DEC- 11-NFPMA-A-D

FPMP-11
USER'S MANUAL

<u>SOFTWARE SUPPORT CATEGORY</u>

The software described in this document is supported by Digital Equipment Corporation under Category I , as defined on page iii of this document.

For additional copies, order No. DEC- 11-NFPMA-A-D from Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754 Price $5.00

First Printing
September, 1972

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.

SOFTWARE SUPPORT CATEGORIES

Digital Equipment Corporation (DEC) makes available four categories of software. These categories reflect the types of support a customer may expect from DEC for a specified software product. DEC reserves the right to change the category of a software product at any time. The four categories are as follows:

CATEGORY I
Software Products Supported at no Charge

This classification includes current versions of monitors, programming languages, and support programs provided by DEC. DEC will provide installation (when applicable), advisory, and remedial support at no charge. These services are limited to original purchasers of DEC computer systems who have the requisite DEC equipment and software products.

At the option of DEC, a software product may be recategorized from Category I to Category II for a particular customer if the software product has been modified by the customer or a third party.

CATEGORY II
Software Products that Receive Support for a Fee

This category includes prior versions of Category I programs and all other programs available from DEC for which support is given. Programming assistance (additional support), as available, will be provided on these DEC programs and non-DEC programs when used in conjunction with these DEC programs and equipment supplied by DEC .

CATEGORY III
Pre-Release Software

DEC may elect to release certain software products to customers in order to facilitate final testing and/or customer familiarization. In this event, DEC will limit the use of such pre-release software to internal, non-competitive applications. Category III software is only supported by DEC where this support is consistent with evaluation of the software product. While DEC will be grateful for the reporting of any criticism and suggestions pertaining to a pre-release, there exists no commitment to respond to these reports.

CATEGORY IV
Non-Supported Software

This category includes all programs for which no support is given

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-GGPC-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 Operating 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

S	exponent	high-order mantissa
15	14 7	6 0

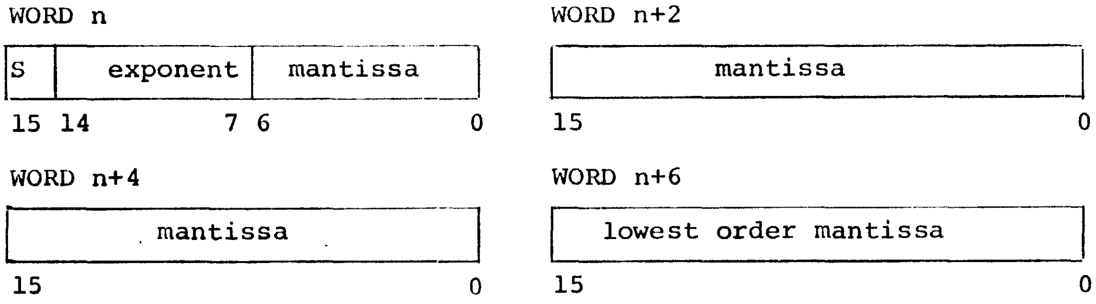
WORD n+2

low-order mantissa
15 0

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
$\sqrt{2}$	$0.10110101... \times 2^{1}$	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
$0.1 \le M < 1$	F-type with p=0,w=W-4 and d=D
$1 \le M < 10$	F-type with p=0,w=W-4 and d=D-1.
$10^{D-2} \le M < 10^{D-1}$	F-type with p=0,w=W-4 and d=1.
$10^{D-1} \le M < 10^{D}$	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:

```
|0 |10000100|0100000|   |0000000000000000|
 15 14     7 6     0     15             0
```

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.
```
 | flags | relative address |
 15      13                 0
```

```
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 number    |     error class    |
 --------------------------------------
15               8 7                  0
```

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-11/45 extended instruction set (EIS)

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)
		$ CRESS
		[illegible]

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
       ØØØØØØ           RØ=%Ø
       ØØØØØ1           R1=%1
       ØØØØØ2           R2=%2
       ØØØØØ3           R3=%3
       ØØØØØ4           R4=%4
       ØØØØØ5           R5=%5
       ØØØØØ6           SP=%6
       ØØØØØ7           PC=%7
       ØØØ1ØØ           ARM=1ØØ
       ØØØ2ØØ           IMM=2ØØ
       ØØØ3ØØ           RELM=3ØØ
       ØØØØ71           LDR=71
       ØØØØ73           STR=73
       ØØØØ12           ADR=12
       ØØØØ13           SBR=13
       ØØØØ21           MLR=21
       ØØØØ25           DVR=25
       ØØØØ46           SQRT=46
       ØØØØ11           MSGLEN=9.
       ØØØØØ2           RESET=2
       ØØØØ11           READOP=11
       ØØØØØ4           WAITR=4
       ØØØØ12           WRITE=12
                        .GLOBL  TRAPH,$RCI,$FCO
ØØØØØØ Ø127Ø6 BEGIN:    MOV     #2ØØØ,SP;       INITIALIZE STACK
       ØØ2ØØØ
ØØØØØ4 ØØØØØ4           IOT
ØØØØØ6 ØØØØØØ           .WORD   Ø;              INIT THE IOX PACKAGE
ØØØØ1Ø    ØØ2           .BYTE   RESET,Ø
ØØØØ11    ØØØ
ØØØØ12 ØØØØØ4           IOT
ØØØØ14 ØØØ5Ø4'          .WORD   TITLE
ØØØØ16    Ø12           .BYTE   WRITE,1;        WRITE THE TITLE
ØØØØ17    ØØ1
ØØØØ2Ø Ø12737'          MOV     #TRAPH,@#34;    INITIALIZE TRAP VECTOR
       ØØØØØØ
       ØØØØ34
ØØØØ26 Ø12737           MOV     #34Ø,@#36
       ØØØ34Ø
       ØØØØ36
ØØØØ34 ØØ4767 RESTAR:   JSR     PC,READ;        READ ONE INPUT LINE INTO BUFFR
       ØØØ374
ØØØØ4Ø Ø127Ø1'          MOV     #BUFFR+6,R1;    GET ADDR OF BEGINNING OF BUFFER
       ØØØ646
ØØØØ44 Ø127ØØ'          MOV     #A,RØ;          GET ADDR OF VAR 'A'
       ØØØ454
```

```
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 WAIT1:       IOT;                        CALL IOX
000444 000442'             .WORD   WAIT1;              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
WAITI      000442R      WAITO     000410R       WAITR   = 000004
WRITE    = 000012       ZERO      000230R       $FCO    = ****** G
$RCI     = ****** G     .       = 001034R
```

```
TEST OF FPMP11
       2.Ø      4.ØØ   2.Ø              ;Teletype input in three
                 2.ØØØØØØØØØØ           ;10-character fields

                 4.ØØØØØØØØØØ
                                        ;program verification
                 2.ØØØØØØØØØØ           ;of input

