH 2**((2**N)*D/16)
4264 014546                     MOV     -(R5),-(SP)
4266 014546                     MOV     -(R5),-(SP)
4270 014546                     MOV     -(R5),-(SP)
4272 004467                     JSR     R4,$POLSH
      005346
4276 016146!                    .WORD   $MLD,ASR$19     ;MULTIPLY BY ABOVE FACTOR AND TE
     4300 014254!
4302 001403 NML$19: BEQ         SC1$19
4304 162705                     SUB     #8.,R5  ;POINT TO NEXT POWER OF 2
      000010
4310 000761                     BR      ASR$19
4312 012600 SC1$19: MOV         (SP)+,R0        ;POP RESULT
```

```
132 4314 012601          MOV     (SP)+,R1
133 4316 012602          MOV     (SP)+,R2
134 4320 012603          MOV     (SP)+,R3
135 4322 005726          TST     (SP)+   ;FLUSH D
136 4324 012604          MOV     (SP)+,R4        ;GET Z
137 4326 000304          SWAB    R4
138 4330 105004          CLRB    R4      ;MAKE INTO EXPONENT MODIFIER
139 4332 006204          ASR     R4
140 4334 060400          ADD     R4,R0   ;APPLY TO RESULT
141 4336 100653          BMI     OVR$19  ;JUMP IF OVERFLOW
142 4340 012605 OUT$19:  MOV     (SP)+,R5        ;POP RETURN
143 4342 000205          RTS     R5      ;RETURN TO USER
144                      ;
145 4344 005775 ADJ$19:  TST     @2(R5)  ;TEST X
            000002
146 4350 002001          BGE     ARN$19  ;JUMP IF +
147 4352 005316          DEC     @SP     ;Z=Z-1
148 4354 011666 ARN$19:  MOV     @SP,28.(SP)     ;SAVE Z AS AN INTEGER
            000034
149 4360 000134          JMP     @(R4)+
150                      ;
151 4362 062716 M16$19:  ADD     #1000,@SP       ;16* STACK ITEM
            001000
152 4366 000134          JMP     @(R4)+
153                      ;
154 4370 162716 D16$19:  SUB     #1000,@SP       ;1/16*STACK ITEM
            001000
155 4374 100001          BPL     D6R$19  ;JUMP IF NO UNDERFLOW
156 4376 005016          CLR     @SP     ;UNDERFLOW=0
157 4400 000134 D6R$19:  JMP     @(R4)+
158                      ;
159 4402 011666 DSV$19:  MOV     @SP,26.(SP)     ;SAVE D AS AN INTEGER
            000032
160 4406 000134          JMP     @(R4)+
161                      ;
162 4410 012666 AUP$19:  MOV     (SP)+,38.(SP)   ;A TO WORK SPACE
            000046
163 4414 012666          MOV     (SP)+,38.(SP)
            000046
164 4420 012666          MOV     (SP)+,38.(SP)
            000046
165 4424 012666          MOV     (SP)+,38.(SP)
            000046
166 4430 000134          JMP     @(R4)+
167                      ;
168 4432 012666 ABP$19:  MOV     (SP)+,22.(SP)   ;MOVE A+B TO WORD SPACE
            000026
169 4436 012666          MOV     (SP)+,22.(SP)
            000026
170 4442 012666          MOV     (SP)+,22.(SP)
            000026
171 4446 012666          MOV     (SP)+,22.(SP)
            000026
172 4452 000134          JMP     @(R4)+
173                      ;
174 4454 016646 DUP$19:  MOV     6(SP),-(SP)     ;DUPLICATE STACK ITEM
            000006
```

```
175 4460 016646          MOV     6(SP),-(SP)
         000006
176 4464 016646          MOV     6(SP),-(SP)
         000006
177 4470 016646          MOV     6(SP),-(SP)
         000006
178 4474 000134          JMP     @(R4)+
179                      ;
180 4476 012700 TWO$19:  MOV     #8.,R0  ;EIGHT ITEMS
         000010
181 4502 016646 TW1$19:  MOV     14.(SP),-(SP)   ;DUPLICATE 2 DOUBLES
         000016
182 4506 005300          DEC     R0
183 4510 003374          BGT     TW1$19
184 4512 000134          JMP     @(R4)+
185                      ;
186                      ;
187 4514 040265          .WORD   040265,002363,031771,157145     ;2**1/2
    4516 002363
    4520 031771
    4522 157145
188 4524 040230          .WORD   040230,033760,050615,134251     ;2**1/4
    4526 033760
    4530 050615
    4532 134251
189 4534 040213          .WORD   040213,112701,161752,105727     ;2**1/8
    4536 112701
    4540 161752
    4542 105727
190 4544 040205 RT2$19:  .WORD   040205,125303,063714,044173     ;2**1/16
    4546 125303
    4550 063714
    4552 044173
191                      .ENDC
192                      ;
193                      .IFDF   FPU
194                      SETD    ;               DOUBLE PRECISION FP
195                      SETI    ;               SHORT INTEGERS
196                      MOV     #FCO$19,R0;     POINTER TO CONSTANTS
197                      LDD     @2(R5),F2;      GET ARGUMENT
198                      MODD    (R0)+,F2;       F2=FRACT(X*LOG2(E))
199                      STCDI   F3,R4;          Z=INT(X*LOG2(E))
200                      TSTD    F2;
201                      CFCC    ;
202                      BGE     M16$19;         TEST F2
203                      ADDD    #1.0,F2;        MAKE F2 POSITIVE
204                      DEC     R4;             AND ADJUST Z=Z-1
205                      ;
206                      M16$19: MODD    #16.0,F2;       F2=FRACT(16*(X*LOG2(E)-FLOAT(Z))
207                              STCDI   F3,R5;          D=INT   (16*(...
208                              DIVD    #16.0,F2;       E=F2/16
209                              LDD     F2,F3;
210                              MULD    F3,F3;          E*E
211                      ;
212                              LDD     F3,F1;
213                              ADDD    (R0)+,F1;       A=E*E+Q0
214                              MULD    (R0)+,F3;
```

```
                        ADDD    (R0)+,F3;
                        MULD    F2,F3;          B=(E*E*P1 + P0)*E
                        LDD     F1,F0;
                        ADDD    F3,F0;          A+B
                        SUBD    F3,F1;          A-B
                        DIVD    F1,F0;          (A+B)/(A-B)
                ;
SCL$19:                 ASR     R3;             SHIFT D
                        BCC     NML$19;
                        MULD    (R0)+,F0;       MULTIPLY BY ROOT OF 2
                        BR      SCL$19;
NML$19:                 BEQ     SC1$19;
                        ADD     #8.,R0;         POINT TO NEXT ROOT OF 2
                        BR      SCL$19;
                ;
SC1$19:                 STD     F0,-(SP);       MOVE RESULT TO STACK
                        MOV     (SP)+,R0;       AND THENCE TO R0...R3
                        MOV     (SP)+,R1;
                        MOV     (SP)+,R2;
                        MOV     (SP)+,R3;
                        SWAB    R4;             CONVERT Z TO EXPONENT MODIFIER
                        CLRB    R4;
                        ASR     R4;
                        ADD     R4,R0;          APPLY TO RESULT
                        BMI     OVR$19;         JUMP IF OVERFLOW
                        RTS     R5;             EXIT
                ;
ONE$19:                 MOV     #40200,R0       ;RESULT IS 1.
                        BR      Z1$19
OVR$19:                 MOV     #1004,R0        ;ERROR 4,2
                        BR      ECL$19
ZER$19:                 MOV     #2005,R0        ;ERROR 5,4
ECL$19:                 JSR     R5,$ERRA
                        CLR     R0      ;RESULT IS 0
Z1$19:                  CLR     R1
                        CLR     R2
                        CLR     R3
                        RTS     R5;             EXIT
                ;
                ;       ORDER-DEPENDENT CONSTANTS
                ;       R0 POINTS AT NEXT CONSTANT IN FPU VERSION
                ;
FCO$19:                 .WORD   40270,125073,024534,013761;     LOG2(E)
                ;
                        .WORD   041246,101232,074433,171042;    Q0
                        .WORD   037154,113360,153011,153703;    P1
                        .WORD   040746,152405,015345,033343;    P0
                        .WORD   040205,125303,063714,044173;    2**1/16
                        .WORD   040213,112701,161752,105727;    2**1/8
                        .WORD   040230,033760,050615,134251;    2**1/4
                        .WORD   040265,002363,031771,157145;    2**1/2
                        .ENDC
                        .ENDC
```

```
1                               .TITLE  $EXP04
2                               .IFDF   CND$20
3                       ;
4                       ;       EXP     V004A
5                       ;
6                       ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
7                       ;
8
9                               .GLOBL  EXP,$ERRA;
10                              .IFNDF  FPU
11                              .GLOBL  $ADR,$SBR,$MLR,$DVR,$IR,$RI,$POLSH;
12                              .ENDC
13                      ;       EXP     THE REAL EXPONENTIATION ROUTINE
14                      ;       CALLING SEQUENCE:
15                      ;       JSR     R5,EXP
16                      ;       BR      A
17                      ;       .WORD   ARG ADDRESS
18                      ;A:
19                      ;       RETURNS EXPONENTIAL IN R0 AND R1.
20                      ;
21              000000          R0=%0
22              000001          R1=%1
23              000002          R2=%2
24              000003          R3=%3
25              000004          R4=%4
26              000005          R5=%5
27              000006          SP=%6
28              000007          PC=%7
29              000000          F0=%0
30              000001          F1=%1
31              000002          F2=%2
32              000003          F3=%3
33  14554 016504 EXP:   MOV     2(R5),R4        ;GET ARGUMENT POINTER
                000002
34  14560 011400        MOV     @R4,R0  ;GET HIGH ORDER ARG
35  14562 003004        BGT     POS$20  ;JUMP IF ARG +
36  14564 020027        CMP     R0,#141662
                141662
37  14570 101146        BHI     ZER$20  ;JUMP IF EXPONENT < -88.7
38  14572 000403        BR      SMT$20
39  14574 020027 POS$20: CMP    R0,#41660
                041660
40  14600 101137        BHI     OVR$20  ;JUMP IF EXPONENT > 87
41  14602 006300 SMT$20: ASL    R0      ;DUMP SIGN
42  14604 020027        CMP     R0,#63000
                063000
43  14610 103527        BLO     ONE$20  ;JUMP IF EXPONENT MAGNITUDE < 2**-28
44                              .IFNDF  FPU
45  14612 005746        TST     -(SP)   ;SAVE SPACE FOR SCALE
46  14614 005046        CLR     -(SP)   ;PUSH A 1.
47  14616 012746        MOV     #40200,-(SP)
                040200
48  14622 016446        MOV     2(R4),-(SP)     ;GET LOW ORDER ARGUMENT
                000002
49  14626 011446        MOV     @R4,-(SP)       ;HIGH ORDER
50  14630 016445        MOV     2(R4),-(SP)     ;NEED TWO COPIES OF IT
                000002
```

```
51 14634 011446          MOV     @R4,-(SP)
52 14636 004467          JSR     R4,$POLSH       ;ENTER POLISH MODE
         005002
53 14642 014732!         .WORD   PLE$20  ;PUSH LOG2(E)
54 14644 017162!         .WORD   $MLR
55 14646 017650!         .WORD   $RI     ;FIX LOG2(E)*X
56 14650 014744!         .WORD   ESV$20  ;SAVE EXPONENT SCALE
57 14652 016062!         .WORD   $IR     ;FLOAT IT
58 14654 014732!         .WORD   PLE$20  ;PUSH LOG2(E)
59 14656 013256!         .WORD   $DVR
60 14660 002004!         .WORD   $SBR
61 14662 014752!         .WORD   CFR$20  ;PUSH CONTINUED FRACTION CONSTANTS
62 14664 017162!         .WORD   $MLR    ;Y*Y
63 14666 002010!         .WORD   $ADR    ;B1+Y*Y
64 14670 013256!         .WORD   $DVR    ;A1/(B1+Y*Y)
65 14672 002010!         .WORD   $ADR    ;Y+A1/(B1+Y*Y)
66 14674 002010!         .WORD   $ADR    ;A0+Y+A1/(B1+Y*Y)
67 14676 013256!         .WORD   $DVR    ;Y/(A0+Y+A1/(B1+Y*Y))
68 14700 014712!         .WORD   INC$20  ;-2*Y/(A0+Y+A1/(B1+Y*Y))
69 14702 002010!         .WORD   $ADR    ;1-2*Y/.........
70 14704 014720!         .WORD   DUP$20  ;DUPLICATE IT
71 14706 017162!         .WORD   $MLR    ;(1-2*Y/.....)**2
72 14710 015046!         .WORD   SCL$20  ;EXIT POLISH MODE AND SCALE RESULT
73 14712 062716 INC$20: ADD     #100200,@SP     ;MULTIPLY BY -2.0
         100200
74 14716 000134          JMP     @(R4)+  ;GO BACK TO LIST
75                       ;
76 14720 016646 DUP$20: MOV     2(SP),-(SP)     ;DUPLICATE STACK ITEM
         000002
77 14724 016646          MOV     2(SP),-(SP)
         000002
78 14730 000134          JMP     @(R4)+
79                       ;
80 14732 012746 PLE$20: MOV     #125073,-(SP)   ;PUSH LOG2(E)
         125073
81 14736 012746          MOV     #40270,-(SP)
         040270
82 14742 000134          JMP     @(R4)+
83                       ;
84 14744 011666 ESV$20: MOV     @SP,10,(SP)     ;SAVE EXPONENT SCALE
         000012
85 14750 000134          JMP     @(R4)+
86                       ;
87 14752 006116 CFR$20: ROL     @SP     ;SHIFT MODIFIED ARG
88 14754 006100          ROL     R0      ;SAVE SIGN
89 14756 162716          SUB     #400,@SP        ;DIVIDE BY 2.
         000400
90 14762 101430          BLOS    ZFR$20  ;UNDERFLOW. MAKE ARG 0
91 14764 006000          ROR     R0      ;GET SIGN BACK
92 14766 006016          ROR     @SP
93 14770 011600          MOV     @SP,R0  ;GET MODIFIED ARGUMENT
94 14772 016601          MOV     2(SP),R1        ;IN REGISTERS
         000002
95 14776 012746          MOV     #036602,-(SP)   ;PUSH -12,01501675 **********
         036602
96 15002 012746          MOV     #141100,-(SP)
         141100
```

```
97  15006 010146          MOV     R1,-(SP)
98  15010 010046          MOV     R0,-(SP)
99  15012 012746          MOV     #071571,-(SP)   ;PUSH 601.8042667 **************
          071571
100 5016  012746          MOV     #042426,-(SP)
          042426
101 5022  012746          MOV     #056133,-(SP)   ;PUSH 60.0901907 ***********
          056133
102 5026  012746          MOV     #041560,-(SP)
          041560
103 5032  010146          MOV     R1,-(SP)
104 5034  010046          MOV     R0,-(SP)
105 5036  010146          MOV     R1,-(SP)
106 5040  010046          MOV     R0,-(SP)
107 5042  000134          JMP     @(R4)+
108                       .ENDC
109             ;
110                       .IFDF   FPU
111                       SETD    ;                   DOUBLE PRECISION ARGUMENT REDUCT
112                       SETI    ;                   SHORT INTEGERS
113                       MOV     #FCO$20,R0;         POINTER TO CONSTANTS
114                       LDCFD   @R4,F2;             GET ARGUMENT
115                       MODD    (R0)+,F2;           F2=FRACT(X*LOG2(E))
116                       STCDI   F3,R4;              R4=INT  (X*LOG2(E))
117                       LDD     #1.0,F0;            F0=1.0
118                       DIVD    (R0)+,F2;           Y=F2/(2*LOG2(E))
119                       SETF    ;
120                       LDCDF   F2,F2;              REST IN SINGLE PRECISION
121                       CFCC    ;                   TEST FOR UNDERFLOW
122                       BEQ     SC1$20;             APPROXIMATION RESULT IS 1.0
123                       LDF     F2,F3;
124                       MULF    F3,F3;              Y*Y
125                       ADDF    (R0)+,F3;           B1+Y*Y
126                       LDF     (R0)+,F1;
127                       DIVF    F3,F1;              A1/(B1+Y*Y)
128                       ADDF    F2,F1;
129                       ADDF    (R0)+,F1;           A0+Y+A1/(B1+Y*Y)
130                       DIVF    F1,F2;              Y/(A0+Y+A1/(B1+Y*Y))
131                       MULF    #2.0,F2;
132                       SUBF    F2,F0;              1-2*Y/. . .
133                       MULF    F0,F0;              (1-2*Y/. . . )**2
134             SC1$20:   STF     F0,-(SP);           MOVE APPROXIMATION TO STACK
135                       .ENDC
136             ;
137                       .IFNDF  FPU
138 5044  022626 ZFR$20:  CMP     (SP)+,(SP)+         ;FLUSH CFRACT ARG
139             ;                                     RESULT IS 1.
140                       .ENDC
141 5046  012600 SCL$20:  MOV     (SP)+,R0;           GET APPROXIMATION RESULT
142 5050  012601          MOV     (SP)+,R1;
143                       .IFNDF  FPU
144 5052  012604          MOV     (SP)+,R4;           GET INT(X*LOG2(E))
145                       .ENDC
146 5054  000304          SWAB    R4;                 MAKE INTO EXPONENT MODIFIER
147 5056  105004          CLRB    R4;
148 5060  006204          ASR     R4;
149 5062  060400          ADD     R4,R0;              ADD IN EXPONENT MODIFIER
```

```
150 5064 100405          BMI     OVR$20;          TEST OVERFLOW
151 5066 000205          RTS     R5;
152                      ;
153 5070 005001 ONE$20:  CLR     R1
154 5072 012700          MOV     #40200,R0        ;EXP(TINY) = 1.
         040200
155 5076 000205          RTS     R5
156 5100 012700 OVR$20:  MOV     #2404,R0         ;ERROR 4,5
         002404
157 5104 000402          BR      ECL$20
158 5106 012700 ZER$20:  MOV     #2405,R0         ;ERROR 5,5
         002405
159 5112 004567 ECL$20:  JSR     R5,$ERRA
         004700
160 5116 005000          CLR     R0       ;RETURN 0
161 5120 005001          CLR     R1
162 5122 000205          RTS     R5
163                      ;
164                      .IFDF   FPU
165                      ;       ORDER-DEPENDENT CONSTANTS
166                      ;
167                      FCV$20: .WORD   040270,125073;  LOG2(E) DOUBLE PRECISION
168                              .WORD   024534,013761;
169                      ;
170                              .WORD   040470,125073;  2*LOG2(E) DOUBLE PRECISION
171                              .WORD   024534,013761;
172                      ;
173                              .WORD   041560,056133;  B1=60.0901907
174                      ;
175                              .WORD   042426,071571;  A1=601.8042667
176                      ;
177                              .WORD   141100,036602;  A0=-12.01501675
178                              .ENDC
179                              .ENDC
```

```
1                               .TITLE  $FCL02
2                               .IFDF   CND$21
3                       ;
4                       ;       $FCALL  V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  $FCALL
9                       ;       $FCALL --- ROUTINE FOR CALLING SINGLE ARG FORTRAN
10                      ;       FUNCTIONS FROM WITHIN OTHER FORTRAN FUNCTIONS.
11                      ;       CALLING SEQUENCE:
12                      ;       MOV     ARG POINTER,R5
13                      ;       MOV     #FUNCTION NAME,R4
14                      ;       JSR     PC,$FCALL
15                      ;       FLUSH ARGUMENT
16              000000          R0=%0
17              000004          R4=%4
18              000005          R5=%5
19              000006          SP=%6
20  15124 012746 $FCALL: MOV     #RET$21,-(SP)   ;PUSH $FCALL RETURN
                015146'
21  15130 012746         MOV     #137,-(SP)      ;JMP    @PC
                000137
22  15134 010546         MOV     R5,-(SP)        ;.WORD  ARG
23  15136 012746         MOV     #401,-(SP)      ;BR     .+4
                000401
24  15142 010605         MOV     SP,R5           ;JSR    R5,FUNCT
25  15144 004014         JSR     R0,@R4
26  15146 062706 RET$21: ADD     #8.,SP  ;FLUSH CALL
                000010
27  15152 000136         JMP     @(SP)+  ;RETURN TO USER WITH ARG ON STACK
28                      ;                       AND FUNCT(ARG) IN REGS.
29                              .ENDC
```