ROOT =          -1.ØØØØØØØØØØ           ;result

      12.5     3.25     5.43
                12.5ØØØØØØØØØ

                 3.25ØØØØØØØØ

                 5.4299998283


ROOTS ARE IMAGINARY***

    3.E-Ø1   .Ø6EØØ2   4ØE-ØØ1
                 Ø.3ØØØØØØ119

                 6.ØØØØØØØØØØ

                 4.ØØØØØØØØØØ

ROOT 1 =        -Ø.69Ø5Ø62795

ROOT 2 =       -19.3Ø94921112

         5        15          3
                 5.ØØØØØØØØØØ

                15.ØØØØØØØØØØ

                 3.ØØØØØØØØØØ

ROOT 1 =        -Ø.2154767Ø66

ROOT 2 =        -2.7845234871

         Ø     4.Ø        3.75
                 Ø.ØØØØØØØØØØ

                 4.ØØØØØØØØØØ

                 3.75ØØØØØØØØ
```

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

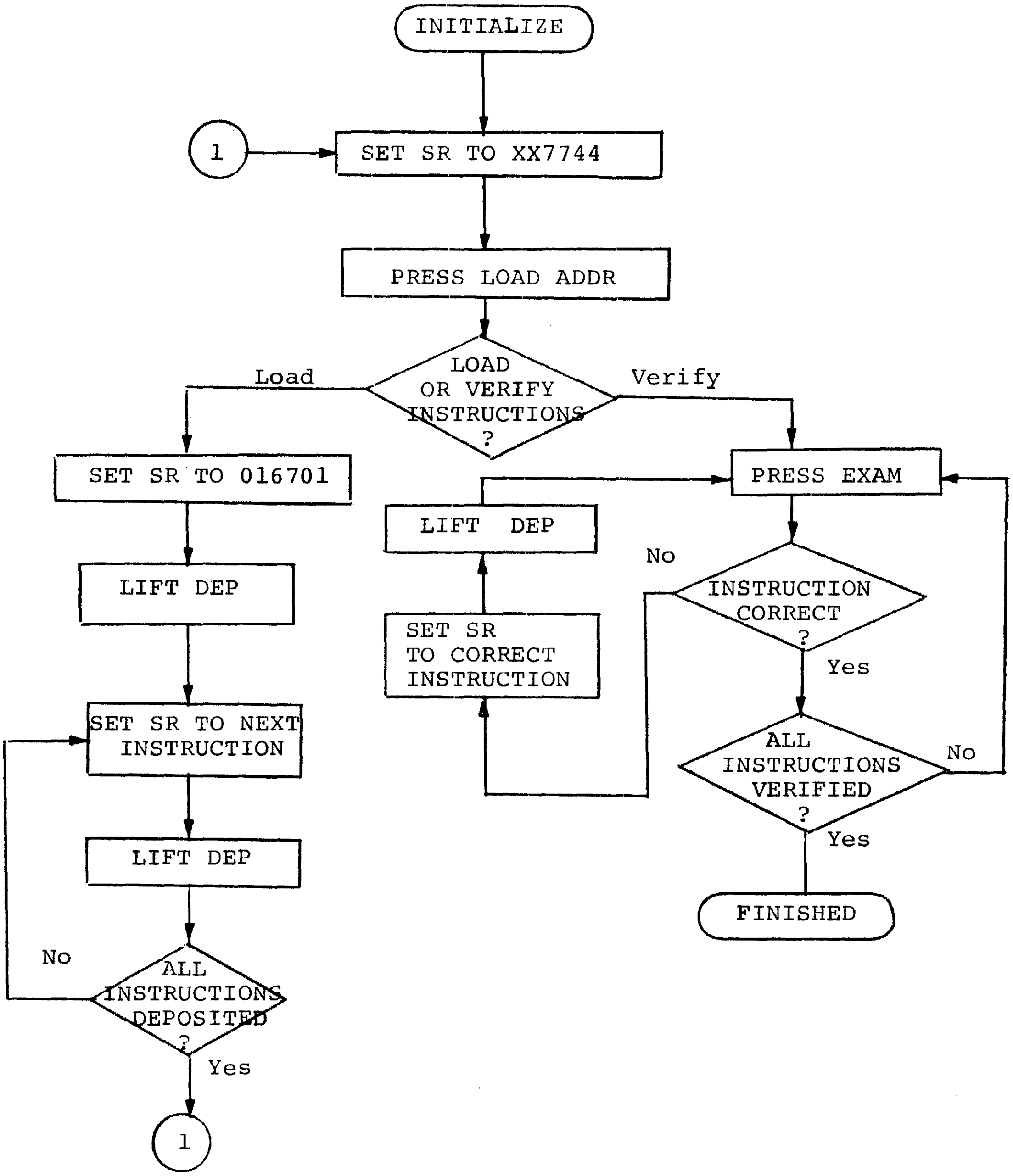


Figure A-1 Loading and Verifying the Bootstrap Loader

A.1.2 Loading with the Bootstrap Loader

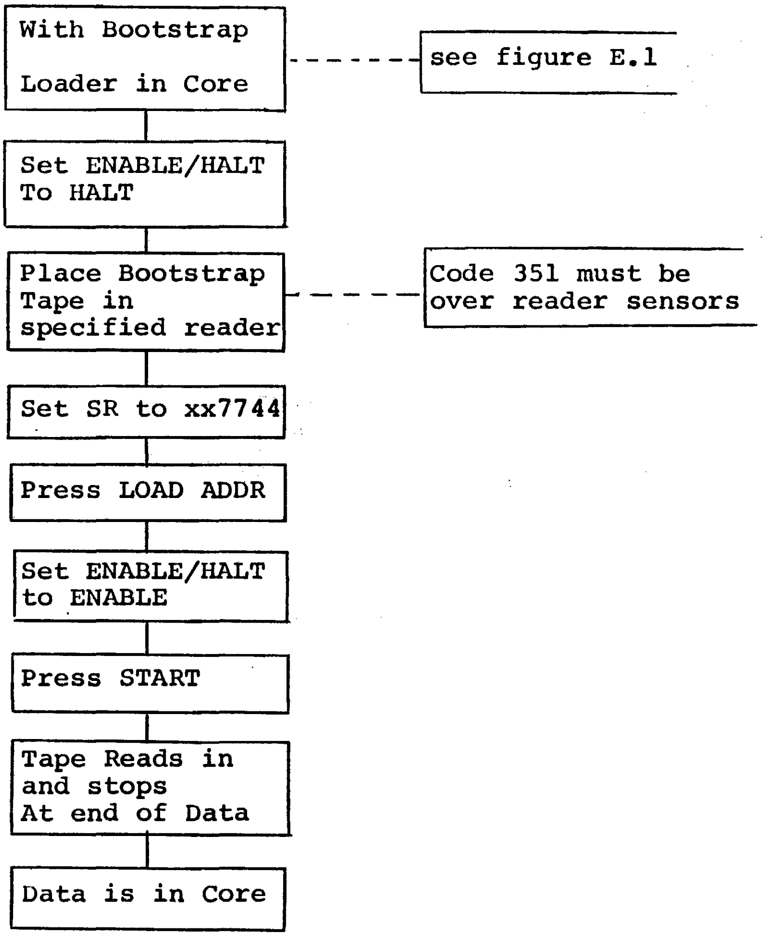

With Bootstrap Loader in Core
see figure E.1
Set ENABLE/HALT To HALT
Place Bootstrap Tape in specified reader
Code 351 must be over reader sensors
Set SR to xx7744
Press LOAD ADDR
Set ENABLE/HALT to ENABLE
Press START
Tape Reads in and stops At end of Data
Data is in Core

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.

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

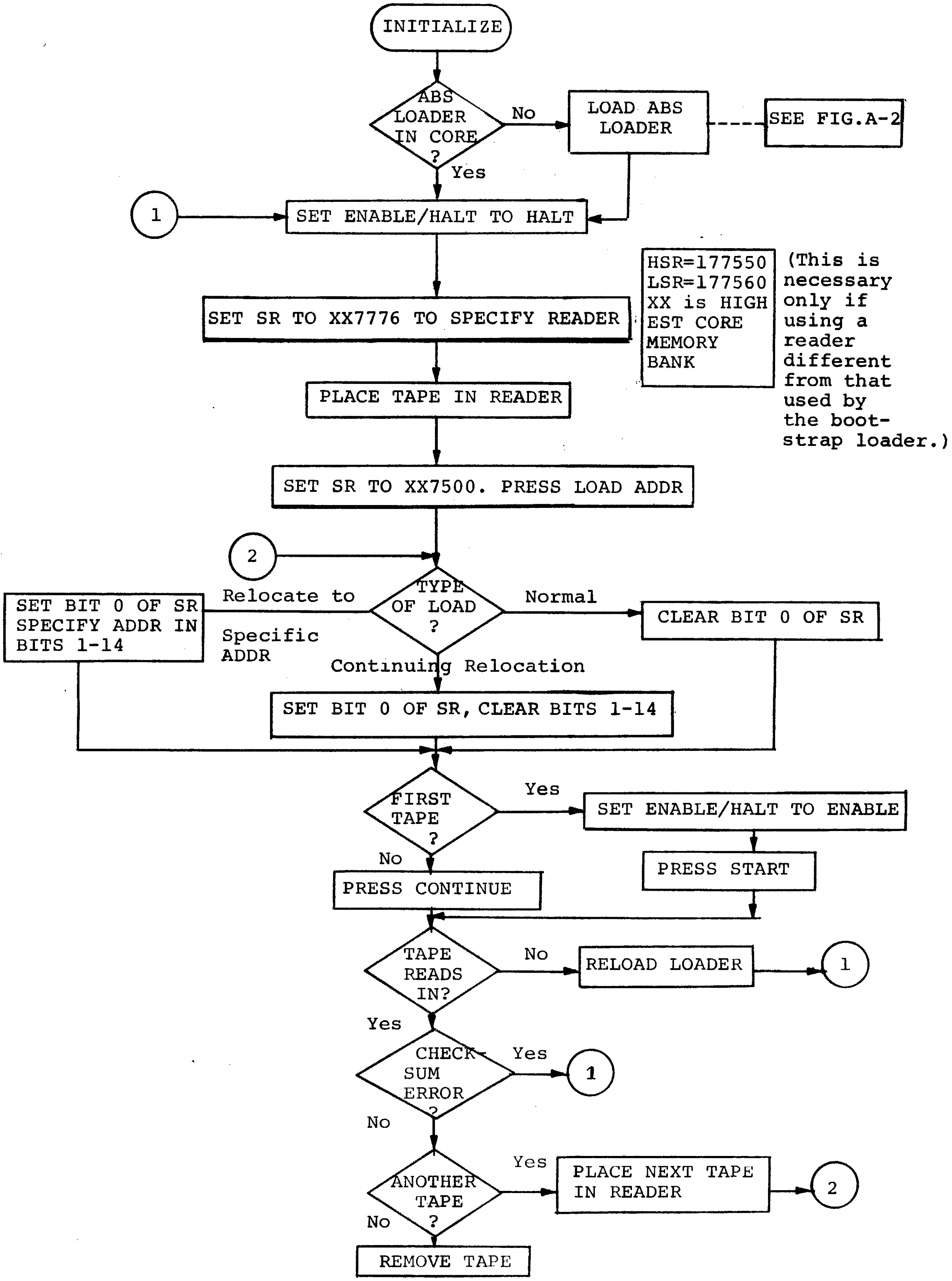

INITIALIZE
ABS LOADER IN CORE ?
No
Yes
LOAD ABS LOADER
SEE FIG.A-2
1
SET ENABLE/HALT TO HALT
SET SR TO XX7776 TO SPECIFY READER
HSR=177550
LSR=177560
XX is HIGHEST CORE MEMORY BANK
(This is necessary only if using a reader different from that used by the bootstrap loader.)
PLACE TAPE IN READER
SET SR TO XX7500. PRESS LOAD ADDR
2
TYPE OF LOAD ?
Relocate to Specific ADDR
Normal
Continuing Relocation
SET BIT 0 OF SR SPECIFY ADDR IN BITS 1-14
CLEAR BIT 0 OF SR
SET BIT 0 OF SR, CLEAR BITS 1-14
FIRST TAPE ?
Yes
No
SET ENABLE/HALT TO ENABLE
PRESS START
PRESS CONTINUE
TAPE READS IN?
No
Yes
RELOAD LOADER
1
CHECK-SUM ERROR ?
Yes
No
1
ANOTHER TAPE ?
Yes
No
PLACE NEXT TAPE IN READER
2
REMOVE TAPE

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
                $subrl              ;call desired subroutines
                $subr2
                .
                .
                .
                $subrn              ;call last subroutine desired
                .WORD   .+2         ;leave Polish mode.
                .
                .
                .
--------------------------------------------------------------
J5RR:           .
                .
                .
                JSR R5,subr         ;call desired subroutine
                BR      XX
                .WORD   argl        ;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.

	OCTAL		# OF		
NAME	CODE	PKG	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.
$ICO INT ASC	(67)		3	JPC	Integer to ASCII conversion. Calling sequence: Push address of ASCII field. Push length in bytes of ASCII field. Push integer value to be converted. JSR PC,$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
```

```
;PRODUCT CODE             DEC-11-NFPMA-A-LA

;COMPUTER                 PDP-11

;CONFIGURATION            PAPER TAPE CONFIGURATION IS MINIMUM
;                         8192 WORDS MEMORY

;SOFTWARE REQUIREMENTS    PAL-11S (OR MACRO-11)
;                         LINK-11S (OR LINK-11)

;PROGRAM NAME             FPMP-11

;VERSION                  VERSION LEVEL  1
;                         PATCH LEVEL    A

;DESCRIPTION              FLOATING POINT MATH PACKAGE
;                         PLUS TRAP HANDLER
;                         (FLOATING POINT SUBROUTINES TAKEN FROM
;                         DOS-11 FORTRAN IV OTS)

;AUTHOR                   E. PETERS (TRAP HANDLER & PACKAGE
;                                 INTEGRATION)

;DATE                     AUGUST, 1972

;                         COPYRIGHT 1972, DIGITAL EQUIPMENT CORP.,
;                         MAYNARD, MASSACHUSETTS 01754
```

```
        000000'         .CSECT

                ;       CONDITIONALS TO GENERATE THE STANDARD PACKAGES.
                        .IFDF   SINGLE; SINGLE PRECISION PACKAGE?
        000001          CND$2=1         ;$ADR,$SBR
        000001          CND$3=1         ;ALOG,ALOG10
        000001          CND$4=1         ;AINT
        000001          CND$6=1         ;$CMR
        000001          CND$18=1        ;$DVR
        000001          CND$20=1        ;EXP
        000001          CND$30=1        ;$MLR
        000001          CND$37=1        ;SIN,COS
        000001          CND$38=1        ;TANH
        000001          CND$39=1        ;ATAN,ATAN2
        000001          CND$41=1        ;SQRT
        000001          CND$44=1
        000001          CND$46=1
                        .ENDC

                        .IFDF   DOUBLE; DOUBLE PRECISION PACKAGE?
        000001          CND$1=1         ;$ADD,$SBD
        000001          CND$5=1         ;$CMD
        000001          CND$10=1        ;DLOG,DLOG10
        000001          CND$13=1        ;DSIN,DCOS
        000001          CND$14=1        ;DSQRT
        000001          CND$15=1        ;DATAN,DATAN2
        000001          CND$16=1        ;$DVD
        000001          CND$19=1        ;DEXP
        000001          CND$28=1        ;$MLD
        000001          CND$45=1        ;$LDD
        000001          CND$47=1        ;$STD
                        .ENDC

                        .IFDF   SINGLE!DOUBLE
        000001          CND$8=1         ;$DCI,$RCI
        000001          CND$9=1         ;$ECO,$FCO,$GCO,$DCO
        000001          CND$31=1        ;$NGI,$NGR,$NGD
        000001          CND$42=1        ;TRAPH
                        .ENDC
```

```
                         .IFDF   CND$38; TANH?
        000001           CND$2=1         ;$ADR,$SBR
        000001           CND$18=1        ;$DVR
        000001           CND$20=1        ;EXP
        000001           CND$21=1        ;$FCALL
        000001           CND$30=1        ;$MLR
        000001           CND$32=1        ;$PSHR3
                         .ENDC

                         .IFNDF  FPU
                         .IFDF   CND$3!CND$20!CND$37!CND$39
        000001           CND$2=1         ;$ADR,$SBR
        000001           CND$18=1        ;$DVR
        000001           CND$30=1        ;$MLR
                         .IFDF   CND$37; SIN,COS?
        000001           CND$4=1         ;$INTR
                         .ENDC
                         .ENDC
                         .IFDF   CND$10!CND$13!CND$15!CND$19
        000001           CND$1=1         ;$ADD,$SBD
        000001           CND$16=1        ;$DVD
        000001           CND$28=1        ;$MLD
        000001           CND$33=1        ;$POPR4
                         .IFDF   CND$13; DSIN,DCOS?
        000001           CND$11=1        ;$DINT
                         .ENDC
                         .ENDC
                         .IFDF   CND$3!CND$10;   ALOG OR DLOG?
        000001           CND$27=1        ;$IR,$ID
                         .ENDC
                         .IFDF   CND$19!CND$20;  EXP OR DEXP?
        000001           CND$27=1        ;$IR,$ID
        000001           CND$35=1        ;$RI,$DI
                         .ENDC
                         .IFDF   CND$14; DSQRT?
        000001           CND$1=1         ;$ADD
        000001           CND$16=1        ;$DVD
                         .ENDC
                         .IFDF   CND$41; SQRT?
        000001           CND$2=1         ;$ADR
        000001           CND$18=1        ;$DVR
                         .ENDC
                         .ENDC

                         .IFDF   CND$23; FLOAT?
        000001           CND$27=1        ;$IR,$ID
        000001           CND$33=1        ;$POPR3
                         .ENDC
                         .IFDF   CND$22!CND$26;  IFIX, INT, OR IDINT?
        000001           CND$35=1        ;$RI,$DI
                         .ENDC
                         .IFDF   CND$39; ATAN OR ATAN2?
        000001           CND$33=1        ;POPR3
                         .ENDC
```

```
1                          .TITLE  TRAPØ2
2                          .IFDF   CND$42
3                          .GLOBL  TRAPH,$ERRA
4                       ;  THE FPMP-11 TRAP HANDLER
5          ØØØØØØ          RØ=%Ø
6          ØØØØØ1          R1=%1
7          ØØØØØ2          R2=%2
8          ØØØØØ3          R3=%3
9          ØØØØØ4          R4=%4
10         ØØØØØ5          R5=%5
11         ØØØØØ6          SP=%6
12         ØØØØØ7          PC=%7
13
14 ØØØØØ ØØ2766 TRAPH:  BIC     #17,2(SP);       CLEAR ALL USER COND CODES
         ØØØØ17
         ØØØØØ2
15 ØØØØ6 ØØ5Ø46         CLR     -(SP);           SPACE FOR ADDR MODE
16 ØØØ1Ø Ø1Ø546         MOV     R5,-(SP);        SAVE THE REGISTERS
17 ØØØ12 Ø1Ø446         MOV     R4,-(SP)
18 ØØØ14 Ø1Ø346         MOV     R3,-(SP)
19 ØØØ16 Ø1Ø246         MOV     R2,-(SP)
20 ØØØ2Ø Ø1Ø146         MOV     R1,-(SP)
21 ØØØ22 Ø1ØØ46         MOV     RØ,-(SP)
22 ØØØ24 Ø166Ø3         MOV     2Ø(SP),R3;       GET USER'S STATUS WORD
         ØØØØ2Ø
23 ØØØ3Ø Ø427Ø3         BIC     #2Ø,R3;          CLEAR T-BIT FOR US
         ØØØØ2Ø
24 ØØØ34 Ø1Ø337         MOV     R3,@#177776;     ESTABLISH AS CURRENT STATUS
         177776
25 ØØØ4Ø Ø166Ø1         MOV     16(SP),R1;       GET USER'S PC
         ØØØØ16
26 ØØØ44 Ø1Ø1Ø5         MOV     R1,R5;           COPY USER'S PC
27 ØØØ46 Ø141Ø4         MOV     -(R1),R4;        PICK UP TRAP INSTRUCTION
28 ØØØ5Ø Ø1Ø4Ø3         MOV     R4,R3;           COPY
29 ØØØ52 Ø427Ø4         BIC     #1777ØØ,R4;      CALC TABLE INDEX
         1777ØØ
30 ØØØ56 ØØ63Ø4         ASL     R4;              TIMES TWO
31 ØØØ6Ø Ø164Ø4         MOV     TBL$42(R4),R4;   GET TABLE ENTRY
         ØØØ5ØØ'
32 ØØØ64 ØØ1556         BEQ     ERR$42;          ERROR: NO ENTRY IN TABLE
33 ØØØ66 Ø1Ø4Ø2         MOV     R4,R2;           COPY TABLE ENTRY
34 ØØØ7Ø Ø427Ø2         BIC     #14ØØØØ,R2;      CLEAR MODE BITS
         14ØØØØ
35 ØØØ74 Ø6Ø7Ø2         ADD     PC,R2;           RELOCATE ROUTINE ADDRESS
36 ØØØ76 Ø327Ø4 PTS$42: BIT     #4ØØØØ,R4;       ADDRESSING REQUIRED
         Ø4ØØØØ
37 ØØ1Ø2 ØØ1514         BEQ     NAD$42;          BRANCH IF NONE REQUIRED
38 ØØ1Ø4 1Ø61Ø3         ROLB    R3;              TEST OPERAND ADDRESS MODE
39 ØØ1Ø6 1ØØ122         BPL     PLM$42;          BRANCH IF BIT 6 EQUALS Ø
40 ØØ11Ø 1Ø3ØØ4         BCC     STK$42;          BRANCH IF @RØ MODE
41                      ;  RELATIVE MODE
42 ØØ112 Ø1Ø5ØØ         MOV     R5,RØ;           COPY USER'S PC
43 ØØ114 Ø625ØØ         ADD     (R5)+,RØ;        CALC ACTUAL OPERAND ADDRESS
44 ØØ116 Ø1Ø566 UPC$42: MOV     R5,16(SP);       UPDATE USER'S PC
         ØØØØ16
45 ØØ122 Ø127Ø5 STK$42: MOV     #FAC$42+6,R5;    ADDRESS OF FLAC
         ØØØ44Ø'
```

```
00126 005704            TST     R4;               SINGLE OR DOUBLE?
00130 002403            BLT     ST4$42;           BRANCH IF DOUBLE
00132 005015            CLR     @R5;              CLEAR LAST 2 WORDS OF FLAC
00134 005045            CLR     -(R5)
00136 005725            TST     (R5)+;            INCR R5
00140 011546 ST4$42:    MOV     @R5,-(SP);        PUST THE FLAC
00142 014546            MOV     -(R5),-(SP)
00144 014546            MOV     -(R5),-(SP)
00146 014546            MOV     -(R5),-(SP)
00150 005704            TST     R4;               SINGLE OR DOUBLE?
00152 002402            BLT     ST6$42;           BRANCH IF DOUBLE
00154 022020            CMP     (R0)+,(R0)+;      INCR R0 BY 4
00156 000404            BR      OT2$42
00160 062700 ST6$42:    ADD     #8.,R0
      000010
00164 014046            MOV     -(R0),-(SP);      PUSH OPERAND
00166 014046            MOV     -(R0),-(SP)
00170 014046 OT2$42:    MOV     -(R0),-(SP)
00172 014046            MOV     -(R0),-(SP)

                 ;      CALL ROUTINE IN POLISH MODE.
                 ;      THIS IS NOT A STANDARD POLISH CALL
                 ;      IN ORDER TO REDUCE OVERHEAD.
00174 012704            MOV     #ADR$42,R4;       ADDRESS OF RETURN ADDR
      000202'
00200 000112            JMP     @R2;              CALL SUBROUTINE
00202 000204'ADR$42:    .WORD   .+2;              RETURN ADDRESS
                 ;      NOW POP RESULT TO FLAC
00204 012705            MOV     #FAC$42,R5;       ADDR OF FLAC
      000432'
00210 012625            MOV     (SP)+,(R5)+
00212 012625            MOV     (SP)+,(R5)+
00214 012625            MOV     (SP)+,(R5)+
00216 012625            MOV     (SP)+,(R5)+
00220 011700 RET$42:    MOV     @PC,R0;           MAKE R0 POSITIVE
00222 012705            MOV     #FAC$42,R5;       ADDR OF FLAC
      000432'
00226 005725            TST     (R5)+;            TEST THE FLAC
00230 002410            BLT     NEG$42;           BRANCH IF FLAC MINUS
00232 003013            BGT     PLS$42;           BRANCH IF PLUS
00234 005725            TST     (R5)+
00236 001011            BNE     PLS$42
00240 005725            TST     (R5)+
00242 001007            BNE     PLS$42
00244 005725            TST     (R5)+
00246 001005            BNE     PLS$42
00250 005000            CLR     R0;               FLAG FLAC AS ZERO
00252 005400 NEG$42:    NEG     R0;               FLAC IS NEG
00254 053766 CMF$42:    BIS     @#177776,20(SP) ;SET USER'S CONDS
      177776
      000020
00262 005700 PLS$42:    TST     R0;               SET COND CODES
00264 012600 CM1$42:    MOV     (SP)+,R0;         RESTORE USER'S REGS
00266 012601            MOV     (SP)+,R1
00270 012602            MOV     (SP)+,R2
00272 012603            MOV     (SP)+,R3
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 JSRR CALLS
110 0334 004512 NAU$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
```

```
    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-PTS42+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-PTS42+PMODE          ;21
193                       .ENDC
194                       .IFNDF  CND$30
195                       .WORD   0                         ;21
196                       .ENDC
197                       .IFDF   CND$28
198 0544 156050           .WORD   $MLD-PTS42+PMODE+DMODE    ;22
199                       .ENDC
200                       .IFNDF  CND$28
201                       .WORD   0                         ;22
202                       .ENDC
203                       .IFDF   CND$16
204 0546 152112           .WORD   $DVD-PTS42+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-PTS42+PMODE          ;25
212                       .ENDC
213                       .IFNDF  CND$18