```
1                               .TITLE  $FIX03
2                               .IFDF   CND$22
3                       ;
4                       ;       IFIX    V003A
5                       ;
6                       ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
7                       ;
8                               .GLOBL  IFIX,$RI,$POLSH
9                       ;       THE FORTRAN IFIX FUNCTION
10                      ;       CALLING SEQUENCE:
11                      ;       JSR     R5,IFIX
12                      ;       BR      A
13                      ;       .WORD   ARGUMENT ADDRESS
14                      ;A:
15                      ;       RETURNS THE TRUNCATED AND FIXED REAL
16                      ;       ARGUMENT AS AN INTEGER IN R0.
17                      ;
18              000000          R0=%0
19              000004          R4=%4
20              000005          R5=%5
21              000006          SP=%6
22  15154 016504 IFIX:  MOV     2(R5),R4        ;GET ARG ADDRESS
              000002
23  15160 016446        MOV     2(R4),-(SP)     ;PUSH ARG
              000002
24  15164 011446        MOV     @R4,-(SP)
25  15166 004467 RND$22: JSR    R4,$POLSH       ;ENTER POLISH MODE
              004452
26  15172 017650'       .WORD   $RI,UPL$22      ;TRUNCATE AND FIX
    15174 015176'
27  15176 012600 UPL$22: MOV    (SP)+,R0        ;POP INTEGER RESULT
28  15200 000205        RTS     R5      ;RETURN TO CALLER
29                              .ENDC
```

```
1                               .TITLE  $FLT02
2                               .IFDF   CND$23
3                       ;
4                       ;       FLOAT   V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  FLOAT,$IR,$POLSH,$POPR3
9                       ;       FLOAT   THE FORTRAN FLOAT FUNCTION
10                      ;       CALLING SEQUENCE:
11                      ;       JSR     R5,FLOAT
12                      ;       BR      A
13                      ;       .WORD   ADDRESS OF INTEGER
14                      ;A:
15                      ;       RETURNS REAL EQUIVALENT IN R0 AND R1.
16                      ;       USES $IR.
17                      ;
18              000000          R0=%0
19              000001          R1=%1
20              000004          R4=%4
21              000005          R5=%5
22              000006          SP=%6
23  15202 017546 FLOAT: MOV     @2(R5),-(SP)    ;GET ARGUMENT ON STACK
                000002
24  15206 004467            JSR     R4,$POLSH       ;ENTER POLISH MODE
                004432
25  15212 016062'           .WORD   $IR     ;CALL $IR TO CONVERT TO REAL
26  15214 017610'           .WORD   $POPR3  ;POP RESULT TO REGS
27  15216 015220'           .WORD   UPL$23
28  15220 000205 UPL$23: RTS    R5      ;RETURN TO CALLER
29                              .ENDC
```

```
1                               .TITLE  $ICI02
2                               .IFDF   CND$24
3                       ;
4                       ;       $ICI    V002A
5                       ;
6                       ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
7                       ;
8                               .GLOBL  $ICI,$OCI
9                       ;       $OCI    ASCII TO OCTAL CONVERSION
10                      ;       $ICI    ASCII TO INTEGER CONVERSION
11                      ;       CALLING SEQUENCE:
12                      ;       PUSH    CHARACTER FIELD START
13                      ;       PUSH    CHARACTER FIELD LENGTH
14                      ;       JSR     PC,$ICI   OR $OCI
15                      ;       RETURNS WITH INTEGER RESULT ON TOP OF STACK.
16              000000          R0=%0
17              000001          R1=%1
18              000002          R2=%2
19              000006          SP=%6
20              000007          PC=%7
21  15222 012746 $OCI:  MOV     #67,-(SP)       ;SET OCTAL FLAGS
                000067
22  15226 000402        BR      GO$24
23  15230 012746 $ICI:  MOV     #471,-(SP)      ;SET DECIMAL FLAGS
                000471
24  15234 010146 GO$24: MOV     R1,-(SP)        ;SAVE R1
25  15236 016601        MOV     8.(SP),R1       ;GET STRING START
                000010
26  15242 066666        ADD     6(SP),8.(SP)    ;GET END+1
                000006
                000010
27  15250 016665        MOV     4(SP),6(SP)     ;FIDDLE RETURN POINTER
                000004
                000006
28  15256 010066        MOV     R0,4(SP)        ;SAVE R0
                000004
29  15262 010246        MOV     R2,-(SP)        ;SAVE R2
30  15264 005046        CLR     -(SP)   ;CLEAR SIGN
31  15266 005000        CLR     R0      ;CLEAR WORK SPACE
32  15270 112102 STT$24: MOVB   (R1)+,R2        ;GET NEXT CHAR.
33  15272 042702        BIC     #177600,R2
                177600
34  15276 120227        CMPB    R2,#' ;
                000040
35  15302 001004        BNE     SGS$24  ;JUMP IF NOT BLANK
36  15304 020166        CMP     R1,12.(SP)
                000014
37  15310 002767        BLT     STT$24  ;JUMP IF MORE TO SCAN
38  15312 000454        BR      SGN$24  ;DONE
39  15314 105766 SGS$24: TSTB   7(SP);          IF OCTAL CONVERSION
                000007
40  15320 001002        BNE     SN1$24;         DO NOT PERMIT SIGNS
41  15322 005216        INC     @SP;            OCTAL - FAKE THE SIGN BIT
42  15324 000420        BR      NCK$24;         GO PROCESS THE DIGIT
43  15326 120227 SN1$24: CMPB   R2,#'+;
                000053
44  15332 001441        BEQ     FLD$24  ;JUMP IF +
```

```
45 15334 120227          CMPB    R2,#'-
         000055
46 15340 001012          BNE     NCK$24  ;JUMP IF NOT -
47 15342 005216          INC     @SP     ;SET SIGN -
48 15344 000434          BR      FLD$24
49 15346 112102 NXT$24:  MOVB    (R1)+,R2        ;GET NEXT CHAR.
50 15350 042702          BIC     #177600,R2
         177600
51 15354 120227          CMPB    R2,#' ;
         000040
52 15360 001002          BNE     NCK$24  ;JUMP IF NOT BLANK
53 15362 112702          MOVB    #60,R2  ;BLANK =ZERO
         000060
54 15366 120227 NCK$24:  CMPB    R2,#'0
         000060
55 15372 002440          BLT     ERR$24  ;JUMP IF TOO SMALL
56 15374 120266          CMPB    R2,6(SP)
         000006
57 15400 003035          BGT     ERR$24  ;JUMP IF TOO BIG
58 15402 162702          SUB     #60,R2  ;MAKE NUMERIC
         000060
59 15406 105766          TSTB    7(SP)   ;OCTAL OR BINARY
         000007
60 15412 001435          BEQ     OCL$24
61 15414 006300          ASL     R0      ;R0=BASE*R0+R2
62 15416 102426          BVS     ERR$24
63 15420 160002          SUB     R0,R2
64 15422 006300          ASL     R0
65 15424 102423          BVS     ERR$24
66 15426 006300          ASL     R0
67 15430 102421          BVS     ERR$24
68 15432 160200          SUB     R2,R0
69 15434 102417          BVS     ERR$24
70 15436 020166 FLD$24:  CMP     R1,12.(SP)
         000014
71 15442 002741          BLT     NXT$24  ;JUMP IF MORE FIELD TO SCAN
72 15444 006026 SGN$24:  ROR     (SP)+   ;TEST SIGN
73 15446 103403          BCS     DNE$24  ;JUMP IF -
74 15450 005400          NEG     R0      ;MAKE +
75 15452 102411          BVS     NGM$24  ;JUMP IF =NEGMAX
76 15454 000241          CLC     ;SET SUCCESS FLAG
77 15456 012602 DNE$24:  MOV     (SP)+,R2        ;RESTORE R2
78 15460 012601          MOV     (SP)+,R1        ;RESTORE R1
79 15462 006126          ROL     (SP)+   ;FLUSH FLAG AND SET C BIT IF ERROR
80 15464 010066          MOV     R0,4(SP)        ;RETURN RESULT
         000004
81 15470 012600          MOV     (SP)+,R0
82 15472 000207          RTS     PC
83 15474 005726 ERR$24:  TST     (SP)+   ;FLUSH SIGN
84 15476 005000 NGM$24:  CLR     R0
85 15500 005166          COM     4(SP)   ;SET ERROR FLAG
         000004
86 15504 000764          BR      DNE$24
87                       ;
88 15506 006100 OCL$24:  ROL     R0;     SHIFT 3 BITS LEFT,
89 15510 103771          BCS     ERR$24; CHECKING AS YOU GO
90 15512 006100          ROL     R0;
```

```
91 15514 103767          BCS     ERR$24;
92 15516 006100          ROL     R0;
93 15520 103765          BCS     ERR$24;
94 15522 060200          ADD     R2,R0;  ADD IN THE DIGIT
95 15524 000744          BR      FLD$24; DO NEXT
96                       .ENDC
```

```
1                               .TITLE  $ICO02
2                               .IFDF   CND$25
3                       ;
4                       ;       $ICO    V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8
9                               .GLOBL  $ICO,$OCO
10                      ;       $OCO    OCTAL TO ASCII CONVERSION
11                      ;       $ICO    INTEGER TO ASCII CONVERSION
12                      ;       CALLING SEQUENCE:
13                      ;       PUSH    FIELD START LOCATION
14                      ;       PUSH    FIELD LENGTH
15                      ;       PUSH VALUE
16                      ;       JSR     PC,$ICO (OR $OCO)
17                      ;       ERROR WILL RETURN WITH C BIT SET ON
18                      ;       R0, R1, R2, R3 ARE DESTROYED
19              000000          R0=%0
20              000001          R1=%1
21              000002          R2=%2
22              000003          R3=%3
23              000004          R4=%4
24              000006          SP=%6
25              000007          PC=%7
26  15526 012700 $OCO:  MOV     #OCT$25-REL$25,R0       ;POINT TO OCTAL TABLE
                000166
27  15532 000402        BR      GO$25
28  15534 012700 $ICO:  MOV     #DEC$25-REL$25,R0       ;POINT TO DECIMAL TABLE
                000154
29  15540 010446 GO$25: MOV     R4,-(SP)
30  15542 016603        MOV     8.(SP),R3       ;GET FIELD START
                000010
31  15546 016602        MOV     6.(SP),R2       ;GET FIELD LENGTH
                000006
32  15552 002003        BGE     LPS$25  ;JUMP IF LENGTH NOT NEG
33  15554 005002        CLR     R2
34  15556 005066        CLR     6(SP)
                000006
35  15562 016604 LPS$25: MOV    4.(SP),R4       ;GET VALUE TO BE CONVERTED
                000004
36  15566 012746        MOV     #' ,-(SP)       ;CLEAR SIGN
                000040
37  15572 020027        CMP     R0,#OCT$25-REL$25       ;CHECK IF DOING OCTAL
                000166
38  15576 001405        BEQ     POS$25  ;YES, GIVE MAGNITUDE RESULT
39  15600 005704        TST     R4
40  15602 002003        BGE     POS$25  ;JUMP IF +
41  15604 005404        NEG     R4      ;GET ABSOLUTE VALUE
42  15606 012716        MOV     #'-,@SP ;SAVE -
                000055
43  15612 005046 POS$25: CLR    -(SP)   ;SET FENCE
44  15614 060700        ADD     PC,R0
45  15616           REL$25:
46  15616 005710 TST$25: TST    @R0
47  15620 001416        BEQ     MOV$25  ;JUMP IF ALL POWERS DONE
48  15622 005001        CLR     R1
```

```
49 15624 161004 SUB$25: SUB     @R0,R4  ;SEE IF CURRENT POWER WILL GO AGAIN
50 15626 103402         BLO     BAC$25
51 15630 005201         INC     R1      ;BUMP DIGIT
52 15632 000774         BR      SUB$25
53 15634 062004 BAC$25: ADD     (R0)+,R4        ;TOO MUCH, BACK UP
54 15636 005701         TST     R1
55 15640 001002         BNE     NZE$25  ;JUMP IF DIGIT NOT 0
56 15642 005716         TST     @SP
57 15644 001764         BEQ     TST$25  ;JUMP IF NO NON-ZERO DIGITS YET
58 15646 062701 NZE$25: ADD     #60,R1  ;CONVERT TO ASCII
           000060
59 15652 010146         MOV     R1,-(SP)
60 15654 000760         BR      TST$25
61 15656 060203 MOV$25: ADD     R2,R3   ;POINT TO FIELD END
62 15660 062704         ADD     #60,R4  ;CONVERT LEAST SIGNIFICANT DIGIT
           000060
63 15664 110443         MOVB    R4,-(R3)
64 15666 005302 DCR$25: DEC     R2
65 15670 003410         BLE     FUL$25  ;JUMP IF COUNT EXHAUSTED
66 15672 112643         MOVB    (SP)+,-(R3)     ;MOVE DIGIT
67 15674 001374         BNE     DCR$25  ;JUMP IF NOT THE FENCE
68 15676 112613         MOVB    (SP)+,@R3       ;MOVE OUT THE SIGN
69 15700 005302 FIL$25: DEC     R2
70 15702 001410         BEQ     DNE$25  ;JUMP IF FIELD FILLED
71 15704 112743         MOVB    #' ,-(R3)       ;MOVE IN LEADING BLANKS
           000040
72 15710 000773         BR      FIL$25
73 15712 005726 FUL$25: TST     (SP)+
74 15714 001011         BNE     ERR$25  ;NUMBER TOO BIG FOR FIELD
75 15716 022726         CMP     #' ,(SP)+
           000040
76 15722 001011         BNE     STS$25-4.       ;JUMP IF NO ROOM FOR -
77 15724 012604 DNE$25: MOV     (SP)+,R4
78 15726 012666         MOV     (SP)+,4(SP)     ;MOVE RETURN UP
           000004
79 15732 005726         TST     (SP)+   ;FLUSH VALUE
80 15734 006126         ROL     (SP)+   ;FLUSH FLAG AND SET C BIT ON IF ERROR
81 15736 000207         RTS     PC
82 15740 005726 ERR$25: TST     (SP)+
83 15742 001376         BNE     ERR$25
84 15744 005726         TST     (SP)+   ;FLUSH SIGN
85 15746 016603         MOV     8.(SP),R3
           000010
86 15752 112723 STS$25: MOVB    #'*,(R3)+       ;FILL FIELD WITH *
           000052
87 15756 005366         DEC     6(SP)
           000006
88 15762 003373         BGT     STS$25  ;JUMP IF MORE TO DO
89 15764 005166         COM     6(SP)   ;FLAG ERROR
           000006
90 15770 000755         BR      DNE$25
91 15772 023420 DEC$25: .WORD   10000.,1000.,100.,10.,0
   15774 001750
   15776 000144
   16000 000012
   16002 000000
92 16004 100000 OCT$25: .WORD   100000,10000,1000,100,10,0
```

```
   16006 010000
   16010 001000
   16012 000100
   16014 000010
   16016 000000
93                          .ENDC
```

```
1                               .TITLE  $INT02
2                               .IFDF   CND$26
3                       ;
4                       ;       INT     V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  INT,IDINT,$RI,$POLSH
9                       ;       THE FORTRAN INT AND IDINT FUNCTIONS.
10                      ;       CALLING SEQUENCE:
11                      ;       JSR     R5,INT  (OR IDINT)
12                      ;       BR      A
13                      ;       .WORD   ARGUMENT ADDRESS
14                      ;A:
15                      ;       RETURNS INTEGER EQUIVALENT IN R0.
16                      ;       USES $RI.
17                      ;
18              000000          R0=%0
19              000004          R4=%4
20              000005          R5=%5
21              000006          SP=%6
22  16020               INT:
23  16020 016504 IDINT: MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
            000002
24  16024 016446        MOV     2(R4),-(SP)     ;PUSH LOW ORDER REAL PART
            000002
25  16030 011446        MOV     @R4,-(SP)       ;HIGH ORDER
26  16032 004467        JSR     R4,$POLSH       ;CALL $RI TO CONVERT TO
            003606
27  16036 017650'       .WORD   $RI,UPL$26      ;INTEGER
    16040 016042'
28  16042 012500 UPL$26: MOV    (SP)+,R0        ;POP INTEGER RESULT
29  16044 000205        RTS     R5
30                              .ENDC
```

```
1                                       .TITLE  $IR04
2                                       .IFDF   CND$27
3                                       .GLOBL  $IR,$ID
4                               ;       INTEGER TO REAL CONVERSION.
5                               ;
6                               ;       $IR     V004A
7                               ;
8                               ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
9                               ;       ARGUMENT IS A FULL WORD ON THE TOP OF THE STACK
10                              ;       CONVERT IT TO REAL FORMAT AND RETURN IT AS THE TOP
11                              ;       TWO WORDS ON THE STACK.
12          000000                      R0=%0
13          000001                      R1=%1
14          000002                      R2=%2
15          000003                      R3=%3
16          000004                      R4=%4
17          000006                      SP=%6
18          177304                      MQ=177304
19          177312                      NOR=177312
20          000000                      F0=%0
21                                      .IFDF   FPU
22                              $ID:    SETD;
23                                      BR      IDI$27
24                              $IR:    SETF    ;
25                              IDI$27: SETI    ;                       SHORT INTEGERS
26                                      LDCIF   (SP)+,F0;               CONVERT
27                                      STF     F0,-(SP);               RESULT TO STACK
28                                      JMP     @(R4)+
29                                      .ENDC
30                                      .IFNDF  FPU
31  16046 011646                $ID:    MOV     @SP,-(SP)               ;PUSH ARGUMENT DOWN
32  16050 011646                        MOV     @SP,-(SP)
33  16052 005066                        CLR     2(SP)   ;CLEAR LOWEST ORDER DOUBLE
          000002
34  16056 005066                        CLR     4(SP)
          000004
35  16062 005046                $IR:    CLR     -(SP)   ;MAKE ROOM FOR RESULT
36  16064 016601                        MOV     2(SP),R1                ;GET INTEGER ARGUMENT
          000002
37  16070 003002                        BGT     POS$27
38  16072 001424                        BEQ     ZER$27
39  16074 005401                        NEG     R1              ;GET ABSOLUTE VALUE
40  16076 006146                POS$27: ROL     -(SP)   ;SAVE SIGN
41                                      .IFNDF  EAE
42  16100 012702                        MOV     #220,R2 ;GET MAX. POSSIBLE EXPONENT +1
          000220
43                                      .ENDC
44                              ;       EAE CODE
45                                      .IFDF   EAE
46                                      MOV     #217,R2 ;GET MAX. POSSIBLE EXPONENT
47                                      .ENDC
48  16104 105066                        CLRB    4(SP)   ;CLEAR LOWEST ORDER FRACTION
          000004
49  16110                       NOM$27:
50                                      .IFNDF  EAE
51  16110 006101                        ROL     R1              ;LOOK FOR NORMAL BIT
52  16112 103402                        BCS     NOD$27  ;JUMP IF FOUND
```

```
53 16114 005302          DEC     R2          ;DECREASE EXPONENT
54 16116 000774          BR      NOM$27      ;TRY AGAIN
55                       .ENDC
56               ;       EAE CODE
57                       .IFDF   EAE
58                       MOV     #MQ,R3      ;POINT TO MQ
59                       CLR     @R3
60                       MOV     R1,-(R3)            ;MOVE ARG
61                       MOV     #NOR,R0     ;POINT TO NOR IN EAE
62                       CLR     @R0         ;NORMALIZE FRACTION
63                       SUB     (R0)+,R2            ;TELL EXPONENT
64                       MOV     #2,@R0      ;SHIFT OUT NORMAL BIT BY LSH
65                       MOV     @R3,R1      ;RESULT TO R1
66                       .ENDC
67 16120 110166 NOD$27: MOVB    R1,5(SP)            ;SAVE LOW ORDER FRACTION
         000005
68 16124 105001          CLRB    R1
69 16126 150201          BISB    R2,R1       ;COMBINE EXPONENT AND HIGH ORDER FRACTIO
70 16130 000301          SWAB    R1
71 16132 006026          ROR     (SP)+       ;GET SIGN
72 16134 006001          ROR     R1          ;INSERT SIGN IN RESULT
73 16136 106066          RORB    3(SP)
         000003
74 16142 010116          MOV     R1,@SP      ;OUTPUT RESULT
75 16144 000134 ZER$27: JMP     @(R4)+
76                       .ENDC
77                       .ENDC
```