214                       .WORD   0                         ;25
215                       .ENDC
216                       .IFDF   CND$4
217 0554 003026           .WORD   AINT-PTS42                ;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-PTS42,COS-PTS42       ;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-PTS42+DMODE          ;40
231 0602 107576           .WORD   DCOS-PTS42+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 000636 000000               .WORD   0,0,0,0,0,0,0,0,0,0     ;57-70
    000640 000000
    000642 000000
    000644 000000
    000646 000000
    000650 000000
    000652 000000
    000654 000000
    000656 000000
    000660 000000
295                             .IFDF   CND$44
296 000662 061746               .WORD   $LDR-PT$42+PMODE        ;71
297                             .ENDC
298                             .IFNDF  CND$44
299                             .WORD   0                       ;71
300                             .ENDC
301                             .IFDF   CND$45
302 000664 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 000666 062000               .WORD   $STR-PT$42+PMODE        ;73
309                             .ENDC
310                             .IFNDF  CND$46
311                             .WORD   0                       ;73
312                             .ENDC
313                             .IFDF   CND$47
314 000670 162054               .WORD   $STD-PT$42+PMODE+DMODE  ;74
315                             .ENDC
316                             .IFNDF  CND$47
317                             .WORD   0                       ;74
318                             .ENDC
319 000672 000000               .WORD   0,0,0                   ;75-77
    000674 000000
    000676 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
```

```
        000006          SP=%6
        000007          PC=%7
        000006          A1=6
        000010          B1=8.
        000012          C1=10.
        000014          D1=12.
        000016          A2=14.
        000020          B2=16.
        000022          C2=18.
        000024          D2=20.
        000000          SIGNS=0.
        177304          MQ=177304
        177312          NOR=177312
        177314          LSH=177314
        177316          ASH=177316
        000000          F0=%0
   0700 062716 $SBD:    ADD     #100000,@SP     ;NEGATE TOP STACK ITEM
        100000
                        .IFDF   FPU
                $ADD:   .WORD   170011  ;;SETD
                        .WORD   172426  ;;LDD   (SP)+,F0        ;GET OPERAND
                        .WORD   172026  ;;ADDD  (SP)+,F0        ;ADD
                        .WORD   174046  ;;STD   F0,-(SP)        ;SUM TO STACK
                        JMP     @(R4)+
                        .ENDC
                        .IFNDF  FPU
   0704 010446 $ADD:    MOV     R4,-(SP)
   0706 010546          MOV     R5,-(SP)
   0710 005046          CLR     -(SP)   ;CLEAR SIGNS
   0712 005004          CLR     R4      ;CLEAR EXPONENTS
   0714 005005          CLR     R5
   0716 006366          ASL     D1(SP)  ;SHIFT OUT SIGN OF TOP ITEM
        000014
   0722 006166          ROL     C1(SP)
        000012
   0726 006166          ROL     B1(SP)
        000010
   0732 006166          ROL     A1(SP)  ;SHIFT A1
        000006
   0736 156604          BISB    A1+1(SP),R4     ;GET E1
        000007
   0742 001441          BEQ     A1ZS1   ;JUMP IF ZERO
   0744 106116          ROLB    @SP     ;GET S1
   0746 006366          ASL     D2(SP)  ;SHIFT OUT SIGN OF SECOND ITEM
        000024
   0752 006166          ROL     C2(SP)
        000022
   0756 006166          ROL     B2(SP)
        000020
   0762 006166          ROL     A2(SP)  ;SHIFT A2
        000016
   0766 156605          BISB    A2+1(SP),R5     ;GET E2
        000017
   0772 001030          BNE     A2NS1   ;JUMP IF NOT 0
   0774 106016          RORB    @SP     ;RECONSTRUCT A1
   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 A1L$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 126016 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      NOU$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
```

```
                               MOV     (SP)+,R4         ;RESTORE EXPONENT TO R4
                               BR      SFUS1
                               .ENDC
1376 005726 SBAS1:  TST     (SP)+    ;POP EXTENSION
1400 006200 SRBS1:  ASR     R0       ;SHIFT RIGHT
1402 006001         ROR     R1
1404 006002         ROR     R2
1406 006003         ROR     R3
1410 005205         INC     R5       ;COUNT LOOP
1412 002772         BLT     SRBS1
1414 066603 SFUS1:  ADD     D1(SP),R3        ;FORM THE SUM
     000014
1420 005502         ADC     R2
1422 005501         ADC     R1
1424 005500         ADC     R0
1426 066602         ADD     C1(SP),R2
     000012
1432 005501         ADC     R1
1434 005500         ADC     R0
1436 066601         ADD     B1(SP),R1
     000010
1442 005500         ADC     R0
1444 066600         ADD     A1(SP),R0
     000006
1450 126616         CMPB    SIGNS+1(SP),@SP ;CHECK FOR UNEQUAL SIGNS
     000001
1454 001065         BNE     SUBS1    ;GO CLEAN UP SUBTRACT
1456 030027         BIT     R0,#1000
     001000
1462 001405         BEQ     NODS1    ;JUMP IF NO NORMAL BIT OVERFLOW
1464 006200         ASR     R0
1466 006001         ROR     R1
1470 006002         ROR     R2
1472 006003         ROR     R3
1474 005204         INC     R4       ;INCREASE EXPONENT
1476 000304 NODS1:  SWAB    R4       ;MOVE EXPONENT LEFT
1500 001031         BNE     OVFS1    ;JUMP IF OVERFLOW
1502 150004 NFLS1:  BISB    R0,R4    ;INSERT HIGH ORDER FRACTION
1504 006026         ROR     (SP)+    ;INSERT SIGN
1506 006004         ROR     R4
1510 006001         ROR     R1
1512 006002         ROR     R2
1514 006003         ROR     R3
1516 005503         ADC     R3
1520 005502         ADC     R2
1522 005501         ADC     R1
1524 005504         ADC     R4
1526 102417         BVS     OVRS1    ;JUMP IF OVERFLOW ON ROUND
1530 103416         BCS     OVRS1
1532 010466         MOV     R4,A2+0-2(SP)    ;STORE EXPONENT AND SIGN
     000014
1536 010166         MOV     R1,B2+0-2(SP)    ;INSERT LOW ORDER FRACTION
     000016
1542 010266         MOV     R2,C2+0-2(SP)
     000020
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
```

```
611                             MOV     @R4,R1  ;SAVE PARTIAL R1
612                             MOV     R2,@R4  ;GET NEXT
613                             CLR     -(R4)
614                             MOV     @SP,@#LSH       ;SHIFT IT
615                             BIS     (R4)+,R1        ;FINISH R1
616                             MOV     R3,@R4  ;GET NEXT
617                             MOV     R2,-(R4)
618                             MOV     @SP,@#LSH       ;SHIFT IT
619                             MOV     (R4)+,R2        ;FINISH R2
620                             MOV     @R4,R3  ;R3 DONE
621                             SUB     (SP)+,@SP       ;COMPENSATE EXPONENT
622                             MOV     (SP)+,R4        ;RESTORE IT TO R4
623                             BGT     NOD$1   ;JUMP IF NO UNDERFLOW
624                             BR      UNF$1
625                             .ENDC
626 1666 Ø3ØØ27 B9A$1:  BIT     RØ,#4ØØ ;CHECK NORMAL BIT
         ØØØ4ØØ
627 1672 ØØ1341         BNE     UTS$1   ;JUMP IF FOUND
628 1674 ØØ53Ø4         DEC     R4      ;DECREASE EXPONENT
629 1676 ØØ63Ø3         ASL     R3      ;DOUBLE FRACTION
63Ø 17ØØ ØØ61Ø2         ROL     R2
631 17Ø2 ØØ61Ø1         ROL     R1
632 17Ø4 ØØ61ØØ         ROL     RØ
633 17Ø6 ØØØ767         BR      B9A$1   ;TRY AGAIN
634 171Ø 1627Ø4 ZTS$1:  SUB     #8.,R4  ;REDUCE EXPONENT
         ØØØØ1Ø
635 1714 ØØ57Ø1         TST     R1
636 1716 ØØ1Ø2Ø         BNE     ZT1$1   ;JUMP IF ONLY RØ#Ø
637 172Ø 1627Ø4         SUB     #16.,R4
         ØØØØ2Ø
638 1724 Ø1Ø2Ø1         MOV     R2,R1
639 1726 ØØ1Ø12         BNE     ZT2$1   ;JJUMP IF R2 NOT Ø
64Ø 173Ø 1627Ø4         SUB     #16.,R4
         ØØØØ2Ø
641 1734 ØØ57Ø3         TST     R3
642 1736 ØØ1731         BEQ     ZER$1   ;ANSWER IS Ø
643 174Ø 15Ø3Ø1         BISB    R3,R1   ;MOVE BYTES TO RØ,R1
644 1742 ØØØ3Ø1         SWAB    R1
645 1744 ØØØ3Ø3         SWAB    R3
646 1746 15Ø3ØØ         BISB    R3,RØ
647 175Ø ØØ5ØØ3         CLR     R3      ;MAKE ALL OTHERS Ø
648 1752 ØØØ745         BR      BT9$1   ;GO NORMALIZE
649 1754 Ø1Ø3Ø2 ZT2$1:  MOV     R3,R2
65Ø 1756 ØØ5ØØ3         CLR     R3
651 176Ø ØØØ3Ø1 ZT1$1:  SWAB    R1      ;MOVE ALL BYTES LEFT
652 1762 15Ø1ØØ         BISB    R1,RØ
653 1764 1Ø5ØØ1         CLRB    R1
654 1766 ØØØ3Ø2         SWAB    R2
655 177Ø 15Ø2Ø1         BISB    R2,R1
656 1772 1Ø5ØØ2         CLRB    R2
657 1774 ØØØ3Ø3         SWAB    R3
658 1776 15Ø3Ø2         BISB    R3,R2
659 2ØØØ 1Ø5ØØ3         CLRB    R3
66Ø 2ØØ2 ØØØ731         BR      BT9$1   ;GO NORMALIZE WHAT'S LEFT
661                             .ENDC
662                             .ENDC
```

```
                                .TITLE  $ADR04
                                .IFDF   CND32
                                .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     OUTS2   ;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
```

```
85                           .IFDF    EAE
86                  SFR$2:   MOV      R1,@#MQ ;MOVE FRACTION TO AC,MQ
87                           MOV      R0,@#AC
88                           MOV      R2,@#ASH        ;SHIFT RIGHT TO EQUALIZE EXPONEN
89                           MOV      @#MQ,R1 ;RECOVER SHIFTED FRACTION
90                           MOV      @#AC,R0
91                           .ENDC
92                           .IFDF    MULDIV
93                  SFR$2:   .WORD    073002  ;;ASHC  R2,R0
94                           .ENDC
95                           .IFNDF   EAE&MULDIV
96 02236 022702 SFR$2:   CMP      #-8.,R2 ;CHECK # OF BITS TO SHIFT
         177770
97 02242 003431          BLE      SF0$2   ;JUMP IF NOT MORE THAN 1/2 WORD
98 02244 005004          CLR      R4      ;SET UP EXTENSION BITS
99 02246 005700          TST      R0      ;BASED ON HIGH ORDER FRACTION
100 2250 100001          BPL      NCP$2   ;JUMP IF +
101 2252 005104          COM      R4      ;- OTHERWISE
102 2254 022702 NCP$2:   CMP      #-16.,R2
         177760
103 2260 002405          BLT      SRL$2   ;JUMP IF LESS THAN ONE WORD TO SHIFT
104 2262 010001          MOV      R0,R1   ;SHIFT RIGHT A WHOLE WORD
105 2264 010400          MOV      R4,R0   ;USE EXTENSION BITS
106 2266 062702          ADD      #16.,R2 ;ACCOUNT FOR SHIFT
         000020
107 2272 001421          BEQ      SFD$2
108 2274 022702 SRL$2:   CMP      #-8.,R2
         177770
109 2300 003412          BLE      SF0$2   ;JUMP IF NOT MORE THAN 1/2 WORD
110 2302 062702          ADD      #16.,R2 ;SHIFT LEFT 16-X
         000020
111 2306 006301 SFL$2:   ASL      R1
112 2310 006100          ROL      R0
113 2312 006104          ROL      R4
114 2314 005302          DEC      R2      ;COUNT LOOP
115 2316 003373          BGT      SFL$2
116 2320 010001          MOV      R0,R1   ;PUT RESULT IN R0, R1
117 2322 010400          MOV      R4,R0
118 2324 000404          BR       SFD$2
119 2326 006200 SF0$2:   ASR      R0      ;SHIFT A MIN AND B MIN
120 2330 006001          ROR      R1
121 2332 005202          INC      R2      ;REDUCE EXPONENT DIFFERENCE
122 2334 002774          BLT      SF0$2
123                      .ENDC
124 2336 066600 SFD$2:   ADD      A1(SP),R0       ;A1+A2
         000004
125 2342 066601          ADD      B1(SP),R1       ;B1+B2
         000006
126 2346 005500          ADC      R0
127 2350 126616          CMPB     SIGNS+1(SP),@SP
         000001
128 2354 001034          BNE      SUB$2   ;GO CLEAN UP SUBTRACT
129 2356 030027          BIT      R0,#1000
         001000
130 2362 001403          BEQ      NOD$2   ;JUMP IF NO NORMAL BIT OVERFLOW
131 2364 006200          ASR      R0
132 2366 006001          ROR      R1
```

```
2370 005203          INC     R3      ;INCREASE EXPONENT
2372 000303 NOU$2:   SWAB    R3      ;MOVE EXPONENT LEFT
2374 001020          BNE     OVR$2   ;JUMP IF OVERFLOW
2376 150003          BISB    R0,R3
2400 006016          ROR     @SP     ;INSERT SIGN
2402 006003          ROR     R3
2404 006001          ROR     R1
2406 005501          ADC     R1      ;ROUND SUM
2410 005503          ADC     R3
2412 102411          BVS     OVR$2   ;JUMP IF OVERFLOW ON ROUND
2414 103410          BCS     OVR$2
2416 010366 STR$2:   MOV     R3,A2(SP)       ;STORE EXPONENT AND SIGN
     000010
2422 010166          MOV     R1,B2(SP)       ;INSERT LOW ORDER FRACTION
     000012
2426 005726 OUT$2:   TST     (SP)+   ;POP SIGNS
2430 012604          MOV     (SP)+,R4
2432 022626          CMP     (SP)+,(SP)+     ;POP FIRST ARGUMENT
2434 000134          JMP     @(R4)+  ;DONE, RETURN
            ;
2436 004567 OVR$2:   JSR     R5,$ERR ;ERROR 3,2
     017344
2442 000771          BR      OUT$2
2444    003          .BYTE   3
2445    002          .BYTE   2
            ;
2446 005700 SUB$2:   TST     R0      ;CHECK HIGH ORDER RESULT FRACTION
2450 003005          BGT     BT9$2   ;IF POSITIVE SIGN IS OK
2452 001413          BEQ     ZTS$2   ;CHECK FOR ZERO RESULT
2454 005400          NEG     R0      ;GET ABSOLUTE VALUE
2456 005401          NEG     R1
2460 005600          SBC     R0
2462 000316          SWAB    @SP     ;EXCHANGE SIGNS
2464        BT9$2:
                     .IFDF   EAE
                     BIT     R0,#700
                     BNE     B9A$2   ;JUMP IF NOT MORE THAN 2 TO SHIFT
                     MOV     R1,@#MQ ;RESULT FRACTION TO AC,MQ
                     MOV     R0,@#AC
                     CLR     @#NOR   ;NORMALIZE
                     SUB     @#NOR,R3        ;ADJUST EXPONENT
                     MOV     #-6,@#ASH       ;SHIFT TO CORRECT POSITION
                     ADD     #6,R3   ;COMPENSATE EXPONENT
                     BLE     UNF$2   ;JUMP IF UNDERFLOW
                     MOV     @#AC,R0
                     MOV     @#MQ,R1 ;GET FRACTION BACK
                     BR      NOU$2
                     .ENDC
2464 030027 B9A$2:   BIT     R0,#400
     000400
2470 001014          BNE     UTS$2   ;JUMP IF NORMAL BIT FOUND
2472 005303          DEC     R3      ;DECREASE EXPONENT
2474 006301          ASL     R1      ;DOUBLE FRACTION
2476 006100          ROL     R0
2500 000771          BR      B9A$2   ;TRY AGAIN
2502 005701 ZTS$2:   TST     R1      ;CHECK LOW ORDER PART
                     .IFDF   EAE
```