```
1                                       .TITLE  $MLD05
2                                       .IFDF   CND$28
3                                       .GLOBL  $MLD,$ERRA
4                               ;       $MLD     THE DOUBLE MULTIPLY ROUTINE
5                               ;
6                               ;       $MLD    V005A
7                               ;
8                               ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
9                               ;       CALLED IN POLISH MODE.
10                              ;       REPLACES THE TOP TWO DOUBLES ON THE STACK
11                              ;       WITH THEIR PRODUCT.
12              000000                  R0=%0
13              000001                  R1=%1
14              000002                  R2=%2
15              000003                  R3=%3
16              000004                  R4=%4
17              000005                  R5=%5
18              000006                  SP=%6
19              000007                  PC=%7
20              177304                  MQ=177304
21              000010                  A=8.
22              000020                  B=16.
23              000014                  RESLT=12.
24              000002                  SIGN=2
25              000000                  F0=%0
26                                      .IFDF   FPU
27                              $MLD:   .WORD   170011  ;;SETD
28                                      .WORD   172428  ;;LDD    (SP)+,F0        ;GET OPERAND
29                                      .WORD   171026  ;;MULD   (SP)+,F0        ;PRODUCT
30                                      .WORD   174046  ;;STD    F0,-(SP)        ;PRODUCT TO STAC
31                                      JMP     @(R4)+
32                                      .ENDC
33                                      .IFNDF  FPU
34  16146 010446                $MLD:   MOV     R4,-(SP)
35  16150 010546                        MOV     R5,-(SP)
36  16152 006366                        ASL     A+0-4(SP)       ;SHIFT MULTIPLICAND
            000004
37  16156 006146                        ROL     -(SP)   ;KEEP SIGN
38  16160 005046                        CLR     -(SP)   ;CLEAR EXPONENT
39  16162 116615                        MOVB    A+1(SP),@SP     ;KEEP MULTIPLICAND EXPONENT
            000011
40  16166 001436                        BEQ     ZER$28  ;JUMP IF ANSWER IS ZERO
41  16170 116666                        MOVB    A(SP),A+1(SP)   ;SHIFT FRACTION LEFT
            000010
            000011
42  16176 000261                        SEC             ;INSERT NORMAL BIT
43  16200 006066                        ROR     A(SP)
            000010
44  16204 116666                        MOVB    A+3(SP),A(SP)
            000013
            000010
45  16212 000366                        SWAB    A+2(SP)
            000012
46  16216 116666                        MOVB    A+5(SP),A+2(SP)
            000015
            000012
47  16224 000366                        SWAB    A+4(SP)
```

```
                000014
48  16230 116566        MOVB    A+7(SP),A+4(SP)
                000017
                000014
49  16236 000366        SWAB    A+6(SP)
                000016
50  16242 105066        CLRB    A+6(SP) ;MAKE ROOM FOR EXTRA BITS
                000016
51  16246 006366        ASL     B(SP)   ;SHIFT HIGH MULTIPLIER
                000020
52  16252 005566        ADC     SIGN(SP)        ;GET PRODUCT SIGN
                000002
53  16256 105766        TSTB    B+1(SP)
                000021
54  16262 001003        BNE     NNZ$28  ;JUMP IF NOT ZERO
55  16264 022626 ZER$28: CMP    (SP)+,(SP)+     ;FLUSH SIGN AND EXPONENT
56  16266 000167 ZE1$28: JMP    ZE2$28
                000334
57  16272 005000 NNZ$28: CLR    R0      ;CLEAR PRODUCT
58  16274 005001        CLR     R1
59                      .IFNDF  EAE&MULDIV
60  16276 005002        CLR     R2
61  16300 005003        CLR     R3
62  16302 005005        CLR     R5      ;CLEAR C BIT OVERFLOW CATCHER
63  16304 006066        ROR     B(SP)   ;SIGN IS NOW 0
                000020
64  16310 012746        MOV     #16.,-(SP)      ;PUSH ITERATION COUNT
                000020
65  16314 016604        MOV     B+6+2(SP),R4    ;GET LOWEST ORDER MULTIPLER
                000030
66  16320 001404        BEQ     B6Z$28  ;JUMP IF NO BITS HERE
67  16322 004767        JSR     PC,MT0$28
                000410
68  16326 012716        MOV     #16.,@SP        ;RESTORE COUNT
                000020
69  16332 016604 B6Z$28: MOV    B+4+2(SP),R4    ;GET NEXT LOWEST FRACTION
                000026
70  16336 001003        BNE     B4N$28  ;JUMP IF WORK TO DO
71  16340 005766        TST     B+6+2(SP)
                000030
72  16344 001406        BEQ     B4Z$28  ;JUMP IF NO PRODUCT YET
73  16346 004767 B4N$28: JSR    PC,MT2$28
                000360
74  16352 004767        JSR     PC,MLT$28       ;ONE BIT FULL PRECISION
                000262
75  16356 012716        MOV     #16.,@SP
                000020
76  16362 016604 B4Z$28: MOV    B+2+2(SP),R4    ;GET NEXT TO HIGHEST ORDER FRACT
                000024
77  16366 001006        BNE     B2N$28
78  16370 005766        TST     B+4+2(SP)
                000026
79  16374 001003        BNE     B2N$28
80  16376 005766        TST     B+6+2(SP)
                000030
81  16402 001402        BEQ     B2Z$28
82  16404 004767 B2N$28: JSR    PC,MLT$28
```

117

```
                000230
83  16410 016604 B2Z$28: MOV     B+0+2(SP),R4    ;GET HIGH ORDER BITS
                000022
84  16414 012716         MOV     #7,@SP  ;THERE ARE ONLY SEVEN OF THEM
                000007
85  16420 004767         JSR     PC,MLT$28
                000214
86  16424 004767         JSR     PC,MT1$28       ;GO DO THE NORMAL BIT
                000214
87  16430 005726         TST     (SP)+   ;FLUSH ITERATION COUNT
88  16432 062604         ADD     (SP)+,R4        ;ADD EXPONENTS
89                       .ENDC
90                       .IFDF   EAE!MULDIV
91                       CLR     R4
92                       BISB    B+1(SP),R4      ;GET EXPONENT
93                       ADD     R4,@SP  ;GET SUM OF EXPONENTS
94                       MOVB    #1,B+1(SP)      ;INSERT NORMAL BIT
95                       ROR     B(SP)
96                       SWAB    B(SP)   ;LEFT JUSTIFY FRACTION
97                       MOVB    B+3(SP),B(SP)
98                       SWAB    B+2(SP)
99                       MOVB    B+5(SP),B+2(SP)
100                      SWAB    B+4(SP)
101                      MOVB    B+7(SP),B+4(SP)
102                      SWAB    B+6(SP)
103                      CLRB    B+6(SP)
104                      .ENDC
105                      .IFDF   EAE
106                      MOV     #MQ,R4  ;POINT TO MQ
107                      MOV     A(SP),-(SP)
108                      MOV     B+6+2(SP),@R4   ;GET A1*B4
109                      JSR     R5,EMU$28
110                      MOV     (SP)+,R2        ;RESULT TO PRODUCT
111                      MOV     (SP)+,R3
112                      MOV     A+2(SP),-(SP)
113                      MOV     B+4+2(SP),@R4   ;GET A2*B3
114                      JSR     R5,EMU$28
115                      ADD     (SP)+,R2        ;ADD TO PRODUCT
116                      ADC     R1
117                      ADD     (SP)+,R3
118                      ADC     R2
119                      ADC     R1
120                      MOV     A+4(SP),-(SP)
121                      MOV     B+2+2(SP),@R4   ;GET A3*B2
122                      JSR     R5,EMU$28
123                      ADD     (SP)+,R2
124                      ADC     R1
125                      ADD     (SP)+,R3
126                      ADC     R2
127                      ADC     R1
128                      MOV     A+6(SP),-(SP)
129                      MOV     B+0+2(SP),@R4   ;GET A4*B1
130                      JSR     R5,EMU$28
131                      ADD     (SP)+,R2
132                      ADC     R1
133                      ADD     (SP)+,R3
134                      ADC     R2
```

```
135                     ADC     R1
136                     MOV     R2,R3   ;DIVIDE BY 2**16
137                     MOV     R1,R2
138                     CLR     R1
139                     MOV     A(SP),-(SP)
140                     MOV     B+4+2(SP),@R4   ;GET A1*B3
141                     JSR     R5,EMU$28
142                     ADD     (SP)+,R2
143                     ADC     R1
144                     ADD     (SP)+,R3
145                     ADC     R2
146                     ADC     R1
147                     MOV     A+2(SP),-(SP)
148                     MOV     B+2+2(SP),@R4   ;GET A2*B2
149                     JSR     R5,EMU$28
150                     ADD     (SP)+,R2
151                     ADC     R1
152                     ADD     (SP)+,R3
153                     ADC     R2
154                     ADC     R1
155                     MOV     A+4(SP),-(SP)
156                     MOV     B+0+2(SP),@R4   ;GET A3*B1
157                     JSR     R5,EMU$28
158                     ADD     (SP)+,R2
159                     ADC     R1
160                     ADD     (SP)+,R3
161                     ADC     R2
162                     ADC     R1
163                     MOV     A(SP),-(SP)
164                     MOV     B+2+2(SP),@R4   ;GET A1*B2
165                     JSR     R5,EMU$28
166                     ADD     (SP)+,R1
167                     ADC     R0
168                     ADD     (SP)+,R2
169                     ADC     R1
170                     ADC     R0
171                     MOV     A+2(SP),-(SP)
172                     MOV     B+0+2(SP),@R4   ;GET A2*B1
173                     JSR     R5,EMU$28
174                     ADD     (SP)+,R1
175                     ADC     R0
176                     ADD     (SP)+,R2
177                     ADC     R1
178                     ADC     R0
179                     MOV     A(SP),-(SP)
180                     MOV     B+0+2(SP),@R4   ;GET A1*B1
181                     JSR     R5,EMU$28
182                     ADD     (SP)+,R0
183                     ADD     (SP)+,R1
184                     ADC     R0
185                     MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
186                     .ENDC
187                     .IFDF   MULDIV
188                     MOV     A(SP),-(SP)
189                     MOV     B+6+2(SP),R4    ;GET A1*B4
190                     JSR     PC,EMU$28
191                     MOV     R4,R2   ;RESULT TO PRODUCT
```

```
192                     MOV     R5,R3
193                     MOV     A+2(SP),-(SP)
194                     MOV     B+4+2(SP),R4    ;GET A2*B3
195                     JSR     PC,EMU$28
196                     ADD     R4,R2   ;ADD TO PRODUCT
197                     ADC     R1
198                     ADD     R5,R3
199                     ADC     R2
200                     ADC     R1
201                     MOV     A+4(SP),-(SP)
202                     MOV     B+2+2(SP),R4    ;GET A3*B2
203                     JSR     PC,EMU$28
204                     ADD     R4,R2
205                     ADC     R1
206                     ADD     R5,R3
207                     ADC     R2
208                     ADC     R1
209                     MOV     A+6(SP),-(SP)
210                     MOV     B+0+2(SP),R4    ;GET A4*B1
211                     JSR     PC,EMU$28
212                     ADD     R4,R2
213                     ADC     R1
214                     ADD     R5,R3
215                     ADC     R2
216                     ADC     R1
217                     MOV     R2,R3   ;DIVIDE BY 2**16
218                     MOV     R1,R2
219                     CLR     R1
220                     MOV     A(SP),-(SP)
221                     MOV     B+4+2(SP),R4    ;GET A1*B3
222                     JSR     PC,EMU$28
223                     ADD     R4,R2
224                     ADC     R1
225                     ADD     R5,R3
226                     ADC     R2
227                     ADC     R1
228                     MOV     A+2(SP),-(SP)
229                     MOV     B+2+2(SP),R4    ;GET A2*B2
230                     JSR     PC,EMU$28
231                     ADD     R4,R2
232                     ADC     R1
233                     ADD     R5,R3
234                     ADC     R2
235                     ADC     R1
236                     MOV     A+4(SP),-(SP)
237                     MOV     B+0+2(SP),R4    ;GET A3*B1
238                     JSR     PC,EMU$28
239                     ADD     R4,R2
240                     ADC     R1
241                     ADD     R5,R3
242                     ADC     R2
243                     ADC     R1
244                     MOV     A(SP),-(SP)
245                     MOV     B+2+2(SP),R4    ;GET A1*B2
246                     JSR     PC,EMU$28
247                     ADD     R4,R1
248                     ADC     R0
```

120

```
                        ADD     R5,R2
                        ADC     R1
                        ADC     R0
                        MOV     A+2(SP),-(SP)
                        MOV     B+0+2(SP),R4    ;GET A2*B1
                        JSR     PC,EMU$28
                        ADD     R4,R1
                        ADC     R0
                        ADD     R5,R2
                        ADC     R1
                        ADC     R0
                        MOV     A(SP),-(SP)
                        MOV     B+0+2(SP),R4    ;GET A1*B1
                        JSR     PC,EMU$28
                        ADD     R4,R0
                        ADD     R5,R1
                        ADC     R0
                        MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
                        .ENDC
6434 006303             ASL     R3      ;SHIFT OUT NORMAL BIT
6436 006102             ROL     R2
6440 006101             ROL     R1
6442 006100             ROL     R0
6444 103405             BCS     NOM$28  ;JUMP IF IT WAS FOUND
6446 006303             ASL     R3
6450 006102             ROL     R2
6452 006101             ROL     R1
6454 006100             ROL     R0      ;MUST HAVE GOT IT NOW
6456 005304             DEC     R4      ;ADJUST EXPONENT
6460 162704 NOM$28: SUB     #200,R4 ;TAKE OUT ONE OF THE EXCESS 128'S
     000200
6464 003453             BLE     UND$28  ;JUMP IF UNDERFLOW
6466 022704             CMP     #377,R4
     000377
6472 002445             BLT     OVR$28  ;JUMP IF OVERFLOW
6474 105003             CLRB    R3
6476 150203             BISB    R2,R3   ;SHIFT FRACTION RIGHT
6500 000303             SWAB    R3
6502 105002             CLRB    R2
6504 150102             BISB    R1,R2
6506 000302             SWAB    R2
6510 105001             CLRB    R1
6512 150001             BISB    R0,R1
6514 000301             SWAB    R1
6516 105000             CLRB    R0
6520 150400             BISB    R4,R0
6522 000300             SWAB    R0
6524 006026             ROR     (SP)+   ;GET PRODUCT SIGN
6526 006000             ROR     R0      ;INSERT IT IN RESULT
6530 006001             ROR     R1
6532 006002             ROR     R2
6534 006003             ROR     R3
6536 005503             ADC     R3      ;ROUND RESULT
6540 005502             ADC     R2
6542 005501             ADC     R1
6544 005500             ADC     R0
6546 103416             BCS     OV1$28  ;JUMP IF OVERFLOW ON ROUND
```

```
304 6550 102415          BVS     OV1$28
305 6552 010066 OUT$28:  MOV     R0,RESLT(SP)     ;PUT OUT ANSWER
         000014
306 6556 010166          MOV     R1,RESLT+2(SP)
         000016
307 6562 010266          MOV     R2,RESLT+4(SP)
         000020
308 6566 010366          MOV     R3,RESLT+6(SP)
         000022
309 6572 012605          MOV     (SP)+,R5
310 6574 012604          MOV     (SP)+,R4
311 6576 062706          ADD     #8.,SP   ;FLUSH TOP ARGUMENT
         000010
312 6602 000134          JMP     @(R4)+   ;RETURN
313 6604 005746 OV1$28:  TST     -(SP)    ;FAKE SIGN
314 6606 012700 OVR$28:  MOV     #5003,R0         ;ERROR 3,10
         005003
315 6612 000402          BR      ECL$28
316 6614 012700 UND$28:  MOV     #3005,R0         ;ERROR 5,6
         003005
317 6620 005726 ECL$28:  TST     (SP)+    ;FLUSH SIGN
318 6622 004567          JSR     R5,$ERRA         ;CALL ERROR
         003170
319 6626 005000 ZE2$28:  CLR     R0       ;CLEAR HIGH ORDER RESULT
320 6630 005001          CLR     R1       ;CLEAR LOW ORDER
321 6632 005002          CLR     R2
322 6634 005003          CLR     R3
323 6636 000745          BR      OUT$28
324                      .IFNDF  EAE&MULDIV
325 6640 006204 MLT$28:  ASR     R4       ;TEST NEXT MULTIPLIER BIT
326 6642 103022          BCC     X0$28    ;JUMP IF IT IS 0
327 6644 066603 MT1$28:  ADD     A+6+4(SP),R3     ;ADD IN MULTIPLICAND
         000022
328 6650 005502          ADC     R2
329 6652 005501          ADC     R1
330 6654 005500          ADC     R0
331 6656 005505          ADC     R5       ;SAVE OVERFLOW
332 6660 066602          ADD     A+4+4(SP),R2
         000020
333 6664 005501          ADC     R1
334 6666 005500          ADC     R0
335 6670 005505          ADC     R5
336 6672 066601          ADD     A+2+4(SP),R1
         000016
337 6676 005500          ADC     R0
338 6700 005505          ADC     R5
339 6702 066600          ADD     A+0+4(SP),R0
         000014
340 6706 005505          ADC     R5
341 6710 006205 X0$28:   ASR     R5       ;RECOVER OVERFLOW IF ANY
342 6712 006000          ROR     R0       ;NOW SHIFT PRODUCT
343 6714 006001          ROR     R1
344 6716 006002          ROR     R2
345 6720 006003          ROR     R3
346 6722 005366          DEC     2(SP)    ;COUNT LOOP
         000002
347 6726 003344          BGT     MLT$28   ;AGAIN PLEASE
```

```
348 6730 000207          RTS     PC      ;RETURN TO CALLER
349 6732 005366 MT2$28:  DEC     2(SP)   ;DO ONLY 15 BITS THIS PASS
         000002
350 6736 006204 MT0$28:  ASR     R4      ;TEST NEXT MULTIPLIER BIT
351 6740 103007          BCC     X00$28  ;JUMP IF 0
352 6742 066601          ADD     A+2+4(SP),R1    ;USE ONLY HIGH ORDER MULTIPLICAN
         000016
353 6746 005500          ADC     R0
354 6750 005505          ADC     R5
355 6752 066600          ADD     A+0+4(SP),R0
         000014
356 6756 005505          ADC     R5
357 6760 006205 X00$28:  ASR     R5      ;RECOVER ANY OVERFLOW
358 6762 006000          ROR     R0
359 6764 006001          ROR     R1
360 6766 006002          ROR     R2
361 6770 006003          ROR     R3
362 6772 005366          DEC     2(SP)   ;COUNT LOOP
         000002
363 6776 003357          BGT     MT0$28
364 7000 000207          RTS     PC      ;RETURN TO CALLER
365                      .ENDC
366                      .IFDF   EAE
367              EMU$28: CLR     @SP     ;CLEAR PRODUCT
368                      TST     @R4
369                      BEQ     MZ$28   ;JUMP IF MULTIPLIER 0
370                      BGT     MPL$28
371                      TST     2(SP)   ;TEST MULTIPLICAND
372                      BEQ     MZ$28   ;JUMP IF 0
373                      BGT     MNG$28  ;JUMP IF +*-
374                      ADD     (R4)+,@SP       ;CORRECT 2'S COMPLEMENT
375                      ADD     2(SP),@SP
376                      BR      EML$28
377              MPL$28: TST     2(SP)   ;TEST MULTIPLICAND
378                      BEQ     MZ$28   ;JUMP IF 0
379                      BGT     MLQ$28  ;JUMP IF +
380                      ADD     (R4)+,@SP
381                      BR      EML$28
382              MNG$28: ADD     2(SP),@SP
383              MLQ$28: TST     (R4)+   ;POINT TO MUL
384              EML$28: MOV     2(SP),@R4       ;MULTIPLY
385                      MOV     -(R4),2(SP)     ;GET PRODUCT
386                      ADD     -(R4),@SP
387                      TST     (R4)+   ;POINT TO MQ
388                      JMP     @R5     ;RETURN
389              MZ$28:  CLR     2(SP)   ;RETURN 0
390                      JMP     @R5
391                      .ENDC
392                      .IFDF   MULDIV
393              EMU$28: CLR     -(SP)   ;CLEAR HIGH PRODUCT
394                      TST     R4      ;TEST MULTIPLICAND
395                      BEQ     MZ$28   ;JUMP IF 0
396                      BGT     MPL$28  ;+
397                      TST     4(SP)   ;TEST MULTIPLIER
398                      BEQ     MZ$28   ;JUMP IF 0
399                      BGT     MN1$28  ;+
400                      BR      MNG$28
```

123

```
401             MPL$28: TST     4(SP)   ;TEST MULTIPLIER
402                     BEQ     MZ$28   ;JUMP IF 0
403                     BGT     MLQ$28  ;+
404                     ADD     R4,@SP
405                     BR      MLQ$28
406             MNG$28: ADD     R4,@SP
407             MN1$28: ADD     4(SP),@SP
408             MLQ$28: .WORD   070466,4        ;;MUL   4(SP),R4        ;GET PRO
409             MDN$28: ADD     (SP)+,R4        ;ADD IN HIGH ORDER PARTS
410                     MOV     (SP)+,@SP       ;FLUSH MULTIPLIER
411                     RTS     PC      ;RETURN
412             MZ$28:  CLR     R4      ;RESULT IS 0
413                     CLR     R5
414                     BR      MDN$28
415                     .ENDC
416                     .ENDC
417                     .ENDC
```

```
1                                       .TITLE  $MLI05
2                                       .IFDF   CND$29
3                                       .GLOBL  $MLI,$ERR
4                       ;               $MLI ---- INTEGER MULTIPLY
5                       ;
6                       ;               $MLI    V005A
7                       ;
8                       ;               COPYRIGHT 1971, DIGITAL EQUIPMET CORP., MAYNARD, MASS.
9                       ;               CALLED IN THE POLISH MODE
10                      ;               REPLACE THE TWO INTEGERS ON THE TOP OF THE STACK
11                      ;               WITH THEIR PRODUCT
12              000000                  R0=%0
13              000001                  R1=%1
14              000002                  R2=%2
15              000003                  R3=%3
16              000004                  R4=%4
17              000005                  R5=%5
18              000006                  SP=%6
19              177311                  SR$29=177311
20              177304                  MQ=177304
21                                      .IFNDF  EAE&MULDIV
22  17002  005000  $MLI:                CLR     R0              ;CLEAR PRODUCT SIGN
23  17004  012601                      MOV     (SP)+,R1                ;GET MULTIPLICAND
24  17006  003003                      BGT     P1$29   ;JUMP IF +
25  17010  001455                      BEQ     ZER$29  ;JUMP IF ANSWER IS ZERO
26  17012  005200                      INC     R0              ;NOTE -
27  17014  005401                      NEG     R1
28  17016  011603  P1$29:              MOV     @SP,R3  ;GET MULTIPLIER
29  17020  003003                      BGT     P2$29
30  17022  001450                      BEQ     ZER$29
31  17024  005200                      INC     R0              ;FORM RESULT SIGN
32  17026  005403                      NEG     R3
33  17030  010446  P2$29:              MOV     R4,-(SP)                ;SAVE R4
34  17032  012704                      MOV     #8.,R4  ;SET UP FOR LOW EIGHT BITS
             000010
35  17036  020103                      CMP     R1,R3
36  17040  002003                      BGE     CLR$29  ;JUMP IF MULTIPLIER SMALLER
37  17042  010102                      MOV     R1,R2   ;IF NOT MAKE IT SO
38  17044  010301                      MOV     R3,R1
39  17046  010203                      MOV     R2,R3
40  17050  005002  CLR$29:             CLR     R2              ;CLEAR HIGH ORDER PRODUCT
41  17052  006002  MUL$29:             ROR     R2              ;SHIFT PRODUCT
42  17054  006003                      ROR     R3
43  17056  103001                      BCC     CYC$29  ;JUMP IF MULTIPLIER BIT IS 0
44  17060  060102                      ADD     R1,R2   ;ADD IN MULTIPLICAND
45  17062  005304  CYC$29:             DEC     R4              ;COUNT LOOP
46  17064  003372                      BGT     MUL$29
47  17066  012604                      MOV     (SP)+,R4                ;RESTORE R4
48  17070  105703                      TSTB    R3              ;TEST HIGH MULTIPLIER
49  17072  001026                      BNE     OVR$29  ;JUMP IF MULTIPLIER NOT GONE
50  17074  150203                      BISB    R2,R3   ;MOVE PRODUCT RIGHT
51  17076  000303                      SWAB    R3
52  17100  105002                      CLRB    R2
53  17102  000302                      SWAB    R2
54  17104  006202                      ASR     R2              ;ONE LAST SHIFT
55  17106  001020                      BNE     OVR$29  ;JUMP IF PRODUCT EXCEEDS 15 BITS
56  17110  006003                      ROR     R3
```

```
57 17112 005403          NEG     R3      ;MAKE -
58 17114 100015          BPL     OVR$29  ;JUMP IF TOO BIG
59 17116 006000          ROR     R0      ;GET PRODUCT SIGN
60 17120 103402          BCS     OUT$29  ;JUMP IF -
61 17122 005403          NEG     R3      ;MAKE +
62 17124 102411          BVS     OVR$29
63 17126 010316 OUT$29:  MOV     R3,@SP  ;MOVE OUT RESULT
64 17130 000134          JMP     @(R4)+  ;RETURN
65 17132 005403 NGM$29:  NEG     R3      ;TEST FOR OCTAL 100000
66 17134 102005          BVC     OVR$29  ;JUMP IF NOT
67 17136 006000          ROR     R0      ;TEST FOR NEGATIVE RESULT
68 17140 103772          BCS     OUT$29  ;YES, WE CAN HANDLE THIS
69 17142 000402          BR      OVR$29  ;OVERFLOW
70 17144 005016 ZER$29:  CLR     @SP     ;CLEAR PRODUCT
71 17146 000134          JMP     @(R4)+  ;RETURN
72                       .ENDC
73                ;      $MLI CODE FOR THE EAE
74                       .IFDF   EAE
75                $MLI:  MOV     #MQ,R0  ;GET MQ ADDRESS
76                       MOV     (SP)+,(R0)+     ;MULTIPLIER TO MQ
77                       MOV     (SP)+,@R0       ;MULTIPLICAND TO MUL
78                       MOV     -(R0),-(SP)     ;PRODUCT TO STACK
79                       BITB    #2,SR$29
80                       BEQ     OVR$29  ;JUMP IF PRODUCT NOT SINGLE PRECISION
81                       JMP     @(R4)+  ;RETURN TO USER
82                       .ENDC
83                ;      $MLI FOR THE MULDIV
84                       .IFDF   MULDIV
85                $MLI:  MOV     (SP)+,R0        ;MOVE MULTIPLIER
86                       .WORD   070026  ;;MUL   (SP)+,R0        ;MULTIPLY
87                       MOV     R1,-(SP)        ;PUSH PRODUCT
88                       BCS     OVR$29  ;JUMP IF OVERFLOW
89                       JMP     @(R4)+
90                       .ENDC
91 17150 005016 OVR$29:  CLR     (SP)    ;RETURN 0
92 17152 004567          JSR     R5,$ERR ;ERROR 3,14
         002630
93 17156 000134          JMP     @(R4)+
94 17160     003         .BYTE   3
95 17161     016         .BYTE   14.
96                       .ENDC
```

126

```
1                               .TITLE  $MLR05
2                               .IFDF   CND$$0
3                               .GLOBL  $MLR,$ERRA
4                       ;       $MLR    THE REAL MULTIPLY ROUTINE
5                       ;
6                       ;
7                       ;       $MLR    V005A
8                       ;
9                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
10                      ;
11                      ;       CALLED IN POLISH MODE.
12                      ;       REPLACES THE TOP TWO REALS ON THE STACK
13                      ;       WITH THEIR PRODUCT.
14              000000          R0=%0
15              000001          R1=%1
16              000002          R2=%2
17              000003          R3=%3
18              000004          R4=%4
19              000005          R5=%5
20              000006          SP=%6
21              000007          PC=%7
22              177304          MQ=177304
23              177311          SR=177311
24              177314          LSH=177314
25              000000          F0=%0
26              000010  A=8.
27              000014  B=12.
28              000010  RESLT=8.
29              000002  SIGN=2
30                              .IFDF   FPU
31                      $MLR:   .WORD   170001  ;;SETF
32                              .WORD   172426  ;;LDF    (SP)+,F0        ;GET MULTIPLICAN
33                              .WORD   171026  ;;MULF   (SP)+,F0        ;MULTIPLY
34                              .WORD   174046  ;;STF    F0,-(SP)        ;PRODUCT TO STAC
35                              JMP     @(R4)+
36                              .ENDC
37                              .IFNDF  FPU
38  17162 010446        $MLR:   MOV     R4,-(SP)
39  17164 010546                MOV     R5,-(SP)
40                              .IFNDF  EAE&MULDIV
41  17166 016602                MOV     A+0-4(SP),R2
          000004
42  17172 006302                ASL     R2              ;SHIFT MULTIPLICAND
43  17174 006146                ROL     -(SP)           ;KEEP SIGN
44  17176 005046                CLR     -(SP)           ;CLEAR EXPONENT
45  17200 000302                SWAB    R2
46  17202 110216                MOVB    R2,@SP          ;KEEP MULTIPLICAND EXPONENT
47  17204 001507                BEQ     ZE1$$0          ;JUMP IF ANSWER IS ZERO
48  17206 000261                SEC                     ;INSERT NORMAL BIT
49  17210 006002                ROR     R2
50  17212 105002                CLRB    R2
51  17214 156602                BISB    A+3(SP),R2
          000013
52  17220 005003                CLR     R3
53  17222 156603                BISB    A+2(SP),R3
          000012
54  17226 000303                SWAB    R3
```

```
55 17230 006366          ASL     B(SP)   ;SHIFT HIGH MULTIPLIER
         000014
56 17234 005566          ADC     SIGN(SP)        ;GET PRODUCT SIGN
         000002
57 17240 105766          TSTB    B+1(SP)
         000015
58 17244 001467          BEQ     ZE1$30  ;JUMP IF ZERO
59 17246 006066          ROR     B(SP)   ;SIGN IS NOW ZERO
         000014
60 17252 005000          CLR     R0      ;CLEAR PRODUCT
61 17254 005001          CLR     R1
62 17256 016604          MOV     B+2(SP),R4      ;GET LOW ORDER MULTIPLIER
         000016
63 17262 001406          BEQ     B2Z$30
64 17264 012705 B2N$30:  MOV     #15.,R5
         000017
65 17270 004767          JSR     PC,MT0$30
         000220
66 17274 004767          JSR     PC,MLT$30       ;DO LAST LOW BIT FULL PRECISION
         000160
67 17300 016604 B2Z$30:  MOV     B(SP),R4        ;GET HIGH ORDER BITS
         000014
68 17304 012705          MOV     #7,R5   ;THERE ARE ONLY SEVEN OF THEM
         000007
69 17310 004767          JSR     PC,MLT$30
         000144
70 17314 004767          JSR     PC,MT1$30       ;GO DO THE NORMAL BIT
         000144
71 17320 062604          ADD     (SP)+,R4        ;ADD EXPONENTS
72                       .ENOC
73                ;      EAE CODE
74                       .IFDF   EAE
75                ;      (A1+A2*2**-16)*(B1+B2*2**-16)
76                       MOV     #MQ,R4  ;POINT TO MQ
77                       MOV     #100000,R5      ;GET LEADING BIT
78                       MOV     B+2-4(SP),@R4   ;LOW ORDER B TO MQ
79                       MOV     B+0-4(SP),-(R4) ;HIGH TO AC
80                       BEQ     ZER$30  ;JUMP IF 0
81                       INC     @#LSH   ;GET SIGN
82                       RORB    @#SR
83                       ROL     -(SP)   ;SAVE IT
84                       MOV     (R4)+,-(SP)     ;SAVE EXPONENT
85                       CLRB    @SP     ;RIGHT JUSTIFY IT
86                       SWAB    @SP
87                       MOV     #7,@#LSH        ;MOVE FRACTION LEFT
88                       MOV     @R4,-(SP)       ;SAVE B2
89                       BIS     R5,-(R4)        ;INSERT NORMAL BIT
90                       MOV     (R4)+,-(SP)     ;SAVE B1
91                       MOV     A+2+4(SP),@R4   ;LOW ORDER A TO MQ
92                       MOV     A+0+4(SP),-(R4) ;HIGH TO AC
93                       BEQ     ZE2$30  ;JUMP IF 0
94                       INC     @#LSH   ;GET SIGN
95                       RORB    @#SR
96                       ADC     6(SP)   ;GET RESULT SIGN
97                       MOV     @R4,R3  ;GET EXPONENT
98                       CLRB    R3
99                       SWAB    R3
```

```
100                     ADD     R3,4(SP)        ;GET SUM OF EXPONENTS
101                     MOV     #7,@#LSH        ;LEFT JUSTIFY FRACTION
102                     MOV     (R4)+,R2        ;SAVE A1
103                     BIS     R5,R2   ;INSERT NORMAL BIT
104                     CLR     R0      ;CLEAR PRODUCT
105                     CLR     R1
106                     MOV     (R4)+,R3        ;SAVE A2
107                     BNE     A2N$30
108                     TST     -(R4)   ;POINT TO MQ
109                     BR      A2Z$30  ;SHORT CUT
110             A2N$30: MOV     @SP,@R4 ;GET B1*A2
111                     CMP     -(R4),-(R4)     ;POINT TO AC
112                     ADD     R3,@R4  ;A2, 2'S COMP CORRECTION
113                     TST     R3
114                     BPL     A2P$30
115                     ADD     @SP,@R4 ;B1, CORRECTION
116             A2P$30: MOV     (R4)+,R1        ;HIGH PRODUCT TO R1
117             A2Z$30: MOV     2(SP),(R4)+     ;B2 TO MQ
118                     BNE     B2N$30
119                     TST     -(R4)   ;POINT TO MQ
120                     BR      B2Z$30  ;SHORT CUT
121             B2N$30: MOV     R2,@R4  ;GET B2*A1
122                     CMP     -(R4),-(R4)     ;POINT TO AC
123                     ADD     2(SP),@R4       ;B2, CORRECTION
124                     TST     2(SP)
125                     BPL     B2P$30  ;JUMP IF B2 +
126                     ADD     R2,@R4  ;A1, CORRECTION
127             B2P$30: ADD     (R4)+,R1        ;HIGH PRODUCT TO R1
128                     ADC     R0
129             B2Z$30: MOV     R2,(R4)+        ;A1 TO MQ
130                     ADD     R2,R0
131                     MOV     @SP,@R4 ;GET A1*B1
132                     ADD     (SP)+,R0
133                     ADD     -(R4),R1
134                     ADC     R0
135                     ADD     -(R4),R0        ;AC+R0
136                     TST     (SP)+   ;POP B2
137                     MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
138                     .ENDC
139             ;       MUL/DIV CODE
140                     .IFDF   MULDIV
141             ;       (A1+A2*2**-16)*(B1+B2*2**-16)
142                     MOV     B+2-4(SP),R5    ;LOW ORDER B
143                     MOV     B+0-4(SP),R4    ;HIGH ORDER
144                     BEQ     ZER$30
145                     .WORD   073427,1        ;;      ASHC    #1,R4   ;GET SIG
146                     ROL     -(SP)   ;SAVE IT
147                     MOV     R4,-(SP)        ;SAVE EXPONENT
148                     CLRB    @SP
149                     SWAB    @SP     ;RIGHT JUSTIFY
150                     .WORD   073427,7        ;;      ASHC    #7,R4   ;LEFT JU
151                     MOV     R5,-(SP)        ;SAVE B2
152                     BIS     #100000,R4      ;INSERT NORMAL BIT
153                     MOV     R4,-(SP)        ;SAVE B1
154                     MOV     A+2+4(SP),R3    ;GET A2
155                     MOV     A+0+4(SP),R2    ;GET A1
156                     BEQ     ZE2$30  ;JUMP IF RESULT TO BE 0
```

```
                        .WORD   073227,1        ;;      ASHC    #1,R2   ;GET SIG
                        ADC     6(SP)   ;GET RESULT SIGN
                        MOV     R2,R0   ;GET EXPONENT
                        CLRB    R0
                        SWAB    R0
                        ADD     R0,4(SP)        ;GET SUM OF EXPONENTS
                        .WORD   073227,7        ;;      ASHC    #7,R2   ;GET A1
                        BIS     #100000,R2      ;INSERT NORMAL BIT
                        CLR     R0      ;CLEAR ACCUMULATOR
                        CLR     R1
                        TST     R3      ;CHECK A2
                        BEQ     A2Z$30  ;JUMP IF 0
                        .WORD   070403  ;;      MUL     R3,R4   ;GET A2*B1
                        ADD     R3,R4
                        TST     R3
                        BPL     A2P$30  ;JUMP IF A2 +
                        ADD     @SP,R4  ;B1 CORRECTION
A2P$30:                 MOV     R4,R1   ;A2*B1*2**-16
A2Z$30:                 MOV     2(SP),R4        ;B2 TO MULTIPLIER
                        BEQ     B2Z$30  ;JUMP IF 0
                        .WORD   070402  ;;      MUL     R2,R4   ;GET A1*B2
                        ADD     2(SP),R4
                        TST     2(SP)
                        BPL     B2P$30  ;JUMP IF B2 +
                        ADD     R2,R4   ;A1 CORRECTION
B2P$30:                 ADD     R4,R1   ;A1*B2*2**-16
                        ADC     R0
B2Z$30:                 MOV     R2,R4   ;A1 TO MULTIPLIER
                        ADD     R2,R0
                        .WORD   070416  ;;      MUL     @SP,R4  ;GET A1*B1
                        ADD     (SP)+,R0
                        ADD     R5,R1   ;LOW ORDER A1*B1
                        ADC     R0
                        ADD     R4,R0   ;HIGH ORDER A1*B1
                        TST     (SP)+   ;POP B2
                        MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
                        .ENDC
7322 006101             ROL     R1      ;SHIFT OUT NORMAL BIT
7324 006100             ROL     R0
7326 103403             BCS     NOM$30  ;JUMP IF IT WAS FOUND
7330 006101             ROL     R1
7332 006100             ROL     R0      ;MUST HAVE GOT IT NOW
7334 005304             DEC     R4      ;ADJUST EXPONENT
7336 162704 NOM$30:     SUB     #200,R4 ;TAKE OUT ONE OF THE EXCESS 128'S
     000200
7342 003436             BLE     UND$30  ;JUMP IF UNDERFLOW
7344 022704             CMP     #377,R4
     000377
7350 002427             BLT     OVR$30  ;JUMP IF OVERFLOW
7352 105001             CLRB    R1
7354 150001             BISB    R0,R1
7356 000301             SWAB    R1
7360 105000             CLRB    R0
7362 150400             BISB    R4,R0
7364 000300             SWAB    R0
7366 006026             ROR     (SP)+   ;GET PRODUCT SIGN
7370 006000             ROR     R0      ;INSERT IT IN RESULT
```

```
212 7372 006001          ROR     R1
213 7374 005501          ADC     R1
214 7376 005500          ADC     R0
215 7400 103414          BCS     OV1$30  ;JUMP IF OVERFLOW ON ROUND
216 7402 102413          BVS     OV1$30
217 7404 010066 OUT$30:  MOV     R0,RESLT(SP)    ;PUT OUT ANSWER
         000010
218 7410 010166          MOV     R1,RESLT+2(SP)
         000012
219 7414 012605          MOV     (SP)+,R5
220 7416 012604          MOV     (SP)+,R4
221 7420 022626          CMP     (SP)+,(SP)+     ;FLUSH TOP ARGUMENT
222 7422 000134          JMP     @(R4)+  ;RETURN
223                      .IFDF   EAE!MULDIV
224              ZE2$30: CMP     (SP)+,(SP)+     ;POP B1,B2
225                      .ENDC
226 7424 022626 ZE1$30:  CMP     (SP)+,(SP)+     ;POP SIGN AND EXPONENT
227 7426 000411          BR      ZER$30
228 7430 005726 OVR$30:  TST     (SP)+   ;FLUSH SIGN
229 7432 012700 OV1$30:  MOV     #6003,R0        ;ERROR 3,12
         006003
230 7436 000403          BR      ECL$30
231 7440 012700 UND$30:  MOV     #3405,R0        ;ERROR 5,7
         003405
232 7444 005726          TST     (SP)+   ;FLUSH SIGN
233 7446 004567 ECL$30:  JSR     R5,$ERRA        ;CALL ERROR
         002344
234 7452 005000 ZER$30:  CLR     R0      ;CLEAR RESULT
235 7454 005001          CLR     R1
236 7456 000752          BR      OUT$30
237                      .IFNDF  EAE&MULDIV
238 7460 006204 MLT$30:  ASR     R4      ;TEST NEXT MULTIPLIER BIT
239 7462 103004          BCC     X0$30   ;JUMP IF IT IS 0
240 7464 060301 MT1$30:  ADD     R3,R1
241 7466 005500          ADC     R0
242 7470 103406          BCS     COV$30
243 7472 060200          ADD     R2,R0
244 7474 006000 X0$30:   ROR     R0      ;NOW SHIFT PRODUCT
245 7476 006001          ROR     R1
246 7500 005305          DEC     R5      ;COUNT LOOP
247 7502 003366          BGT     MLT$30  ;AGAIN PLEASE
248 7504 000207          RTS     PC      ;RETURN TO CALLER
249 7506 060200 COV$30:  ADD     R2,R0   ;FIRST ADD OVERFLOWED R0
250 7510 000261          SEC             ;SHOW THIS OVERFLOW TO SHIFT
251 7512 000770          BR      X0$30
252 7514 006204 MT0$30:  ASR     R4      ;REDUCED PRECISION MULTIPLY
253 7516 103001          BCC     X00$30
254 7520 060200          ADD     R2,R0   ;USE ONLY HIGH ORDER MULTIPLICAND
255 7522 006000 X00$30:  ROR     R0
256 7524 006001          ROR     R1
257 7526 005305          DEC     R5
258 7530 003371          BGT     MT0$30
259 7532 000207          RTS     PC
260                      .ENDC
261                      .ENDC
262                      .ENDC
```