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

```
                            .TITLE  $ALGØ3
                            .IFDF   CND$3
                      ;
                      ;     ALOG    VØØ3A
                      ;
                      ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                      ;
                            .GLOBL  ALOG,ALOG1Ø,$ERR;
                            .IFNDF  FPU
                            .GLOBL  $POLSH,$ADR,$SBR,$MLR,$DVR,$IR;
                            .ENDC
                      ;     THE FORTRAN ALOG AND ALOG1Ø FUNCTIONS
                      ;     CALLING SEQUENCE:
                      ;     JSR     R5,ALOG (OR ALOG1Ø)
                      ;     BR      A
                      ;     .WORD   ARGUMENT ADDRESS
                      ;A:
                      ;     RETURNS LN(ARG) (OR LOG1Ø(ARG)) IN RØ,R1.
                      ;
        ØØØØØØ              RØ=%Ø
        ØØØØØ1              R1=%1
        ØØØØØ2              R2=%2
        ØØØØØ3              R3=%3
        ØØØØØ4              R4=%4
        ØØØØØ5              R5=%5
        ØØØØØ6              SP=%6
        ØØØØØ7              PC=%7
        ØØØØØØ              FØ=%Ø
        ØØØØØ1              F1=%1
        ØØØØØ2              F2=%2
        ØØØØØ3              F3=%3
                            .IFNDF  FPU
 Ø2544  Ø11746 ALOG1Ø:      MOV     PC,-(SP)          ;GET ØØ4XX AS A FLAG
 Ø2546  ØØØ4Ø1              BR      LOG$$
 Ø255Ø  ØØ5Ø46 ALOG:        CLR     -(SP)   ;FLAG ALOG
 Ø2552  Ø165Ø4 LOG$$:       MOV     2(R5),R4          ;GET ARG ADDRESS
        ØØØØØ2
 Ø2556  Ø12746              MOV     #Ø71Ø3Ø,-(SP)   ;PUSH -1/2*LN(2)
        Ø71Ø3Ø
 Ø2562  Ø1274Ø              MOV     #137Ø61,-(SP)
        137Ø61
 Ø2566  Ø24646              CMP     -(SP),-(SP)     ;GET WORK SPACE
 Ø257Ø  Ø1644Ø              MOV     2(R4),-(SP)     ;GET ARG
        ØØØØØ2
 Ø2574  Ø1144Ø              MOV     @R4,-(SP)
 Ø2576  ØØ3534              BLE     ERR$$   ;JUMP IF NOT POSITIVE
 Ø26ØØ  ØØ6316              ASL     @SP
 Ø26Ø2  11666Ø              MOVB    1(SP),12.(SP)   ;GET EXPONENT
        ØØØØØ1
        ØØØØ14
 Ø261Ø  112766              MOVB    #2ØØ,1(SP)        ;TRANSFORM ARG TO (1/2,1)
        ØØØ2ØØ
        ØØØØØ1
 Ø2616  ØØ6Ø16              ROR     @SP
 Ø262Ø  Ø12746              MOV     #ØØ2363,-(SP)   ;PUSH 1/2*ROOT2
        ØØ2363
 Ø2624  Ø12746              MOV     #Ø4ØØ65,-(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     #FC0$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)
```

```
                          ADDF    (R0)+,F1;       X+1/2*SQRT(2)
                          DIVF    F1,F2;          W=(X-ROOT2)/(X+ROOT2)
                          LDF     F2,F1;
                          MULF    F1,F1;          Y= W**2
                  ;
                          MOV     #3,R1;          COUNT OF CONSTS FOR POLYNOMIAL
                          LDF     (R0)+,F0;       INITIALIZE ACCUMULATOR FOR POLYN
                  XPDS3:  MULF    F1,F0;
                          DEC     R1;             COUNT
                          ADDF    (R0)+,F0;       F0:= Y*F0 + C(I)
                          BGT     XPDS3;          LOOP
                  ;
                          MULF    F2,F0;
                          ADDF    (R0)+,F0;       F0:= W*F0 - 1/2*LN(2)
                          ADDF    F3,F0;          ADD SCALE FACTOR FOR EXPONENT
                          TST     R4;             TEST ALOG10 FLAG
                          BEQ     LGTS3;
                          MULF    (R0)+,F0;       ALOG10:= ALOG*LOG10(E)
                  ;
                  LGTS3:  STF     F0,-(SP);       MOVE RESULT TO STACK
                          MOV     (SP)+,R0;
                          MOV     (SP)+,R1;       AND THENCE TO R0,R1
                          RTS     R5;
                  ERRS3:  JSR     R5,$ERR;        ERROR 4,10
                          RTS     R5;             EXIT-NO STACK CLEANUP NECESSARY
                          .BYTE   4
                          .BYTE   10.
                  ;       ORDER-DEPENDENT CONSTANTS FOR ROUTINE
                  ;       R0 POINTS AT CURRENT CONSTANT IN FPU VERSION
                  ;
                  FCUS3:  .WORD   040061,071030;  LN(2)
                  ;
                          .WORD   040065,002363;  1/2*SQRT(2)
                          .ENDC
                  ;       CONSTANTS FOR POLYNOMIAL EXPANSION
                  ;
3104 037632               .WORD   037632,014525   ;.300974506
3106 014525
                  ;
3110 037714               .WORD   037714,120036   ;.399659100
3112 120036
                  ;
3114 040052               .WORD   040052,125332   ;.666669471
3116 125332
                  ;
3120 040400       CONS3:  .WORD   040400,000000   ;1.99999999
3122 000000
                          .IFDF   FPU
                  ;       MORE ORDER-DEPENDENT CONSTANTS
                          .WORD   137661,071030;  -1/2*LN(2)
                  ;
                          .WORD   037736,055731;  LOG10(E)
                          .ENDC
                          .ENDC
```

```
                                .TITLE  SANT03
                                .IFDF   CND$4
                                .GLOBL  AINT,SINTR
                      ;         AINT    V003A
                      ;         COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYARD, MASS.
                      ;         AINT    FORTRAN AINT FUNCTION, CALLING SEQUENCE
                      ;         JSR     R5,AINT
                      ;         BR      A
                      ;         .WORD   ADDRESS OF ARGUMENT
                      ;A:
                      ;         RETURNS SIGN OF ARG * GREATEST REAL INTEGER <=
                      ;         ABS(ARG) IN R0 AND R1.
                      ;
                      ;         SINTR   SAME FUNCTION AS AINT, BUT CALLED IN THE
                      ;         POLISH MODE WITH THE ARGUMENT AND RETURN ON THE STACK.
         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
         177314                 LSH=177314
         000000                 F0=%0
         000001                 F1=%1
                                .IFDF   FPU
                      AINT:     .WORD   170001  ;;SETF
                                .WORD   172475,2        ;;LDF   @2(R5),F0       ;GET ARG
                                .WORD   171467,24       ;;MODF  ONE,F0  ;GET INTEGER PAR
                                .WORD   174146  ;;STF   F1,-(SP)
                                MOV     (SP)+,R0        ;POP TO USER REGS
                                MOV     (SP)+,R1
                                RTS     R5      ;RETURN
                      ;
                      SINTR:    .WORD   170001  ;;SETF
                                .WORD   172426  ;;LDF   (SP)+,F0        ;GET ARG
                                .WORD   171467,4        ;;MODF  ONE,F0  ;GET INTEGER PAR
                                .WORD   174146  ;;STF   F1,-(SP)
                                JMP     @(R4)+  ;RETURN
                      ONE$4:    .WORD   040200,0        ;FLOATING 1.
                                .ENDC
                                .IFNDF  FPU
   03124 016504       AINT:     MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
         000002
   03130 011400                 MOV     @R4,R0  ;GET HIGH ORDER ARGUMENT
   03132 016401                 MOV     2(R4),R1        ;LOW ORDER
         000002
   03136 010702                 MOV     PC,R2   ;MAKE R2 NON 0
   03140 000403                 BR      AI1$4
   03142 005002       SINTR:    CLR     R2      ;MAKE R2 0
   03144 012600                 MOV     (SP)+,R0        ;GET HIGH ORDER ARGUMENT
   03146 012601                 MOV     (SP)+,R1        ;LOW ORDER
   03150 010003       AI1$4:    MOV     R0,R3
   03152 006103                 ROL     R3      ;DUMP SIGN
   03154 105003                 CLRB    R3
```

```
03156 000303                SWAB    R3      ;GET EXPONENT
03160 162703                SUB     #230,R3 ;REMOVE EXCESS 200 AND CHECK RANGE
      000230
03164 002020                BGE     DNE$4   ;JUMP IF IT IS ALREADY AN INTEGER
03166 022703                CMP     #-30,R3
      177750
03172 002403                BLT     SHF$4   ;JUMP IF THERE IS WORK TO DO
03174 005000                CLR     R0      ;ARG IS < 1, SO RETURN 0
03176 005001                CLR     R1
03200 000412                BR      DNE$4
03202 010346  SHF$4:        MOV     R3,-(SP)        ;PUSH -SHIFT COUNT
                            .IFNDF  EAE&MULDIV
03204 006000  ROR$4:        ROR     R0      ;SHIFT FRACTION
03206 006001                ROR     R1
03210 005203                INC     R3      ;COUNT LOOP
03212 002774                BLT     ROR$4   ;GO AGAIN
03214 012603                MOV     (SP)+,R3        ;GET COUNT BACK
03216 006301  ASL$4:        ASL     R1      ;SHIFT FRACTION BACK WITH 0'S
03220 006100                ROL     R0
03222 005203                INC     R3      ;COUNT LOOP AGAIN
03224 002774                BLT     ASL$4
                            .ENDC
                     ;      EAE CODE
                            .IFDF   EAE
                            MOV     #MQ,R3  ;POINT TO MQ
                            MOV     R1,@R3  ;INSERT ARG
                            MOV     R0,-(R3)
                            MOV     @SP,@#LSH       ;SHIFT RIGHT
                            NEG     @SP     ;SET FOR LEFT
                            MOV     (SP)+,@#LSH     ;SHIFT LEFT
                            MOV     (R3)+,R0        ;RESULT TO REGS
                            MOV     @R3,R1
                            .ENDC
                     ;      MULDIV CODE
                            .IFDF   MULDIV
                            .WORD   073016  ;;ASHC  @SP,R0  ;SHIFT OUT FRACTION
                            NEG     @SP     ;SET FOR LEFT SHIFT
                            .WORD   073026  ;;ASHC  (SP)+,R0        ;SHIFT INTEGER P
                            .ENDC
03226 005702  DNE$4:        TST     R2      ;CHECK ENTRY FLAG
03230 001401                BEQ     DN1$4   ;JUMP IF $INTR
03232 000205                RTS     R5      ;RETURN IF $AINT
03234 010146  DN1$4:        MOV     R1,-(SP)        ;PUSH RESULT
03236 010046                MOV     R0,-(SP)
03240 000134                JMP     @(R4)+  ;POLISH RETURN
                            .ENDC
                            .ENDC
```

```
                               .TITLE  $CMD02
                               .IFDF   CND$5
                               .GLOBL  $CMD
                       ;       $CMD    THE DOUBLE COMPARE ROUTINE.
                       ;
                       ;       $CMD    V002A
                       ;
                       ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                       ;       CALLED IN THE POLISH MODE WITH THE TWO
                       ;       COMPARANDS ON THE STACK:
                       ;       FIRST IS AT 8(SP), SECOND IS @SP
                       ;       FLUSH THE TWO COMPARANDS AND RETURN
                       ;       THE FOLLOWING CONDITION CODES:
                       ;       FIRST < SECOND  N=1, Z=0
                       ;       FIRST = SECOND  N=0, Z=1
                       ;       FIRST > SECOND  N=0, Z=0
         000000                R0=%0
         000001                R1=%1
         000002                R2=%2
         000004                R4=%4
         000006                SP=%6
         000007                PC=%7
         000000                F0=%0
                               .IFDF   FPU
                       $CMD:   .WORD   170011  ;;SETD
                               .WORD   172426  ;;LDD   (SP)+,F0        ;GET SECOND ARG
                               .WORD   173426  ;;CMPD  (SP)+,F0        ;COMPARE
                               .WORD   170000  ;;CFCC  ;GET CONDITION CODES
                               JMP     @(R4)+
                               .ENDC
                               .IFNDF  FPU
   03242 011700        $CMD:   MOV     @PC,R0  ;GET 00XXXXX   XXXX01 IN R0
   03244 016601                MOV     8.(SP),R1       ;GET HIGH ORDER FIRST ARG
         000010
   03250 002004                BGE     FPS$5   ;JUMP IF FIRST ARG +
   03252 006300                ASL     R0      ;FLAG FIRST ARG -
   03254 012602                MOV     (SP)+,R2        ;GET HIGH SECOND ARG
   03256 002403                BLT     SME$5   ;JUMP IF BOTH SIGNS -
   03260 000422                BR      NEG$5   ;JUMP IF FIRST - AND SECOND +
   03262 012602        FPS$5:  MOV     (SP)+,R2
   03264 002421                BLT     PLS$5   ;JUMP IF FIRST + AND SECOND -
   03266 020102        SME$5:  CMP     R1,R2   ;COMPARE MAGNITUDES
   03270 001014                BNE     OUT$5   ;JUMP IF DIFFERENT
   03272 026616                CMP     8.(SP),@SP
         000010
   03276 001011                BNE     OUT$5
   03300 026666                CMP     10.(SP),2(SP)
         000012
         000002
   03306 001005                BNE     OUT$5
   03310 026666                CMP     12.(SP),4(SP)
         000014
         000004
   03316 001001                BNE     OUT$5
   03320 005000                CLR     R0      ;FLAG =
   03322 006000        OUT$5:  ROR     R0      ;SAVE C BIT AND TEST SECOND ARG -
   03324 103401                BCS     PLS$5   ;JUMP IF SECOND ARG +
   Ø3326 ØØ54ØØ NEG$5:  NEG     RØ       ;REVERSE C BIT
   Ø333Ø Ø627Ø6 PLS$5:  ADD     #14.,SP  ;POP ARGS
         ØØØØ16
   Ø3334 ØØ57ØØ         TST     RØ       ;SET Z AND N BITS CORRECTLY
   Ø3336 ØØØ134         JMP     @(R4)+   ;RETURN TO CALLER
                        .ENDC
                        .ENDC
```

```
                                .TITLE  SCMR02
                                .IFDF   CNDS6
                                .GLOBL  SCMR
                        ;       SCMR    THE REAL COMPARE ROUTINE.
                        ;
                        ;       SCMR    V002A
                        ;
                        ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYARD, MASS.
                        ;       CALLED IN THE POLISH MODE WITH THE TWO
                        ;       COMPARANDS ON THE STACK:
                        ;       FIRST IS AT 4(SP), SECOND IS @SP
                        ;       FLUSH THE TWO COMPARANDS AND RETURN
                        ;       THE FOLLOWING CONDITION CODES:
                        ;       FIRST < SECOND  N=1, Z=0
                        ;       FIRST = SECOND  N=0, Z=1
                        ;       FIRST > SECOND  N=0, Z=0
         000000                 R0=%0
         000001                 R1=%1
         000002                 R2=%2
         000004                 R4=%4
         000006                 SP=%6
         000007                 PC=%7
         000000                 F0=%0
                                .IFDF   FPU
                        SCMR:   .WORD   170001  ;;SETF
                                .WORD   172426  ;;LDF   (SP)+,F0        ;GET SECOND ARG
                                .WORD   173426  ;;CMPF  (SP)+,F0        ;COMPARE
                                .WORD   170000  ;;CFCC  ;GET CONDITION CODES
                                JMP     @(R4)+
                                .ENDC
                                .IFNDF  FPU
   03340 011700 SCMR:   MOV     @PC,R0  ;GET 00XXXX   XXXX01 IN R0
   03342 016601         MOV     4(SP),R1        ;GET HIGH ORDER FIRST ARG
         000004
   03346 002004         BGE     FPSS6   ;JUMP IF FIRST ARG +
   03350 006300         ASL     R0      ;FLAG FIRST ARG -
   03352 012602         MOV     (SP)+,R2        ;GET HIGH SECOND ARG
   03354 002403         BLT     SMES6   ;JUMP IF BOTH SIGNS -
   03356 000412         BR      NEGS6   ;JUMP IF FIRST - AND SECOND +
   03360 012602 FPSS6:  MOV     (SP)+,R2
   03362 002411         BLT     PLSS6   ;JUMP IF FIRST + AND SECOND -
   03364 020102 SMES6:  CMP     R1,R2   ;COMPARE MAGNITUDES
   03366 001004         BNE     OUTS6   ;JUMP IF DIFFERENT
   03370 026616         CMP     4(SP),@SP       ;COMPARE LOW ORDER
         000004
   03374 001001         BNE     OUTS6   ;JUMP IF DIFFERENT
   03376 005000         CLR     R0      ;FLAG +
   03400 006000 OUTS6:  ROR     R0      ;SAVE C BIT AND TEST SECOND ARG -
   03402 103401         BCS     PLSS6   ;JUMP IF SECOND ARG +
   03404 005400 NEGS6:  NEG     R0      ;REVERSE C BIT
   03406 062706 PLSS6:  ADD     #6,SP   ;POP ARGS
         000006
   03412 005700         TST     R0      ;SET Z AND N BITS CORRECTLY
   03414 000134         JMP     @(R4)+  ;RETURN TO CALLER
                        .ENDC
                        .ENDC
```

```
                            .TITLE  $DBL02
                            .IFDF   CND$7
                    ;
                    ;       DBLE    V002A
                    ;
                    ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                    ;

                            .GLOBL  DBLE
                    ;       THE FORTRAN DBLE FUNCTION
                    ;       CALLING SEQUENCE:
                    ;       JSR     R5,DBLE
                    ;       BR      A
                    ;       .WORD   ARGUMENT ADDRESS
                    ;A:
                    ;       RETURNS THE DOUBLE PRESICION EQUIVALENT
                    ;       OF THE REAL ARGUMENT IN R0 - R3.
                    ;
       000000               R0=%0
       000001               R1=%1
       000002               R2=%2
       000003               R3=%3
       000005               R5=%5
 03416 016502 DBLE:         MOV     2(R5),R2        ;GET ARG ADDRESS
       000002
 03422 012200               MOV     (R2)+,R0        ;GET HIGH ORDER
 03424 011201               MOV     @R2,R1  ;GET LOW ORDER
 03426 005002               CLR     R2      ;CLEAR LOWEST ORDER
 03430 005003               CLR     R3
 03432 000205               RTS     R5      ;RETURN TO CALLER
                            .ENDC
```

```
                             .TITLE  $DCI01
                             .IFDF   CND$8