```
                        .TITLE  $NEG02
                        .IFDF   CND$31
                ;
                ;       $NEG    V002A
                ;
                ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                ;
                        .GLOBL  $NGI,$NGR,$NGD,$ERR
                ;       INTEGER, REAL AND DOUBLE PRECISION NEGATION.
                ;       CALLED IN THE POLISH MODE.
                ;       NEGATES THE ITEM ON TOP OF THE STACK.
            000004      R4=%4
            000005      R5=%5
            000006      SP=%6
17534 005416 $NGI:  NEG     @SP     ;NEGATE AN ITEGER
17536 102406        BVS     OVR$31  ;JUMP IF 100000
17540 000134        JMP     @(R4)+  ;RETURN
17542           $NGR:
17542 005716 $NGD:  TST     @SP
17544 001402        BEQ     ZER$31  ;JUMP IF 0 TO AVOID -0.
17546 062716        ADD     #100000,@SP     ;INVERT FLOATING SIGN
      100000
17552 000134 ZER$31: JMP    @(R4)+
17554 004567 OVR$31: JSR    R5,$ERR ;ERROR 3,11
      002226
17560 000134        JMP     @(R4)+
17562    003        .BYTE   3
17563    013        .BYTE   11.
                    .ENDC
```

132

```
1                               .TITLE  $PHR07
2                               .IFDF   CND$32
3                       ;
4                       ;       $PSHR   V007A
5                       ;
6                       ;COPYRIGHT 1971,  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
7                       ;
8
9               000000  R0      =       %0
10              000001  R1      =       %1
11              000002  R2      =       %2
12              000003  R3      =       %3
13              000004  R4      =       %4
14              000005  R5      =       %5
15              000006  SP      =       %6
16              000007  PC      =       %7
17                      ;
18                      ;THIS ROUTINE PLACES ONE, TWO, OR FOUR ITEMS ON THE STACK
19                      ;       FROM THE REGISTERS R0-R3.  IT IS USED AFTER FUNCTION
20                      ;       CALLS TO PLACE THE FUNCTION RESULT ON THE STACK
21                      ;
22                              .GLOBL  $PSHR5,$PSHR4,$PSHR3,$PSHR2,$PSHR1
23  17564               $PSHR5:
24  17564  010346      $PSHR4: MOV     R3,-(SP)        ;PUSH FOUR WORDS
25  17566  010246              MOV     R2,-(SP)
26  17570  010146      $PSHR3: MOV     R1,-(SP)        ;PUSH TWO WORDS
27  17572               $PSHR2:
28  17572  010046      $PSHR1: MOV     R0,-(SP)        ;PUSH ONE WORD
29  17574  000134              JMP     @(R4)+
30                      ;
31                              .ENDC
```

```
                        .TITLE  $PPR04
                        .IFDF   CND$$3
                ;
                ;       $POPR5  V004A
                ;
                ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                ;
                ;
        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 SP=%6
        000007 PC=%7
                ;
                ; THIS ROUTINE REMOVES TWO OR FOUR ITEMS FROM THE STACK
                ;       AND PLACES THEM IN REGISTERS R0-R3.  IT IS USED IN EXTER
                ;       FUNCTIONS TO RETURN THE FUNCTION VALUE IN THE REGISTERS
                ;
                        .GLOBL  $POPR5,$POPR4,$POPR3
                ;
17576           $POPR5:
17576 012600    $POPR4: MOV     (SP)+,R0        ;POP FOUR WORDS
17600 012601            MOV     (SP)+,R1
17602 012602            MOV     (SP)+,R2
17604 012603            MOV     (SP)+,R3
17606 000134            JMP     @(R4)+
17610 012600    $POPR3: MOV     (SP)+,R0        ;POP TWO WORDS
17612 012601            MOV     (SP)+,R1
17614 000134            JMP     @(R4)+
                ;
                        .ENDC
```

134

```
1                               .TITLE  $RD02
2                               .IFDF   CND$$4
3                               .GLOBL  $RD
4                       ;       $RD     THE REAL TO DOUBLE PRECISION CONVERTER
5                       ;
6                       ;       $RD     V002A
7                       ;
8                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
9                       ;       APPEND ZEROS TO THE TOP STACK ITEM TO
10                      ;       MAKE IT DOUBLE PRECISION FORMAT
11              000004          R4=%4
12              000006          SP=%6
13              000000          F0=%0
14              000001          F1=%1
15                              .IFDF   FPU
16                      $RD:    .WORD   170011  ;;SETD
17                              .WORD   177426  ;;LDCFD (SP)+,F0        ;CONVERT ARG
18                              .WORD   174046  ;;STD   F0,-(SP)
19                              JMP     @(R4)+
20                              .ENDC
21                              .IFNDF  FPU
22  17616 016646 $RD:           MOV     2(SP),-(SP)     ;MOVE LOW ORDER PART
          000002
23  17622 016646                MOV     2(SP),-(SP)     ;MOVE HIGH ORDER PART
          000002
24  17626 005066                CLR     4(SP)   ;INSERT TRAILING ZEROS
          000004
25  17632 005066                CLR     6(SP)
          000006
26  17636 000134                JMP     @(R4)+
27                              .ENDC
28                              .ENDC
```

135

```
                        .TITLE  $RI04
                        .IFDF   CND$35
                        .GLOBL  $RI,$DI,$ERR
                ;       REAL TO INTEGER CONVERSION.
                ;
                ;       $RI     V004A
                ;
                ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
                ;       ARGUMENT IS A DOUBLE WORD REAL NUMBER ON THE TOP
                ;       OF THE STACK.
                ;       TRUNCATE IT AND CONVERT IT TO AN INTEGER ON THE
                ;       TOP OF THE STACK.
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        177304          MQ=177304
        177314          LSH=177314
        000000          F0=%0
                        .IFDF   FPU
                $DI:    SETD    ;               DOUBLE PRECISION
                        BR      RID$35;
                $RI:    SETF    ;               SINGLE PRECISION
                RID$35: SETI    ;               SHORT INTEGERS
                        LDD     (SP)+,F0;       GET ARGUMENT
                        STCDI   F0,-(SP);       CONVERT TO STACK
                        JMP     @(R4)+;         RETURN
                        .ENDC
                        .IFNDF  FPU
17640 012666    $DI:    MOV     (SP)+,2(SP);    TRUNCATE TO REAL FORMAT
      000002
17644 012666            MOV     (SP)+,2(SP);
      000002
17650 005002    $RI:    CLR     R2      ;CLEAR WORK SPACE
17652 005202            INC     R2      ;SET UP NORMAL BIT
17654 012601            MOV     (SP)+,R1        ;GET REAL ARGUMENT
17656 006116            ROL     @SP     ;GET SIGN
17660 006101            ROL     R1      ;AND
17662 006146            ROL     -(SP)   ;SAVE IT
17664 110103            MOVB    R1,R3   ;GET HIGH ORDER FRACTION
17666 105001            CLRB    R1
17670 000301            SWAB    R1      ;GET EXPONENT
17672 162701            SUB     #201,R1
      000201
17676 002433            BLT     ZER$35  ;JUMP IF IT IS TOO SMALL
17700 001413            BEQ     ONE$35
17702 022701            CMP     #15.,R1
      000017
17706 002422            BLT     OVR$35  ;JUMP IF IT IS TOO BIG
17710 000303            SWAB    R3      ;FORM 16 BITS OF HIGH ORDER FRACTION
17712 105003            CLRB    R3
17714 156603            BISB    3(SP),R3
      000003
17720           SFT$35:
```

136

```
53                              .IFNDF  EAE&MULDIV
54  17720 006103                ROL     R3          ;GET NEXT BIT
55  17722 006102                ROL     R2
56  17724 005301 DEC$35:        DEC     R1          ;DECREASE EXPONENT
57  17726 003374                BGT     SFT$35      ;GO AGAIN IF NOT DONE
58                              .ENDC
59                      ;
60                      ;       EAE CODE
61                              .IFDF   EAE
62                              MOV     #MQ,R0      ;POINT TO MQ
63                              MOV     R3,@R0      ;INSERT FRACTION
64                              MOV     R2,-(R0)
65                              MOV     R1,@#LSH            ;SHIFT LEFT
66                              MOV     @R0,R2      ;RESULT TO REG
67                              .ENDC
68                      ;       MULDIV CODE
69                              .IFDF   MULDIV
70                              .WORD   073201      ;;ASHC  R1,R2
71                              .ENDC
72  17730 005402 DNE$35:        NEG     R2          ;MAKE -
73  17732 102406                BVS     NGM$35      ;JUMP IF POSSIBLE NEGMAX
74  17734 003007                BGT     OVR$35      ;JUMP IF MORE THAN 15 BITS
75  17736 006026 SGN$35:        ROR     (SP)+       ;GET SIGN
76  17740 103401                BCS     OUT$35      ;JUMP IF -
77  17742 005402                NEG     R2          ;- RESULT
78  17744 010216 OUT$35:        MOV     R2,@SP      ;STORE INTEGER RESULT
79  17746 000134                JMP     @(R4)+      ;RETURN TO CALLER
80  17750 006026 NGM$35:        ROR     (SP)+
81  17752 103774                BCS     OUT$35      ;OK IF RESULT TO BE -
82  17754 005746 OVR$35:        TST     -(SP)       ;FAKE SIGN
83  17756 004567                JSR     R5,$ERR     ;ERROR 3,22
            002024
84  17762 000401                BR      ZER$35
85  17764     003               .BYTE   3
86  17765     026               .BYTE   22.
87  17766 005002 ZER$35:        CLR     R2          ;ANSWER IS 0
88  17770 000762                BR      SGN$35
89                              .ENDC
90                              .ENDC
```

```
1                               .TITLE  $SGL02
2                               .IFDF   CND$36
3                       ;
4                       ;       SNGL    V002A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8                               .GLOBL  SNGL,$ERR
9                       ;       THE FORTRAN SNGL FUNCTION
10                      ;       CALLING SEQUENCE:
11                      ;       JSR     R5,SNGL
12                      ;       BR      A
13                      ;       .WORD   ARGUMENT ADDRESS
14                      ;A:
15                      ;       RETURNS THE ARGUMENT ROUNDED TO SINGLE
16                      ;       PRECISON REAL FORMAT IN R0, R1.
17                      ;
18              000000          R0=%0
19              000001          R1=%1
20              000004          R4=%4
21              000005          R5=%5
22  17772 016504 SNGL:  MOV     2(R5),R4        ;GET ADDRESS
                000002
23  17776 012400        MOV     (R4)+,R0        ;GET HIGH ORDER
24  20000 012401        MOV     (R4)+,R1        ;GET LOW ORDER
25  20002 011404        MOV     @R4,R4  ;GET NEXT WORD
26  20004 006104        ROL     R4      ;GET ROUND BIT
27  20006 005501        ADC     R1      ;ROUND REAL
28  20010 005500        ADC     R0
29  20012 103402        BCS     OVR$36  ;JUMP IF OVERFLOW ON ROUND
30  20014 102401        BVS     OVR$36
31  20016 000205        RTS     R5      ;RETURN TO CALLER
32  20020 004567 OVR$36: JSR    R5,$ERR ;ERROR 4,12
                001762
33  20024 000205        RTS     R5
34  20026     004       .BYTE   4
35  20027     014       .BYTE   12.
36                      .ENDC
```

```
1                               .TITLE  $SIN04
2                               .IFOF   CND$37
3                       ;
4                       ;       SINCOS  V004A
5                       ;
6                       ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
7                       ;
8                               .GLOBL  SIN,COS;
9                               .IFNDF  FPU
10                              .GLOBL  $ADR,$MLR,$SBR,$DVR,$INTR,$POLSH
11                              .ENDC
12                      ;       SIN     COS     THE REAL SIN AND COSINE FUNCTIONS
13                      ;       CALLING SEQUENCE:
14                      ;       JSR     R5,SIN  (OR COS)
15                      ;       BR      A
16                      ;       .WORD   ARG ADDRESS
17                      ;A:
18                      ;       RETURNS SIN OR COS OF ARG IN R0 AND R1
19              000000          R0=%0
20              000001          R1=%1
21              000002          R2=%2
22              000003          R3=%3
23              000004          R4=%4
24              000005          R5=%5
25              000006          SP=%6
26              000007          PC=%7
27              000000          F0=%0
28              000001          F1=%1
29              000002          F2=%2
30              000003          F3=%3
31                              .IFNDF  FPU
32  20030 016504 COS:   MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
                000002
33  20034 005046        CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
34  20036 016446        MOV     2(R4),-(SP)     ;PUSH ARGUMENT
                000002
35  20042 011446        MOV     @R4,-(SP)
36  20044 012746        MOV     #007733,-(SP)   ;PUSH PI/2
                007733
37  20050 012746        MOV     #040311,-(SP)
                040311
38  20054 004467        JSR     R4,$POLSH       ;ENTER POLISH MODE
                001564
39  20060 002010'       .WORD   $ADR,SNC$37     ;COS(X)=SIN(X+PI/2)
    20062 020100'
40  20064 016504 SIN:   MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
                000002
41  20070 005046        CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
42  20072 016446        MOV     2(R4),-(SP)
                000002
43  20076 011446        MOV     @R4,-(SP)       ;PUSH ARGUMENT
44  20100 006316 SNC$37: ASL    @SP     ;REMOVE AND SAVE SIGN
45  20102 006066        ROR     4(SP)   ;IN QUADRANT FLAG
                000004
46  20106 006016        ROR     @SP
47  20110 012746        MOV     #007733,-(SP)   ;PUSH 2*PI
                007733
```

```
48 20114 012746          MOV     #040711,-(SP)
         040711
49 20120 004467          JSR     R4,$POLSH       ;ENTER POLISH MODE
         001520
50 20124 013256'         .WORD   $DVR    ;X/2PI
51 20126 020222'         .WORD   DUP$37  ;2 COPIES
52 20130 003142'         .WORD   $INTR   ;INT(X/2PI)
53 20132 002004'         .WORD   $SBR    ;FRACT(X/2PI)
54 20134 020234'         .WORD   X4$37   ;4*FRACT(X/2PI)
55 20136 020222'         .WORD   DUP$37  ;2 COPIES
56 20140 003142'         .WORD   $INTR   ;INT(4*FRACT(X/2PI))
57 20142 020246'         .WORD   QUD$37  ;SAVE INT(......)
58 20144 002004'         .WORD   $SBR    ;Y=FRACT(4*FRACT(X/2PI))
59 20146 020254'         .WORD   QST$37  ;REDUCE Y TO (-1,1)
60 20150 020222'QSE$37:  .WORD   DUP$37  ;2 COPIES
61 20152 020222'         .WORD   DUP$37  ;3 COPIES
62 20154 017162'         .WORD   $MLR    ;Y*Y
63 20156 020324'         .WORD   PLY$37  ;PUSH COEFFICIENTS
64 20160 017162'         .WORD   $MLR    ;A4*Y**2
65 20162 002010'         .WORD   $ADR    ;A4*Y**2+A3
66 20164 017162'         .WORD   $MLR
67 20166 002010'         .WORD   $ADR
68 20170 017162'         .WORD   $MLR
69 20172 002010'         .WORD   $ADR
70 20174 017162'         .WORD   $MLR
71 20176 002010'         .WORD   $ADR
72 20200 017162'         .WORD   $MLR    ;((((A4*Z+A3)*Z+A3)*Z+A2)*Z
73                                       ;+A1)*Z+A0)*Z      Z=Y*Y
74 20202 020204'         .WORD   RTN$37
75 20204 012600 RTN$37:  MOV     (SP)+,R0        ;POP HIGH ORDER RESULT
76 20206 012601          MOV     (SP)+,R1
77 20210 005725          TST     (SP)+   ;POP QUADRANT FLAG
78 20212 002002          BGE     RT1$37  ;JUMP IF ARGUMENT WAS +
79 20214 062700          ADD     #100000,R0      ;SIN(-X)=-SIN(X)
         100000
80 20220 000205 RT1$37:  RTS     R5      ;BACK TO CALLER
81                       ;
82 20222 016646 DUP$37:  MOV     2(SP),-(SP)     ;DUPLICATE STACK ITEM
         000002
83 20226 016646          MOV     2(SP),-(SP)
         000002
84 20232 000134          JMP     @(R4)+
85                       ;
86 20234 005716 X4$37:   TST     @SP     ;CHECK FOR 0 FRACTION
87 20236 001762          BEQ     RTN$37  ;QUIT NOW
88 20240 105266          INCB    1(SP)   ;QUADRUPLE STACK ITEM
         000001
89 20244 000134          JMP     @(R4)+
90                       ;
91 20246 051666 QUD$37:  BIS     @SP,8.(SP)      ;SAVE QUADRANT NUMBER
         000010
92 20252 000134          JMP     @(R4)+
93                       ;
94 20254 105766 QST$37:  TSTB    4(SP)   ;TEST QUADRANT
         000004
95 20260 001413          BEQ     Q13$37  ;JUMP IF FIRST OR THIRD QUAD
96 20262 062716          ADD     #100000,@SP     ;NEGATE STACK ITEM
```

```
                100000
97  20266 005046          CLR     -(SP)   ;PUSH A FLOATING 1.
98  20270 012746          MOV     #40200,-(SP)
          040200
99  20274 004467          JSR     R4,$POLSH       ;ENTER POLISH
          001344
100 0300  002010'         .WORD   $ADR;QSR$37     ;X=1.-X
    0302  020304'
101 0304  012704 QSR$37:  MOV     #QSE$37,R4      ;POINT BACK INTO LIST
          020150'
102 0310  106266 Q1$37:   ASRB    5(SP)   ;TEST QUADRANT
          000005
103                       ;
104 0314  103002          BCC     QUT$37  ;JUMP IF FIRST OR SECOND
105 0316  062716          ADD     #100000,@SP     ;NEGATE STACK ITEM
          100000
106 0322  000134 QUT$37:  JMP     @(R4)+
107                       ;
108 0324  012600 PLY$37:  MOV     (SP)+,R0        ;SAVE Y*Y
109 0326  012601          MOV     (SP)+,R1
110 0330  012702          MOV     #CON$37+4,R2    ;POINT TO LIST OF COEFFICIENTS
          020404'
111 0334  012703          MOV     #5,R3
          000005
112 0340  000402          BR      PY1$37
113 0342  010146 PY2$37:  MOV     R1,-(SP)        ;PUSH Y*Y
114 0344  010046          MOV     R0,-(SP)
115 0346  014246 PY1$37:  MOV     -(R2),-(SP)
116 0350  014246          MOV     -(R2),-(SP)
117 0352  005303          DEC     R3      ;COUNT COEFFICIENTS
118 0354  003372          BGT     PY2$37
119 0356  000134          JMP     @(R4)+
120                       .ENDC
121                 ;
122                       .IFDF   FPU
123                 COS:  SETD    ;               DOUBLE PRECISION FP
124                       LDCFD   @2(R5),F0;      GET ARGUMENT
125                       ADDD    PI2$37,F0;      COS(X)= SIN(X+PI/2)
126                       BR      SNG$37;
127                 SIN:  SETD    ;               DOUBLE PRECISON FP
128                       LDCFD   @2(R5),F0;      GET ARGUMENT
129                 SNG$37: SETI  ;               SHORT INTEGERS
130                       MOV     #FCO$37,R0;     POINTER TO CONSTANTS
131                       CLR     R4;             SIGN FLAG: + ARG
132                       CFCC    ;               GET SIGN OF ARGUMENT
133                       BGE     POS$37;
134                       INC     R4;             SIGN FLAG: - ARG
135                       ABSD    F0;             REMOVE ARGUMENT SIGN
136                 POS$37: DIVD  (R0)+,F0;       X/(PI/2)
137                       MODD    #0.25,F0;       F0=FRACT(X/2PI)
138                       SETF    ;               SINGLE PRECISION FP
139                       LDCDF   F0,F0;          CONVERT ARGUMENT
140                       CFCC    ;
141                       BEQ     RTN$37;         CHECK FOR 0 FRACTION
142                       MODF    #4.0,F0;        F0=FRACT(4*FRACT(X/2PI))
143                       STCFI   F1,R1;          QUAD=INT(4*FRACT(X/2PI))
144                       ROR     R1;
```

```
                        BCC     Q13$37;         JUMP IF FIRST OR THIRD QUAD
                        NEGF    F0;
                        ADDF    #1.0,F0;        Y=1.0-X
                Q13$37: ROR     R1;
                        BCC     Q12$37;         JUMP IF FIRST OR SECOND QUAD
                        NEGF    F0;             Y= -Y
                ;
                Q12$37: LDF     F0,F2;
                        MULF    F2,F2;          Z=Y**2
                        MOV     #4,R1;          COUNT OF CONSTANTS FOR POLY
                        LDF     (R0)+,F1;       INITIALIZE ACCUMULATOR
                XPO$37: MULF    F2,F1;
                        DEC     R1;             COUN
                        ADDF    (R0)+,F1;       F1:= Z*F1 + C(I)
                        BGT     XPO$37;         LOOP
                        MULF    F1,F0;          F0:= Y*F1
                        TST     R4;             TEST SIGN FLAG
                        BEQ     RTN$37;
                        NEGF    F0;             SIN(-X) = -SIN(X)
                RTN$37: STF     F0,-(SP);       MOVE RESULT TO STACK
                        MOV     (SP)+,R0;               AND THENCE TO R0,R1
                        MOV     (SP)+,R1;
                        RTS     R5;             EXIT
                ;
                FCO$37:
                PI2$37: .WORD   040311,007732;  PI/2 (DOUBLE PRECISION)
                        .WORD   121041,054302;
                ;
                ;       ORDER-DEPENDENT CONSTANTS
                ;
                        .ENDC
                ;
0360 035036             .WORD   035036,153672;  .00015148419
0362 153672
                ;
0364 136231             .WORD   136231,023143;  -.00467376557
0366 023143
                ;
0370 037243             .WORD   037243,032130;  .0796896793
0372 032130
                ;
0374 140045             .WORD   140045,056741;  -.645963711
0376 056741
                ;
0400 040311     CON$37: .WORD   040311,007733;  1.570796318
0402 007733
                        .ENDC
```

```
1                                       .TITLE  $TNH02
2                                       .IFDF   CND$38
3                                       .GLOBL  TANH,EXP,$ADR,$SBR,$MLR,$DVR,$FCALL
4                                       .GLOBL  $POLSH,$PSHR3
5                               ;       THE FORTRAN TANH FUNCTION
6                               ;       TANH    V002A
7                               ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
8                               ;       CALLING SEQUNCE:
9                               ;       JSR     R5,TANH
10                              ;       BR      A
11                              ;       .WORD   ARGUMENT ADDRESS
12                              ;A:
13                              ;       RETURNS (EXP(2*ARG) -1)/(EXP(2*ARG)+1) IN R0,R1.
14                              ;
15          000000                      R0=%0
16          000001                      R1=%1
17          000004                      R4=%4
18          000005                      R5=%5
19          000006                      SP=%6
20          000007                      PC=%7
21  20404   010546  TANH:       MOV     R5,-(SP)        ;SAVE RETURN POINTER
22  20406   016505                  MOV     2(R5),R5        ;GET ARG ADDRESS
              000002
23  20412   011500                  MOV     @R5,R0  ;GET HIGH ORDER ARG
24  20414   001554                  BEQ     ZER$38  ;JUMP IF ARG=0
25  20416   006300                  ASL     R0
26  20420   105000                  CLRB    R0
27  20422   000300                  SWAB    R0      ;GET EXPONENT
28  20424   020027                  CMP     R0,#205
              000205
29  20430   002410                  BLT     STE$38  ;JUMP IF ABS(ARG) <16.
30  20432   012700                  MOV     #40200,R0       ;ANSWER IS 1.*SIGN(ARG)
              040200
31  20436   005001                  CLR     R1
32  20440   005715                  TST     @R5     ;TEST ARG SIGN
33  20442   002052                  BGE     OUT$38
34  20444   062700                  ADD     #100000,R0      ;MAKE -1.
              100000
35  20450   000447                  BR      OUT$38
36  20452   020027  STE$38:  CMP     R0,#177
              000177
37  20456   003007                  BGT     TAN$38  ;JUMP IF >1/2
38  20460   020027                  CMP     R0,#164
              000164
39  20464   002043                  BGE     SML$38  ;USE CONTINUED FRACTION FOR THIS RANGE
40  20466   016501                  MOV     2(R5),R1
              000002
41  20472   011500                  MOV     @R5,R0  ;IF ABS(X)<2**-12, LET TANH=X
42  20474   000435                  BR      OUT$38
43  20476   016546  TAN$38:  MOV     2(R5),-(SP)     ;PUSH 2*ARG ON STACK
              000002
44  20502   011546                  MOV     @R5,-(SP)
45  20504   062716                  ADD     #200,@SP        ;DOUBLE ARG
              000200
46  20510   010605                  MOV     SP,R5   ;SET UP CALL TO EXP, ARG POINTER
47  20512   012704                  MOV     #EXP,R4 ;POINT TO EXP
              014554!
```

```
48 20516 004767          JSR     PC,$FCALL
         174402
49 20522 010146          MOV     R1,-(SP)        ;PUSH E**2ARG
50 20524 010046          MOV     R0,-(SP)
51 20526 005046          CLR     -(SP)   ;PUSH 1.
52 20530 012746          MOV     #40200,-(SP)
         040200
53 20534 010146          MOV     R1,-(SP)        ;PUSH E**2ARG
54 20536 010046          MOV     R0,-(SP)
55 20540 005046          CLR     -(SP)
56 20542 012746          MOV     #40200,-(SP)    ;PUSH 1.
         040200
57 20546 004467          JSR     R4,$POLSH       ;GET (E**2X -1)/(E**2X +1)
         001072
58 20552 002004'         .WORD   $SBR,UP$38,$ADR,$DVR,UPL$38
   20554 020754'
   20556 002010'
   20560 013256'
   20562 020564'
59 20564 012600 UPL$38: MOV      (SP)+,R0        ;POP RESULT
60 20566 012601          MOV     (SP)+,R1
61 20570 012605 OUT$38: MOV      (SP)+,R5        ;RESTORE RETURN
62 20572 000205          RTS     R5      ;RETURN TO USER
63 20574 016501 SML$38: MOV      2(R5),R1        ;GET ARG
         000002
64 20600 011500          MOV     @R5,R0
65 20602 004467          JSR     R4,$POLSH
         001036
66 20606 017570'         .WORD   $PSHR3,$PSHR3,$PSHR3,$MLR,XSQ$38        ;GET X A
   20610 017570'
   20612 017570'
   20614 017162'
   20616 020620'
67 20620 016646 XSQ$38: MOV      2(SP),-(SP)     ;GET X SQUARE
         000002
68 20624 016646          MOV     2(SP),-(SP)
         000002
69 20630 004467          JSR     R4,$POLSH
         001010
70 20634 020734'         .WORD   P35$38,$ADR,ONE$38      ;SET UP NUMERATOR
   20636 002010'
   20640 020666'
71 20642 017570'         .WORD   $PSHR3,P45$38,$PSHR3,$DVR,$ADR,$ADR,$DVR
   20644 020712'
   20646 017570'
   20650 013256'
   20652 002010'
   20654 002010'
   20656 013256'
72 20660 002004'         .WORD   $SBR,$MLR,UPL$38
   20662 017162'
   20664 020564'
73                       ;       THE ABOVE COMPUTES X(1-((Y+35...)/(Y+45...+105../Y)))
74                       ;       WHERE Y=X*X
75 20666 016600 ONE$38: MOV      4(SP),R0        ;GET XSQUARE AGAIN
         000004
76 20672 016601          MOV     6(SP),R1
```

```
                000006
77  20676 005066          CLR     6(SP)   ;INSERT A 1.
                000006
78  20702 012766          MOV     #40200,4(SP)
                040200
                000004
79  20710 000134          JMP     @(R4)+
80  20712 012746 P45$38:  MOV     #136237,-(SP)   ;PUSH 45.1842
                136237
81  20716 012746          MOV     #41464,-(SP)
                041464
82  20722 012746 P15$38:  MOV     #165707,-(SP)   ;PUSH 105.4605
                165707
83  20726 012746          MOV     #41722,-(SP)
                041722
84  20732 000134          JMP     @(R4)+
85  20734 012746 P35$38:  MOV     #116457,-(SP)   ;PUSH 35.1535
                116457
86  20740 012746          MOV     #41414,-(SP)
                041414
87  20744 000134          JMP     @(R4)+
88  20746 005000 ZER$38:  CLR     R0
89  20750 005001          CLR     R1
90  20752 000706          BR      OUT$38
91                      ;
92  20754 012666 UP$38:   MOV     (SP)+,10.(SP)   ;MOVE STACK ITEM UP
                000012
93  20760 012666          MOV     (SP)+,10.(SP)
                000012
94  20764 000134          JMP     @(R4)+
95                      ;
96                        .ENDC
```

```
                        .TITLE  $ATN03
                        .IFDF   CND$39
                ;
                ;       ATAN    V003A
                ;
                ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                ;
                        .GLOBL  ATAN,ATAN2;
                        .IFNDF  FPU
                        .GLOBL  $ADR,$SBR,$MLR,$DVR,$POLSH,$POPR3;
                        .ENDC
                ;       THE FORTRAN ATAN AND ATAN2 FUNCTIONS
                ;       CALLING SEQUENCE FOR ATAN:
                ;       JSR     R5,ATAN
                ;       BR      A
                ;       .WORD   ARGUMENT ADDRESS
                ;A:
                ;       RETURNS ARCTAN(ARG) IN R0 AND R1.
                ;
                ;       CALLING SEQUENCE FOR ATAN2:
                ;       JSR     R5,ATAN2
                ;       BR      A
                ;       .WORD   ARGUMENT 1 ADDRESS
                ;       .WORD   ARGUMENT 2 ADDRESS
                ;A:
                ;       RETURNS ACRTAN(ARG1/ARG2) IN R0 AND R1.
                ;       IF ABS(ARG1/ARG2) > 2**24, THE RESULT IS
                ;       SIGN(ARG1)*PI/2.
                ;       IF ARG2 <0 THE RESULT IS ARCTAN(ARG1/ARG2) +
                ;       SIGN(ARG1)*PI.
                ;
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000000          F0=%0
        000001          F1=%1
        000002          F2=%2
        000003          F3=%3
        000004          F4=%4
        000005          F5=%5
                        .IFNDF  FPU
20766 005046 ATAN2:     CLR     -(SP)   ;CLEAR SIGN FLAG
20770 005046            CLR     -(SP)   ;CLEAR ATAN2 BIAS
20772 005046            CLR     -(SP)
20774 005046            CLR     -(SP)   ;CLEAR QUADRANT BIAS
20776 005046            CLR     -(SP)
21000 016504            MOV     2(R5),R4        ;GET FIRST ARG ADDRESS
      000002
21004 016446            MOV     2(R4),-(SP)     ;GET FIRST ARG
      000002
21010 011446            MOV     @R4,-(SP)
21012 011600            MOV     @SP,R0  ;ARG1 TO R0
```

```
56 21014 016504 000004          MOV     4(R5),R4        ;GET SECOND ARG ADDRESS
57 21020 016446 000002          MOV     2(R4),-(SP)     ;GET SECOND ARG
58 21024 011446                 MOV     @R4,-(SP)
59 21026 011601                 MOV     @SP,R1  ;ARG2 TO R1
60 21030 001437                 BEQ     INF$39  ;JUMP IF DENOMINATOR IS 0
61 21032 006300                 ASL     R0      ;GET ABS VAL ARG1
62 21034 105000                 CLRB    R0      ;GET EXPONENT
63 21036 000300                 SWAB    R0
64 21040 006301                 ASL     R1
65 21042 105001                 CLRB    R1      ;GET EXPONENT ARG2
66 21044 000301                 SWAB    R1
67 21046 160100                 SUB     R1,R0   ;GET EXPONENT DIFFERENCE
68 21050 022700 000032          CMP     #26.,R0 ;CHECK MAGNITUDE
69 21054 002425                 BLT     INF$39  ;TREAT AS INFINITY
70 21056 004467 000562 DIV$39:  JSR     R4,$FOLSH
71 21062 013256'                .WORD   $DVR,UPL$39     ;GET ARG1/ARG2
   21064 021066'
72 21066 005775 000004 UPL$39:  TST     @4(R5)  ;IF ARG2 >0, BIAS =0
73 21072 002014                 BGE     ATE$39  ;IF ARG2<0, BIAS=SIGN(ARG1)*PI
74 21074 012766 040511 000010   MOV     #040511,8.(SP)  ;PI
75 21102 012766 007733 000012   MOV     #007733,10.(SP)
76 21110 005775 000002          TST     @2(R5)  ;TEST ARG1
77 21114 002003                 BGE     ATE$39
78 21116 062766 100000 000010   ADD     #100000,8.(SP)  ;-PI
79 21124 005716 ATE$39: TST     @SP     ;SET CODES
80 21126 000426                 BR      AT1$39  ;JOIN MAIN ROUTINE
81 21130 062705 000022 INF$39:  ADD     #18.,SP ;FLUSH STACK
82 21134 012700 040311          MOV     #040311,R0      ;ANS = SIGN(ARG1)*PI/2
83 21140 012701 007733          MOV     #007733,R1
84 21144 005775 000002          TST     @2(R5)  ;TEST ARG1
85 21150 002002                 BGE     INR$39  ;JUMP IF +PI/2
86 21152 062700 100000          ADD     #100000,R0      ;-PI/2
87 21156 000205 INR$39: RTS     R5      ;RETURN TO USER
88                      ;
89 21160 005046 ATAN:   CLR     -(SP)   ;CLEAR SIGN FLAG
90 21162 005046                 CLR     -(SP)   ;CLEAR ATAN2 BIAS
91 21164 005046                 CLR     -(SP)
92 21166 005046                 CLR     -(SP)   ;CLEAR QUADRANT BIAS
93 21170 005046                 CLR     -(SP)
94 21172 016504                 MOV     2(R5),R4        ;GET ARG ADDRESS
```

```
                000002
95  21176 016446                MOV     2(R4),-(SP)     ;GET LOW ORDER ARG
                000002
96  21202 011446                MOV     @R4,-(SP)       ;GET HIGH ORDER
97  21204 002004 AT1$39:        BGE     PLU$39  ;JUMP IF QUADRANT 1 OR 3
98  21206 062716                ADD     #100000,@SP     ;GET ABS VALUE
                100000
99  21212 005266                INC     12.(SP) ;FLAG =
                000014
100 1216  021627 PLU$39:        CMP     @SP,#40200      ;CHECK IF <1.
                040200
101 1222  103431                BLO     LE1$39  ;JUMP IF <1.
102 1224  003003                BGT     GT1$39  ;>1.
103 1226  005766                TST     2(SP)   ;CHECK LOW ORDER
                000002
104 1232  001425                BEQ     LE1$39  ;=1.
105 1234  012766 GT1$39:        MOV     #140311,4(SP)   ;=PI/2
                140311
                000004
106 1242  012766                MOV     #007733,6(SP)   ;ATAN(X)=PI/2-ATAN(1/X)
                007733
                000006
107 1250  005366                DEC     12.(SP) ;ADJUST SIGN
                000014
108 1254  016646                MOV     2(SP),-(SP)     ;MOVE ARG DOWN
                000002
109 1260  016646                MOV     2(SP),-(SP)
                000002
110 1264  012766                MOV     #40200,4(SP)    ;INSERT 1.
                040200
                000004
111 1272  005066                CLR     6(SP)
                000006
112 1276  004467                JSR     R4,$POLSH       ;COMPUTE 1./X
                000342
113 1302  013256'               .WORD   $DVR,LE1$39
    1304  021306'
114 1306  016646 LE1$39:        MOV     2(SP),-(SP)     ;MOVE ARG DOWN
                000002
115 1312  016646                MOV     2(SP),-(SP)
                000002
116 1316  005066                CLR     4(SP)
                000004
117 1322  005066                CLR     6(SP)
                000006
118 1326  021627                CMP     @SP,#037611     ;TAN(15)
                037611
119 1332  103445                BLO     L15$39  ;JUMP IF LESS THAN TAN(15)
120 1334  101004                BHI     TNS$39  ;JUMP IF >
121 1336  026627                CMP     2(SP),#030243
                000002
                030243
122 1344  101440                BLOS    L15$39
123 1346  012766 TNS$39:        MOV     #040006,4(SP)   ;INSERT PI/6
                040006
                000004
124 1354  012766                MOV     #005222,6(SP)
```

```
          005222
          000006
125 1362 011600          MOV     @SP,R0  ;ARG TO REGS
126 1364 016601          MOV     2(SP),R1
          000002
127 1370 012746          MOV     #131727,-(SP)   ;PUSH -ROOT 3
          131727
128 1374 012746          MOV     #140335,-(SP)
          140335
129 1400 010146          MOV     R1,-(SP)
130 1402 010046          MOV     R0,-(SP)        ;PUSH ARG
131 1404 005046          CLR     -(SP)   ;PUSH 1.
132 1406 012746          MOV     #40200,-(SP)
          040200
133 1412 012746          MOV     #131727,-(SP)   ;PUSH ROOT3
          131727
134 1416 012746          MOV     #040335,-(SP)
          040335
135 1422 010146          MOV     R1,-(SP)        ;PUSH ARG
136 1424 010046          MOV     R0,-(SP)
137 1426 004467          JSR     R4,$POLSH       ;TRANSFORM ARG
          000212
138                      ;               (ROOT3*X-1)/(ROOT3 +X)
139 1432 017162'         .WORD   $MLR,$SBR,UP$39,$SBR,$DVR,L15$39
    1434 002004'
    1436 021536'
    1440 002004'
    1442 013256'
    1444 021446'
140 1446 011600 L15$39:  MOV     @SP,R0  ;GET ARG
141 1450 016601          MOV     2(SP),R1
          000002
142 1454 010146          MOV     R1,-(SP)        ;GET THREE COPIES
143 1456 010046          MOV     R0,-(SP)
144 1460 010146          MOV     R1,-(SP)
145 1462 010046          MOV     R0,-(SP)
146 1464 004467          JSR     R4,$POLSH
          000154
147 1470 017162'         .WORD   $MLR    ;GET ARG**2
148 1472 021550'         .WORD   PLY$39  ;SET UP COEFFICIENTS
149 1474 017162'         .WORD   $MLR,$ADR,$MLR,$ADR,$MLR,$ADR
    1476 002010'
    1500 017162'
    1502 002010'
    1504 017162'
    1506 002010'
150 1510 017162'         .WORD   $MLR,$ADR,$MLR,$ADR
    1512 002010'
    1514 017162'
    1516 002010'
151 1520 002010'         .WORD   $ADR    ;P(X)=0 IF X<=1, P(X)=PI/2 IF X>1
152 1522 021604'         .WORD   SGN$39  ;ADJUST SIGN
153 1524 002010'         .WORD   $ADR    ;ADD ATAN2 BIAS
154 1526 017610'         .WORD   $POPR3,EXI$39   ;POP RESULT TO REGS
    1530 021532'
155 1532 005726 EXI$39:  TST     (SP)+   ;POP SIGN FLAG
156 1534 000205          RTS     R5      ;RETURN TO USER
```

```
157                     ;
158 1536 012666 UP$39:  MOV     (SP)+,10.(SP)   ;MOVE STACK ITEM UP
         000012
159 1542 012666         MOV     (SP)+,10.(SP)
         000012
160 1546 000134         JMP     @(R4)+
161                     ;
162 1550 012600 PLY$39: MOV     (SP)+,R0        ;POP POLY ARG
163 1552 012601         MOV     (SP)+,R1
164 1554 012702         MOV     #CON$39+4,R2    ;POINT TO COEFFICIENT TABLE
         021644'
165 1560 012703         MOV     #5,R3   ;LOOP 5
         000005
166 1564 000402         BR      PY1$39
167 1566 010146 PY2$39: MOV     R1,-(SP)        ;PUSH ARG
168 1570 010046         MOV     R0,-(SP)
169 1572 014246 PY1$39: MOV     -(R2),-(SP)     ;PUSH CONSTANT
170 1574 014246         MOV     -(R2),-(SP)
171 1576 005303         DEC     R3      ;COUNT
172 1600 003372         BGT     PY2$39
173 1602 000134         JMP     @(R4)+
174                     ;
175 1604 005766 SGN$39: TST     8.(SP)  ;CHECK SIGN FLAG
         000010
176 1610 001402         BEQ     SG1$39
177 1612 062716         ADD     #100000,@SP     ;NEGATE RESULT FOR (-1,0) & (1,I
         100000
178 1616 000134 SG1$39: JMP     @(R4)+
179                     .ENDC
180                     ;
181                     .IFOF   FPU
182                 ATAN2:  SETF    ;               SET FP MODE FOR FPU
183                     MOV     2(R5),R3;           ADDRESS OF ARG1
184                     MOV     4(R5),R4;           ADDRESS OF ARG2
185                     MOV     @R3,R0;             HIGH ORDER ARG1
186                     MOV     @R4,R1;             HIGH ORDER ARG2
187                     BEQ     INF$39;             JUMP IF DENOMINATOR 0
188                     ASL     R0;
189                     CLRB    R0;
190                     SWAB    R0;                 EXPONENT OF ARG1
191                     ASL     R1;
192                     CLRB    R1;
193                     SWAB    R1;                 EXPONENT OF ARG2
194                     SUB     R1,R0;              GET EXPONENT DIFFERENCE
195                     CMP     #26.,R0;                    CHECK MAGNITUDE
196                     BLT     INF$39;             TREAT AS INFINITE
197                     LDF     PI$39,F3;                   INITIALIZE BIAS=PI
198                     LDF     @R3,F0;             GET ARG1
199                     CFCC
200                     BGE     A1P$39;             JUMP IF ARG1>0
201                     NEGF    F3;                 BIAS=SIGN(ARG1)*PI
202                 A1P$39: LDF     @R4,F1;         GET ARG2
203                     CFCC
204                     BLT     A2M$39;
205                     CLRF    F3;                 IF ARG2>0, BIAS=0
206                 A2M$39: DIVF    F1,F0;          ARG1/ARG2, SET FLOAT CC
207                     BR      AT1$39;             JOIN MAIN ROUTINE
```

```
208                ;
209     INF$39:    LDF     PI2$39,F1;          RESULT=SIGN(ARG1)*PI/2
210                TST     @R3;                TEST ARG1
211                BGE     EXI$39;             +PI/2
212                NEGF    F1;                 -PI/2
213                BR      EXI$39;
214                ;
215     ATAN:      SETF    ;                   SET FP MODE FOR FPU
216                CLRF    F3;                 CLEAR ATAN2 BIAS
217                LDF     @2(R5),F0;          GET ARGUMENT
218     AT1$39:    CLR     R4;                 CLEAR SIGN FLAG
219                CFCC    ;                   GET SIGN OF ARGUMENT
220                STF     F3,F5;              F5=ATAN2 BIAS
221                CLRF    F3;                 CLEAR QUADRANT BIAS
222                BGE     PLU$39;             JUMP IF QUADRANT 1 OR 3
223                ABSF    F0;                 ABS(X)
224                INC     R4;                 FLAG -
225     PLU$39:    LDF     #1.0,F1;            1.0
226                CMPF    F0,F1;              CHECK IF X<=1.0
227                CFCC
228                BLE     LE1$39;
229     GT1$39:    DEC     R4;                 X>1.0, ADJUST SIGN FLAG
230                DIVF    F0,F1;              1.0/X
231                LDF     F1,F0;              ATAN(X)=PI/2-ATAN(1/X)
232                LDF     PI2$39,F3;                  QUADRANT BIAS=PI/2
233                ;
234     LE1$39:    STF     F3,F4;              F4=QUADRANT BIAS
235                CLRF    F3;                 F3=0.0
236                CMPF    T15$39,F0;          COMPARE TAN(15) : X
237                CFCC
238                BGE     L15$39;             X<= TAN(15)
239                LDF     PI6$39,F3;                  F3=PI/6
240                LDF     F0,F1;
241                MULF    RT3$39,F0;
242                SUBF    #1.0,F0;            X*ROOT3-1.0
243                ADDF    RT3$39,F1;          X+ROOT3
244                DIVF    F1,F0;              (X*ROOT3-1.0)/(X+ROOT3)
245                ;
246     L15$39:    LDF     F0,F2;              X
247                MULF    F0,F0;              X**2
248                MOV     #FC0$39,R0;         POINTER TO POLYNOMIAL CONSTANTS
249                MOV     #4,R1;              COUNT OF COEFFICIENTS
250                LDF     (R0)+,F1;           INITIALIZE ACCUMULATOR
251     XPD$39:    MULF    F0,F1;
252                DEC     R1;                 COUNT
253                ADDF    (R0)+,F1;           F1:= F1* X**2 + C(I)
254                BGT     XPD$39; LOOP
255                MULF    F2,F1;              F1:= F1*X
256                ADDF    F3,F1;              PI/6 OR 0.0
257                SUBF    F4,F1;              P(X)-QUAD BIAS
258                TST     R4;                 TEST SIGN FLAG
259                BEQ     SG1$39;             NO ADJUSTMENT
260                NEGF    F1;                 NEGATE RESULT FOR (-1,0)&(1,INF)
261     SG1$39:    ADDF    F5,F1;              ATAN2 BIAS
262                ;
263     EXI$39:    STF     F1,-(SP);           MOVE RESULT TO STACK
264                MOV     (SP)+,R0;           AND THEN TO REGISTERS
```

```
265                             MOV     (SP)+,R1;
266                             RTS     R5;                     EXIT
267                     ;
268                     PI$39:  .WORD   040511,007733;          PI
269                     PI2$39: .WORD   040311,007733;          PI/2
270                     T15$39: .WORD   037611,030243;          TAN(15)
271                     PI6$39: .WORD   040006,005222;          PI/6
272                     RT3$39: .WORD   040335,131727;          ROOT3
273                             .ENDC
274 1620 037305 FCO$39: .WORD   037305,035302   ;.0963034789
    1622 035302
275 1624 137421         .WORD   137421,056514   ;-.1419574624
    1626 056514
276 1630 037514         .WORD   037514,143333   ;.1999773201
    1632 143333
277 1634 137652         .WORD   137652,125244   ;-.3333331319
    1636 125244
278 1640 040200 CON$39: .WORD   040200,000000   ;.9999999999
    1642 000000
279                     ;
280                             .ENDC
```

```
                        .TITLE  $POL07
                        .GLOBL  $POLSH
                ;
                ; $POLSH - IS CALLED WHENEVER IT IS DESIRED TO ENTER POLISH
                ;       MODE FROM IN-LINE CODE.
                ;       IT MUST BE CALLED VIA A JSR R4,$POLSH
                ;
                ;       $POLSH  V007A
                ;
                ; COPYRIGHT 1971, 1972, DIGITAL EQUIPMENT CORPORATION, MAYNARD,M
                ;
        000004  R4=%4
        000006  SP=%6

                ;
                ; THE FOLLOWING ENTRY POINT IS A DIRTY TRICK TO DEFINE
                ;       THE OTS VERSION NUMBER IN THE OTS LINK MAP TO ALLOW
                ;       THE MAINTAINER TO TELL WHAT VERSION IS BEING USED FOR
                ;       A PARTICULAR LINK.
                ;
                ;       THUS WHEN A USER FORGETS TO TELL US WHAT VERSION HE
                ;       IS USING, ALL IS NOT LOST!
                ;
                        .GLOBL  $V20A   ;THIS IS VERSION 20A
 21644          $V20A:
                ;
 21644 005726   $POLSH: TST     (SP)+           ;DELETE JUNK FROM STACK
 21646 000134           JMP     @(R4)+          ;WE'RE NOW IN POLISH MODE
                ;
                ; THE FOLLOWING CONDITIONALIZED ENTRY POINTS IDENTIFY THE ASSEMB
                ;       OPTIONS USED IN THIS VERSION OF THE OTS.
                        .IFDF   EAE
                        .GLOBL  $EAE
                $EAE:
                        .ENDC
                        .IFDF   EIS!MULDIV
                        .GLOBL  $EIS
                $EIS:
                        .ENDC
                        .IFDF   FPU
                        .GLOBL  $FPU
                $FPU:
                        .ENDC
                        .IFDF   FIS

                        .GLOBL  $FIS
                $FIS:
                        .ENDC
                        .IFDF   RSX
                        .GLOBL  $RSX
                $RSX:
                        .ENDC
```

```
1                               .TITLE  $SQT03
2                               .IFDF   CND$41
3                       ;
4                       ;       SQRT    V003A
5                       ;
6                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
7                       ;
8
9                               .GLOBL  SQRT,$ERR;
10                              .IFNDF  FPU
11                              .GLOBL  $ADR,$DVR,$POLSH;
12                              .ENDC
13                      ;       SQRT    THE REAL SQUARE ROOT FUNCTION
14                      ;       CALLING SEQUENCE:
15                      ;       JSR     R5,SQRT
16                      ;       BR      A
17                      ;       #ARG
18                      ;A:
19                      ;       RETURNS THE SQUARE ROOT IN R0 AND R1.
20                      ;
21              000000          R0=%0
22              000001          R1=%1
23              000004          R4=%4
24              000005          R5=%5
25              000006          SP=%6
26              000000          F0=%0
27              000001          F1=%1
28              000002          F2=%2
29                              .IFDF   FPU
30                      SQRT:   MOV     @2(R5),R1;      GET HIGH ORDER ARGUMENT
31                              .ENDC
32                              .IFNDF  FPU
33 21650 010546         SQRT:   MOV     R5,-(SP)
34 21652 016505                 MOV     2(R5),R5        ;GET ARGUMENT ADDRESS
         000002
35 21656 011501                 MOV     @R5,R1  ;GET HIGH ORDER ARGUMENT
36                              .ENDC
37 21660 100443                 BMI     ERR$41  ;ERROR IF ARGUMENT NEGATIVE
38 21662 001446                 BEQ     ZER$41  ;FAST EXIT IF ZERO
39                              .IFNDF  FPU
40 21664 012746                 MOV     #3,-(SP)        ;PUSH ITERATION COUNT
         000003
41                              .ENDC
42 21670 006201                 ASR     R1      ;FORM INITIAL ESTIMATE
43 21672 062701                 ADD     #20100,R1
         020100
44 21676 005046                 CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
45 21700 010146                 MOV     R1,-(SP)        ;'CAUSE ADD AND DIVIDE ARE
46                              .IFNDF  FPU
47 21702 005046                 CLR     -(SP)   ;FASTER THAT WAY
48 21704 011546                 MOV     @R5,-(SP)
49 21706 005046                 CLR     -(SP)
50 21710 010146                 MOV     R1,-(SP)
51 21712 004467         LUP$41: JSR     R4,$POLSH       ;ENTER POLISH MODE
         177726
52 21716 013256'                .WORD   $DVR,$ADR,UPL$41        ;(X/E+E)
   21720 002010'
```

```
   21722 021724'
53 21724 162716 UPL$41: SUB     #200,@SP        ;(X/E+E)/2
         000200
54 21730 005366         DEC     4(SP)   ;COUNT LOOP
         000004
55 21734 001410         BEQ     OUT$41
56 21736 016546         MOV     2(R5),-(SP)     ;USE LOW ORDER PARTS
         000002
57 21742 011546         MOV     @R5,-(SP)       ;TOO FROM NOW ON
58 21744 016646         MOV     6(SP),-(SP)
         000006
59 21750 016646         MOV     6(SP),-(SP)
         000006
60 21754 000756         BR      LUP$41  ;GO FOR ANOTHER ITERATION
61 21756 012600 OUT$41: MOV     (SP)+,R0        ;GET RESULT INTO R0,R1
62 21760 012601         MOV     (SP)+,R1
63 21762 005726         TST     (SP)+   ;POP ITERATION COUNTER
64 21764 012605 RTN$41: MOV     (SP)+,R5
65 21766 000205         RTS     R5      ;RETURN TO CALLER
66 21770 004567 ERR$41: JSR     R5,$ERR ;ERROR 4,11
         000012
67 21774 000773         BR      RTN$41
68 21776     004        .BYTE   4
69 21777     013        .BYTE   11.
70 22000 005000 ZER$41: CLR     R0
71 22002 005001         CLR     R1
72 22004 000767         BR      RTN$41
73                      .ENDC
74                      .IFDF   FPU
75                      MOV     #3,R0;          ITERATION COUNT
76                      SETF    ;               SINGLE PRECISION FP
77                      LDF     (SP)+,F0;       GET INITIAL ESTIMATE
78                      LDF     @2(R5),F2;      GET X
79                ;
80                LUP$41: LDF     F0,F1;          E=E'
81                      LDF     F2,F0;          X
82                      DIVF    F1,F0;          X/E
83                      ADDF    F1,F0;          X/E+E
84                      DEC     R0;             COUNT
85                      DIVF    #2.0,F0;        E'=(X/E+E)/2
86                      BGT     LUP$41;
87                ;
88                      STF     F0,-(SP);               RESULT TO STACK
89                      MOV     (SP)+,R0;       AND THENCE TO R0,R1
90                      MOV     (SP)+,R1;
91                      RTS     R5;             EXIT
92                ;
93                ERR$41: JSR     R5,$ERR;        ERROR 4,11
94                      RTS     R5;             EXIT
95                      .BYTE   4
96                      .BYTE   11.
97                ZER$41: CLR     R0;
98                      CLR     R1;
99                      RTS     R5;
100                     .ENDC
101                     .ENDC
```

155

```
                        .TITLE  $ERR01
                        .GLOBL  $ERR,$ERRA,$ERVEC
                R0 = %0
                R5 = %5
                SP = %6
                PC = %7

                ;       THE ERROR HANDLER OF FPMP-11
                ;       THIS ROUTINE PASSES CONTROL TO THE USER'S ERROR
                ;       ROUTINE, IF ANY.  DEFAULT ACTION IS TO HALT.
                ;       USER MUST MOVE ADDRESS OF HIS ERROR ROUTINE
                ;       TO GLOBAL LOCATION '$ERVEC'.  CONTROL IS PASSED
                ;       TO THE ADDRESS IN $ERVEC VIA A   JSR PC,@$ERVEC.
                ;       REGISTER ZERO WILL CONTAIN THE ERROR CODE.

                ;       CALLING SEQUENCE:

                ;               JSR     R5,$ERR
                ;               BR      A
                ;               .BYTE   ERROR CLASS
                ;               .BYTE   ERROR NUMBER
                ;       A:

                ;       OR:

                ;               MOV     #ERRNUM,R0
                ;               JSR     R5,$ERRA

22006 010046    $ERR:   MOV     R0,-(SP)        ;       SAVE R0
22010 016500    000002  MOV     2(R5),R0        ;       GET ERROR CLASS/NUMBER
22014 000401            BR      ERR$43
22016 010046    $ERRA:  MOV     R0,-(SP)        ;       SAVE R0
22020           ERR$43: .IFNDF  CLASS5          ;       DEFINE TO GET WARNINGS
22020 120027    000005  CMPB    R0,#5           ;       CLASS 5 (WARNING)?
22024 001402            BEQ     IGN$43          ;       IGNORE IF SO
                        .ENDC
22026 004777    000004  JSR     PC,@$ERVEC      ;       CALL USER ERR ROUTINE
22032 012000    IGN$43: MOV     (SP)+,R0        ;       RESTORE R0
22034 000205            RTS     R5              ;       RETURN TO ERROR ROUTINE
22036 022040'   $ERVEC: .WORD   HLT$43          ;       ADDR OF USER ERR ROUTINE
22040 000000    HLT$43: HALT                    ;       DEFAULT: HALT
22042 000776            BR      HLT$43          ;       HARD STOP
```

```
1                               .TITLE  $LOR01
2                               .IFDF   CND$44&CND$42
3               000004          R4=%4
4               000006          SP=%6
5
6                       ;       LOAD FLAC = SINGLE PRECISION
7
8  022044 012666 $LOR1  MOV     (SP)+,2(SP);    MOVE OPERAND TO RESULT LOC
          000002
9  022050 012666        MOV     (SP)+,2(SP)
          000002
10 22054  000134        JMP     @(R4)+;         POLISH RETURN
11                              .ENDC
```

```
1                               .TITLE  $LDD01
2                               .IFDF   CND$45&CND$42
3               000000          R0=%0
4               000004          R4=%4
5               000006          SP=%6
6
7                       ;       LOAD FLAC = DOUBLE PRECISION
8
9  022056 010600 $LDD:  MOV     SP,R0;          COPY STACK POINTER
10 22060 062700         ADD     #8.,R0;         CALC ADDR OF RESULT
           000010
11 22064 012620         MOV     (SP)+,(R0)+;    MOVE OPRAND TO RESULT LOC
12 22066 012620         MOV     (SP)+,(R0)+
13 22070 012620         MOV     (SP)+,(R0)+
14 22072 012620         MOV     (SP)+,(R0)+
15 22074 000134         JMP     @(R4)+;         POLISH RETURN
16                              .ENDC
```

138

```
                        .TITLE  $STR01
                        .IFDF   CND$46&CND$42
        000000          R0=%0
        000001          R1=%1
        000002          R2=%2
        000003          R3=%3
        000004          R4=%4
        000005          R5=%5
        000006          SP=%6
        000007          PC=%7

                ;       STORE FLAC = SINGLE PRECISION

22076 012705 $STR:      MOV     #FAC$42,R5;     GET ADDRESS OF FLAC
        000432'
22102 005766            TST     30(SP);         TEST FOR STACK MODE
        000030
22106 001415            BEQ     STK$46;         BRANCH IF NOT
22110 005066            CLR     30(SP);         CLEAR STACK MODE FLAG
        000030
22114 010600            MOV     SP,R0;          COPY STACK POINTER
22116 010601            MOV     SP,R1
22120 022121            CMP     (R1)+,(R1)+;    R1 = R1 + 4
22122 012702            MOV     #15,R2;         LOOP COUNT
        000013
22126 012120 LP$46:     MOV     (R1)+,(R0)+;    MOVE UP STACK TO MAKE ROOM
22130 005302            DEC     R2
22132 001375            BNE     LP$46
                ;       R0 POINTS TO OPERAND LOCATION
22134 012520            MOV     (R5)+,(R0)+;    STORE THE FLAC
22136 012520            MOV     (R5)+,(R0)+
22140 000134            JMP     @(R4)+;         POLISH RETURN
22142 012520 STK$46:    MOV     (R5)+,(R0)+;    STORE THE FLAC
22144 012520            MOV     (R5)+,(R0)+
22146 022626            CMP     (SP)+,(SP)+;    POP OPERAND OFF THE STACK
22150 000134            JMP     @(R4)+
                        .ENDC
```

```
                        .TITLE  $STD01
                        .IFDF   CND$47&CND$42
000000                  R0=%0
000001                  R1=%1
000002                  R2=%2
000003                  R3=%3
000004                  R4=%4
000005                  R5=%5
000006                  SP=%6
000007                  PC=%7

22152 012705 $STD:      MOV     #FAC$42,R5
      000432'
22156 005766            TST     34(SP);         TEST FOR STACK MODE
      000034
22162 001420            BEQ     STK$47;         BRANCH IF NOT
22164 005066            CLR     34(SP)
      000034
22170 010500            MOV     SP,R0
22172 010501            MOV     SP,R1
22174 062701            ADD     #10,R1
      000010
22200 012702            MOV     #15,R2
      000013
22204 012120 LP$47:     MOV     (R1)+,(R0)+
22206 005302            DEC     R2
22210 001375            BNE     LP$47
22212 012520            MOV     (R5)+,(R0)+;    STORE THE FLAC
22214 012520            MOV     (R5)+,(R0)+
22216 012520            MOV     (R5)+,(R0)+
22220 012520            MOV     (R5)+,(R0)+
22222 000134            JMP     @(R4)+;         RETURN
22224 012520 STK$47:    MOV     (R5)+,(R0)+
22226 012520            MOV     (R5)+,(R0)+
22230 012520            MOV     (R5)+,(R0)+
22232 012520            MOV     (R5)+,(R0)+
22234 062706            ADD     #10,SP;         POP OPERAND
      000010
22240 000134            JMP     @(R4)+
                        .ENDC
                        .TITLE  FPMP11 FLOATING POINT & MATH PACKAGE

000001'                 .END
```

SYMBOL TABLE

```
A      = 000010          ABP$19  014432R          AC     = 177302
ADJ$19  014344R          ADR$42  000202R          AINT    003124RG
AI1$4   003150R          ALOG    002550RG         ALOG10  002544RG
ARN$19  014354R          ASH    = 177316          ASL$11  007574R
ASL$4   003216R          ASR$19  014254R          AS1$11  007552R
ATAN    02116ØRG         ATAN2   020706RG         ATE$15  011072R
ATE$39  021124R          AT1$15  011204R          AT1$39  021204R
AUP$19  014410R          A1     = 000004          A1Z$1   001046R
A2     = 000010          A2I$8   003666R          A2N$1   001054R
A2N$2   002106R          B      = 000014          BAC$25  015634R
BEXP   = 000004          BT9$1   001666R          BT9$2   002464R
B1     = 000006          B2     = 000012          B2N$28  016404R
B2N$30  017264R          B2Z$28  016410R          B2Z$30  017300R
B4N$28  016346R          B4Z$28  016302R          B6Z$28  016332R
B9A$1   001666R          B9A$2   002464R          CAD$42  000476R
CAR$42  000452R          CFR$20  014752R          CHK$17  013246R
CLE$9   005144R          CLR$29  017050R          CMD$42  000466R
CMF$42  000254R          CMR$42  000442R          CM1$42  000264R
CND$1  = 000001          CND$10= 000001           CND$11= 000001
CND$12= 000001           CND$13= 000001           CND$14= 000001
CND$15= 000001           CND$16= 000001           CND$17= 000001
CND$18= 000001           CND$19= 000001           CND$2  = 000001
CND$20= 000001           CND$21= 000001           CND$22= 000001
CND$23= 000001           CND$24= 000001           CND$25= 000001
CND$26= 000001           CND$27= 000001           CND$28= 000001
CND$29= 000001           CND$3  = 000001          CND$30= 000001
CND$31= 000001           CND$32= 000001           CND$33= 000001
CND$34= 000001           CND$35= 000001           CND$36= 000001
CND$37= 000001           CND$38= 000001           CND$39= 000001
CND$4  = 000001          CND$41= 000001           CND$42= 000001
CND$44= 000001           CND$45= 000001           CND$46= 000001
CND$47= 000001           CND$5  = 000001          CND$6  = 000001
CND$7  = 000001          CND$8  = 000001          CND$9  = 000001
CNV$8   003444R          CON$10  007440R          CON$13  010444R
CON$15  012200R          CON$3   003120R          CON$37  020400R
CON$39  02164ØR          COS     02003ØRG         COV$30  017506R
CYC$29  017052R          C1     = 000012          C2     = 000022
C23$11  007514R          D      = 000010          DATAN   011136RG
DATAN2  010666RG         DBLE    003416RG         DCH$16  012506R
DCH$18  013442R          DCOS    007674RG         DCR$25  015666R
DEC$25  015772R          DEC$35  017724R          DEXP    013710RG
DGS$9   006552R          DG1$9   006570R          DG2$9   006602R
DG3$9   006610R          DHI$16  012504R          DHI$18  013514R
DIGITS= 000004           DIG$9   006632R          DIV$15  011010R
DIV$17  013176R          DIV$39  021056R          DIV$8   004310R
DIV$9   005306R          DLOG    006654RG         DLOG10  006650RG
DLW$16  012552R          DLW$18  013506R          DMODE  = 100000
DNE$11  007604R          DNE$24  015456R          DNE$25  015724R
DNE$35  017730R          DNE$4   003226R          DNE$9   006022R
DN1$4   003234R          DOUBLE= 000001           DPR$8   004610R
DR1$12  007640R          DR2$12  007606R          DSIN    007752RG
DSQRT   010454RG         DSV$19  014402R          DUP$10  007246R
DUP$13  010154R          DUP$19  014454R          DUP$20  014720R
DUP$3   003014R          DUP$37  020222R          DV1$16  012772R
DV1$18  013630R          DV1$8   004326R          DV1$9   005344R
DV2$8   004314R          DV2$9   005350R          DV3$8   004346R
DV4$8   004366R          D1     = 000014          D10$8   004024R
```

SYMBOL TABLE

```
D16$19  014370R     D2     = 000024     D6R$19  014400R
ECK$1   001232R     ECK$2   002212R     ECL$16  012520R
ECL$18  013466R     ECL$19  014100R     ECL$20  015112R
ECL$28  016620R     ECL$30  017446R     EC1$16  012522R
EC1$18  013474R     EEXP   = 000006     EFE$9   005734R
EFT$9   005646R     EF1$8   004176R     EF2$8   004204R
END    = 000044     ENM$8   004212R     EN1$8   004224R
EN2$8   004274R     EQ1$16  013120R     EQ1$18  013700R
EQ2$16  013116R     EQ2$18  013676R     ERB$43  022020R
ERF    = 000052     ERO$10  007352R     ERR$10  007130R
ERR$14  010644R     ERR$24  015474R     ERR$25  015740R
ERR$3   003070R     ERR$41  021770R     ERR$42  000422R
ERR$8   004074R     ERR$9   006276R     ESIGN  = 000010
ESV$20  014744R     EXA$1   001122R     EXA$2   002144R
EXC$8   004114R     EXI$10  007312R     EXI$15  011770R
EXI$3   003040R     EXI$39  021532R     EXP     014554RG
EXP$9   005770R     EXT$8   004130R     EX1$9   005774R
EX2$9   006012R     EX3$9   006000R     FAC$42  000432R
FCO$15  012100R     FCO$39  021620R     FFE$9   005464R
FFT$9   005460R     FF3$9   005600R     FF4$9   005564R
FF5$9   005544R     FF6$9   005626R     FIL$25  015700R
FLD$24  015436R     FLD$8   003712R     FLOAT   015202RG
FLT$16  012660R     FLT$18  013550R     FLT$8   004454R
FL1$16  012656R     FPS$5   003202R     FPS$6   003360R
FUL$25  015712R     F0     =%000000     F1     =%000001
F2     =%000002     F3     =%000003     F4     =%000004
F5     =%000005     GO$16   013046R     GO$18   013654R
GO$24   015234R     GO$25   015540R     GT1$15  011250R
GT1$39  021234R     HLT$43  022040R     IDINT   016020RG
IFIX    015154RG    IGN$43  022052R     INC$20  014712R
INF$15  011076R     INF$39  021130R     INR$15  011134R
INR$39  021156R     INT     016020RG    ISN$9   006506R
ISR$9   006526R     L      = 000024     LENGTH= 000042
LE1$15  011356R     LE1$39  021306R     LFT$8   005010R
LGT$10  007342R     LGT$3   003000R     LOG$10  006656R
LOG$3   002552R     LPS$25  015562R     LP$46   022126R
LP$47   022204R     LSH    = 177314     LUP$14  010542R
LUP$17  013216R     LUP$41  021712R     L15$15  011642R
L15$39  021446R     MDV$8   004036R     MLT$28  016640R
MLT$30  017460R     ML1$9   005204R     ML5$8   004742R
ML8$9   006246R     MOV$25  015656R     MQ     = 177304
MT0$28  016736R     MT0$30  017514R     MT1$28  016644R
MT1$30  017464R     MT2$28  016732R     MUL$29  017052R
MUL$8   004006R     MUL$9   005240R     M16$19  014362R
M45$9   006136R     M5A$8   004702R     M51$9   006156R
M52$9   006172R     M54$8   004674R     M54$9   006050R
M55$8   004714R     M81$9   006254R     N      = 000014
NAD$42  000334R     NCK$24  015306R     NCK$8   003614R
NCP$2   002254R     NEG$42  000252R     NEG$5   003326R
NEG$6   003404R     NFL$1   001502R     NGM$24  015476R
NGM$29  017132R     NGO$16  013104R     NGO$18  013664R
NNZ$18  013664R     NNZ$28  016272R     NNZ$8   003644R
NOD$1   001476R     NOD$2   002372R     NOD$27  016120R
NOD$9   005402R     NOM$27  016110R     NOM$28  016460R
NOM$30  017336R     NOM$9   005356R     NOP$8   003754R
NOR    = 177312     NO1$9   005360R     NQD$16  013112R
NQD$18  013672R
```

```
NT0$18  013540R         NUMEND= 000000          NXT$24  015346R
NXT$8   003574R         NZE$25  015646R         OCL$24  015506R
OCT$25  016004R         ONE$19  014060R         ONE$20  015070R
ONE$38  020656R         OT2$42  000170R         OUT$1   001552R
OUT$14  010626R         OUT$19  014340R         OUT$2   002426R
OUT$28  016552R         OUT$29  017126R         OUT$30  017404R
OUT$35  017744R         OUT$38  020570R         OUT$41  021756R
OUT$5   003322R         OUT$6   003400R         OVF$1   001564R
OVR$1   001566R         OVR$12  007652R         OVR$16  012764R
OVR$18  013454R         OVR$19  014056R         OVR$2   002436R
OVR$20  015100R         OVR$28  016606R         OVR$29  017150R
OVR$30  017430R         OVR$31  017554R         OVR$35  017754R
OVR$36  020020R         OVR$8   004670R         OV1$12  007656R
OV1$16  012762R         OV1$18  013452R         OV1$28  016604R
OV1$30  017432R         P       = 000020        PC      =%000007
PCK$8   004052R         PLE$20  014732R         PLM$42  000354R
PLS$42  000252R         PLS$5   003330R         PLS$6   003406R
PLU$15  011216R         PLU$39  021216R         PLY$13  010300R
PLY$15  012020R         PLY$37  020324R         PLY$39  021550R
PL1$42  000372R         PL2$10  007270R         PL2$3   003026R
PMODE   = 040000        PNT$8   003766R         POINT   = 000002
POINTL= 000002          POS$19  013732R         POS$20  014574R
POS$25  015612R         POS$27  016076R         PR4$13  010134R
PTF$8   004060R         PT$42   000076R         PY1$13  010322R
PY1$15  012042R         PY1$37  020346R         PY1$39  021572R
PY2$13  010312R         PY2$15  012032R         PY2$37  020342R
PY2$39  021566R         P1$17   013144R         P1$29   017016R
P15$38  020722R         P2$17   013156R         P2$29   017030R
P3$17   013236R         P35$38  020734R         P45$38  020712R
Q       = 000014        QSE$13  010060R         QSE$37  020150R
QSR$13  010260R         QSR$37  020304R         QST$13  010224R
QST$37  020254R         QUD$13  010216R         QUD$37  020246R
QUT$13  010276R         QUT$37  020322R         Q13$13  010264R
Q13$37  020310R         RDD$9   006432R         RDF$9   006402R
REG$10  007144R         REG$3   002742R         REL$25  015616R
RESLT   = 000010        RESULT= 000036          RET$21  015146R
RET$42  000220R         RIT$8   005022R         RIT$9   006634R
RND$22  015166R         ROR$11  007564R         ROR$4   003204R
RRN$8   004664R         RR1$11  007542R         RTI$42  000332R
RTN$13  010140R         RTN$14  010640R         RTN$16  012750R
RTN$18  013620R         RTN$37  020204R         RTN$41  021764R
RTS$16  013126R         RTS$18  013706R         RT1$13  010150R
RT1$37  020220R         RT2$19  014544R         RT2$42  000322R
RUD$9   006340R         RU1$9   006476R         RU2$9   006464R
RU3$9   006500R         R0      =%000000        R1      =%000001
R2      =%000002        R3      =%000003        R4      =%000004
R5      =%000005        S       = 000026        SCK$1   001200R
SCK$2   002176R         SCL$10  007232R         SCL$19  014250R
SCL$20  015046R         SCL$3   003000R         SCL$8   003722R
SCN$8   003524R         SC1$19  014312R         SC1$8   003742R
SFD$1   001414R         SFD$2   002336R         SFL$2   002306R
SFR$1   001266R         SFR$2   002236R         SFT$1   001236R
SFT$2   002216R         SFT$35  017720R         SF0$2   002326R
SF1$1   001304R         SGN$15  012064R         SGN$16  012702R
SGN$18  013572R         SGN$24  015444R         SGN$35  017736R
SGN$39  021604R         SGS$24  015314R         SGS$8   003552R
SG1$15  012076R         SG1$39  021616R         SHF$11  007522R
```

SYMBOL TABLE

```
SHF$4   003202R     SIGN  = 000002     SIGNS = 000000
SIN     020064RG    SINGLE= 000001     SL8$1   001346R
SME$5   003266R     SME$6   003304R    SML$38  020574R
SMT$19  013740R     SMT$20  014602R    SNC$13  010000R
SNC$37  020100R     SNGL    017772RG   SN1$24  015326R
SP    =%000006      SPC$9   006532R    SQRT    021650RG
SR    = 177311      SRL$2   002274R    SR$29 = 177311
SR8$1   001400R     START = 000044     STC$10  007166R
STC$3   002760R     STE$38  020452R    STK$10  007156R
STK$3   002754R     STK$42  000122R    STK$46  022142R
STK$47  022224R     STM$42  000376R    STR$2   002416R
STR$8   004564R     STS$25  015752R    STS$9   006322R
STT$24  015270R     ST4$42  000140R    ST6$42  000160R
SUB$1   001630R     SUB$2   002446R    SUB$25  015624R
S16$1   001334R     S8A$1   001376R    TANH    020404RG
TAN$38  020476R     TBL$42  000500R    TEMP  = 000042
TNS$15  011462R     TNS$39  021346R    TRAPH   000000RG
TST$25  015616R     TWC$19  014476R    TW1$19  014502R
TYPE  = 000014      UND$1   001612R    UND$16  012514R
UND$18  013462R     UND$2   002536R    UND$28  015614R
UND$30  017440R     UND$8   004670R    UNF$1   001602R
UNF$2   002526R     UPC$42  000116R    UPL$14  010554R
UPL$15  011020R     UPL$19  014116R    UPL$22  015176R
UPL$23  015220R     UPL$26  016042R    UPL$38  020564R
UPL$39  021066R     UPL$41  021724R    UP$10   007210R
UP$15   011776R     UP$3    002766R    UP$38   020754R
UP$39   021536R     UTS$1   001576R    UTS$2   002522R
XCO$9   005102R     XC1$9   005122R    XPD$10  007054R
XPD$13  010072R     XPD$15  011712R    XSQ$38  020620R
X0$28   016710R     X0$30   017474R    X00$28  015760R
X00$30  017522R     X4$13   010176R    X4$37   020234R
ZER$1   001622R     ZER$13  010210R    ZER$14  010654R
ZER$16  012526R     ZER$17  013242R    ZER$18  013436R
ZER$19  014074R     ZER$2   002540R    ZER$20  015106R
ZER$27  016144R     ZER$28  016264R    ZER$29  017144R
ZER$30  017452R     ZER$31  017552R    ZER$35  017766R
ZER$38  020746R     ZER$41  022000R    ZER$8   004100R
ZE1$28  016266R     ZE1$30  017424R    ZE2$28  016626R
ZFR$20  015044R     ZTS$1   001710R    ZTS$2   002502R
ZT1$1   001760R     ZT2$1   001754R    Z1$19   014106R
$ADD    000704RG    $ADR    002010RG   $CMD    003242RG
$CMR    003340RG    $DCI    003442RG   $DCO    005046RG
$DI     017640RG    $DINT   007450RG   $DR     007622RG
$DVD    012210RG    $DVI    013130RG   $DVR    013256RG
$ECO    005076RG    $ERR    022006RG   $ERRA   022016RG
$ERVEC  022036RG    $FCALL  015124RG   $FCO    005042RG
$GCO    005034RG    $ICI    015230RG   $ICO    015534RG
$ID     016046RG    $INTR   003142RG   $IR     016062RG
$LDD    022056R     $LDR    022044R    $MLD    016146RG
$MLI    017002RG    $MLR    017162RG   $NGD    017542RG
$NGI    017534RG    $NGR    017542RG   $OCI    015222RG
$OCO    015526RG    $POLSH  021644RG   $POPR3  017610RG
$POPR4  017576RG    $POPR5  017576RG   $PSHR1  017572RG
$PSHR2  017572RG    $PSHR3  017570RG   $PSHR4  017564RG
$PSHR5  017564RG    $RCI    003434RG   $RD     017616RG
$RI     017650RG    $SBD    000700RG   $SBR    002004RG
$STD    022152R     $STR    022076R    $V20A   021644RG
```

SYMBOL TABLE

. ABS. 000000 000
022242 001

ERRORS DETECTED: 0
FREE CORE: 14863. WORDS
.LP:<PR:,DT1:FPMP.MAC

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes software newsletters for the various DIGITAL products. Newsletters are published monthly, and keep the user informed about customer software problems and solutions, new software products, documentation corrections, as well as programming notes and techniques.

There are two similar levels of service:

- The Software Dispatch
- The Digital Software News

The Software Dispatch is part of the Software Maintenance Service. This service applies to the following software products:

PDP-9/15
RSX-11D
DOS/BATCH
RSTS-E
DECsystem-10

A Digital Software News for the PDP-11 and a Digital Software News for the PDP-8/12 are available to any customer who has purchased PDP-11 or PDP-8/12 software.

A collection of existing problems and solutions for a given software system is published periodically. A customer receives this publication with his initial software kit with the delivery of his system. This collection would be either a Software Dispatch Review or Software Performance Summary depending on the system ordered.

A mailing list of users who receive software newsletters is also maintained by Software Communications. Users must sign-up for the newsletter they desire. This can be done by either completing the form supplied with the Review or Summary or by writing to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to DIGITAL's software should be reported as follows:

North and South American Submitters:

Upon completion of Software Performance Report (SPR) form remove last copy and send remainder to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

The acknowledgement copy will be returned along with a blank SPR form upon receipt. The acknowledgement will contain a DIGITAL assigned SPR number. The SPR number or the preprinted number should be referenced in any future correspondence. Additional SPR forms may be obtained from the above address.

All International Submitters:

Upon completion of the SPR form, reserve the last copy and send the remainder to the SPR Center in the nearest DIGITAL office. SPR forms are also available from our SPR Centers.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computers Users Society, maintains a user exchange center for user-written programs and technical application information. The Library contains approximately 1,900 programs for all DIGITAL computer lines. Executive routines, editors, debuggers, special functions, games, maintenance and various other classes of programs are available.

DECUS Program Library Catalogs are routinely updated and contain lists and abstracts of all programs according to computer line:

- PDP-8, FOCAL-8, BASIC-8, PDP-12
- PDP-7/9, 9, 15
- PDP-11, RSTS-11
- PDP-6/10, 10

Forms and information on acquiring and submitting programs to the DECUS Library may be obtained from the DECUS office.

In addition to the catalogs, DECUS also publishes the following:

DECUSCOPE	-The Society's technical newsletter, published bi-monthly, aimed at facilitating the interchange of technical information among users of DIGITAL computers and at disseminating news items concerning the Society. Circulation reached 19,000 in May, 1974.
PROCEEDINGS OF THE DIGITAL EQUIPMENT USERS SOCIETY	-Contains technical papers presented at DECUS Symposia held twice a year in the United States, once a year in Europe, Australia, and Canada.
MINUTES OF THE DECsystem-10 SESSIONS	-A report of the DECsystem-10 sessions held at the two United States DECUS Symposia.
COPY-N-Mail	-A monthly mailed communique among DECsystem-10 users.
LUG/SIG	-Mailing of Local User Group (LUG) and Special Interest Group (SIG) communique, aimed at providing closer communication among users of a specific product or application.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS EUROPE
Digital Equipment Corp. International
(Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

Please cut along this line.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754