                ;
                ;            $DCI    V001A
                ;
                ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                ;
                             .GLOBL  $DCI,$RCI
                ;            $DCI --- ASCII TO DOUBLE CONVERSION.
                ;            $RCI --- ASCII TO REAL CONVERSION.
                ;            CALLING SEQUENCE:
                ;            PUSH ADDRESS OF START OF FIELD
                ;            PUSH LENGTH OF FIELD
                ;            PUSH FORMAT SCALE D FROM W.D
                ;            PUSH P FORMAT SCALE
                ;            JSR     PC,$DCI (OR $RCI)

      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 NUMEND=0
      000002 POINTL=2
      000004 DIGITS=4
      000006 BEXP=6
      000010 ESIGN=8.
      000012 SIGN=10.
      000014 EEXP=12.
      000036 P=30.
      000040 D=32.
      000032 ERR=26.
      000042 LENGTH=34.
      000042 TEMP=LENGTH
      000036 RESULT=P
      000044 START=36.
      000044 END=START
03434 005046 $RCI:   CLR     -(SP)   ;CLEAR ERROR FLAG
03436 005216         INC     @SP     ;SET REAL CONVERSION FLAG
03440 000401         BR      CNV$8
03442 005046 $DCI:   CLR     -(SP)   ;CLEAR ERROR FLAG AND SET FOR DOUBLE
03444 010046 CNV$8:  MOV     R0,-(SP)
03446 010146         MOV     R1,-(SP)
03450 010246         MOV     R2,-(SP)
03452 010346         MOV     R3,-(SP)
03454 010446         MOV     R4,-(SP)
03456 010546         MOV     R5,-(SP)
03460 005046         CLR     -(SP)   ;CLEAR EXP
03462 005046         CLR     -(SP)   ;CLEAR SIGN
03464 005046         CLR     -(SP)   ;CLEAR ESIGN
03466 012746         MOV     #65.,-(SP)      ;INITIALIZE BEXP
      000101
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 016505                 MOV     START(SP),R5    ;GET FIELD START ADDRESS
         000044
59 03506 066666                 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      FLO$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 060405             ADD     R4,R5   ;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 FLO$8:      CMP     R5,END(SP)         ;CHECK FOR END OF FIELD
         000044
107 3716 002720             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 021605 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 PTP$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      FLO$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     $TR$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 001362          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 003316         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     DV4S8
221 4430 005304             DEC     R4
222 4432 003345             BGT     DV3S8
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     DIVS8   ;JUMP IF MORE TO DO
227 4454 005366 FLTS8:      DEC     BEXP(SP)        ;POST NORMALIZE THE RESULT
             000006
228 4460 004767             JSR     PC,LFTS8
             000324
229 4464 103373             BCC     FLTS8
230 4466 062766             ADD     #200,BEXP(SP)   ;SET EXCESS 128
             000200
             000006
231 4474 003475             BLE     UNDS8   ;NUMBER TOO SMALL TO REPRESENT
232 4476 026627             CMP     BEXP(SP),#377
             000006
             000377
233 4504 003071             BGT     OVRS8   ;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 156000             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,RITS8        ;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     OVRS8   ;JUMP IF OVERFLOW
253 4562 103442             BCS     OVRS8
254 4564 105766 STRS8:      TSTB    ERF(SP) ;TEST REAL/DOUBLE FLAG
             000032
```

```
255 4570 001407             BEQ     DPRS8   ;JUMP IF DOUBLE
256 4572 006102             ROL     R2      ;ROUND TO REAL PRECISION
257 4574 005501             ADC     R1
258 4576 005500             ADC     R0
259 4600 102433             BVS     OVRS8
260 4602 103432             BCS     OVRS8   ;JUMP IF OVERFLOW ON ROUND
261 4604 010002             MOV     R0,R2   ;MOVE HIGH ORDER RESULT UP
262 4606 010103             MOV     R1,R3
263 4610 010066 DPRS8:      MOV     R0,RESULT(SP)   ;STORE RESULT ON STACK
         000036
264 4614 010166             MOV     R1,RESULT+2(SP)
         000040
265 4620 010266             MOV     R2,RESULT+4(SP)
         000042
266 4624 010366             MOV     R3,RESULT+6(SP)
         000044
267 4630 062706             ADD     #14.,SP ;CLEAR STACK OF JUNK
         000016
268 4634 012605             MOV     (SP)+,R5
269 4636 012604             MOV     (SP)+,R4
270 4640 012603             MOV     (SP)+,R3
271 4642 012602             MOV     (SP)+,R2
272 4644 012601             MOV     (SP)+,R1
273 4646 012600             MOV     (SP)+,R0
274 4650 105716             TSTB    @SP     ;TEST REAL/DOUBLE FLAG
275 4652 001404             BEQ     RRNS8   ;JUMP IF DOUBLE
276 4654 012666             MOV     (SP)+,2(SP)     ;PUSH FLAG UP
         000002
277 4660 012666             MOV     (SP)+,2(SP)     ;PUSH RETURN UP
         000002
278 4664 006126 RRNS8:      ROL     (SP)+   ;FLUSH FLAG AND SET C BIT IF ERROR
279 4666 000207             RTS     PC
280                  ;
281 4670             OVRS8:
282 4670 000167 UNDS8:      JMP     ERRS8
         177200
283                  ;
284 4674 020027 M54S8:      CMP     R0,#146314
         146314
285 4700 103405             BLO     M55S8   ;JUMP IF ROOM FOR 5/4 * FRACT
286 4702 000241             CLC
287 4704 004767             JSR     PC,R1TS8        ;DIVIDE BY 2
         000112
288 4710 005266             INC     BEXP+0+2(SP)    ;MULTIPLY BY 2
         000010
289 4714 010046 M55S8:      MOV     R0,-(SP)
290 4716 010146             MOV     R1,-(SP)
291 4720 010246             MOV     R2,-(SP)
292 4722 010346             MOV     R3,-(SP)
293 4724 000241             CLC
294 4726 004767             JSR     PC,R1TS8        ;HALF
         000070
295 4732 000241             CLC
296 4734 004767             JSR     PC,R1TS8        ;QUARTER
         000062
297 4740 000410             BR      M5AS8   ;GO GET F+F/4
298 4742 010046 ML5S8:      MOV     R0,-(SP)        ;MULT BY 5
```

```
299 4744 010146          MOV      R1,-(SP)
300 4746 010246          MOV      R2,-(SP)
301 4750 010346          MOV      R3,-(SP)
302 4752 004767          JSR      PC,LFTS8          ;DOUBLE
         000032
303 4756 004767          JSR      PC,LFTS8          ;QUADRUPLE
         000026
304 4762 062603 M5AS8:   ADD      (SP)+,R3
305 4764 005502          ADC      R2
306 4766 005501          ADC      R1
307 4770 005500          ADC      R0
308 4772 062602          ADD      (SP)+,R2
309 4774 005501          ADC      R1
310 4776 005500          ADC      R0
311 5000 062601          ADD      (SP)+,R1
312 5002 005500          ADC      R0
313 5004 062600          ADD      (SP)+,R0
314 5006 000207          RTS      PC         ;CODES MAY BE TESTED ON RETURN
315 5010 006303 LFTS8:   ASL      R3
316 5012 006102          ROL      R2
317 5014 006101          ROL      R1
318 5016 006100          ROL      R0
319 5020 000207          RTS      PC
320 5022 006000 RITS8:   ROR      R0
321 5024 006001          ROR      R1
322 5026 006002          ROR      R2
323 5030 006003          ROR      R3
324 5032 000207          RTS      PC
325                      .ENDC
```

```
                    .TITLE  $DC004
                    .IFDF   CND$9
             ;
             ;      $DCO    V004A
             ;
             ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
             ;

                    .GLOBL  $ECO,$FCO,$GCO,$DCO
             ;      $ECO    THE E CONVERSION OUTPUT ROUTINE FOR REALS
             ;      $FCO    THE F CONVERSION OUTPUT ROUTINE FOR REALS
             ;      $GCO    THE G CONVERSION OUTPUT ROUTINE FOR REALS
             ;      $DCO    THE D CONVERSION ROUTINE FOR DOUBLES
             ;      CALLING SEQUENCE:
             ;      PUSH FIELD START
             ;      PUSH FIELD LENGTH
             ;      PUSH D PART OF W.D SPECIFICATION
             ;      PUSH P SCALE
             ;      PUSH VALUE TO BE OUTPUT
             ;      JSR     PC,$ECO     (OR $FCO)  (OR $GCO) (OR $DCO)
             ;      R0, R1, R2, R3 ARE DESTROYED
      000000        R0=%0
      000001        R1=%1
      000002        R2=%2
      000003        R3=%3
      000004        R4=%4
      000005        R5=%5
      000006        SP=%6
      000007        PC=%7
      000002 POINT=2
      000004 BEXP=4
      000006 EEXP=6
      000014 TYPE=12.
      000020 P=16.
      000022 D=18.
      000024 L=20.
      000026 S=22.
05034 012700 $GCO:  MOV     #42403,R0       ;FLAG G FORMAT
      042403
05040 000420        BR      XCO$9
05042 005000 $FCO:  CLR     R0      ;FLAG F FORMAT
05044 000416        BR      XCO$9
05046 012600 $DCO:  MOV     (SP)+,R0        ;POP RETURN
05050 012601        MOV     (SP)+,R1        ;GET HIGHEST ORDER ARG
05052 012602        MOV     (SP)+,R2        ;GET NEXT
05054 011603        MOV     @SP,R3  ;THIRD ARG WORD
05056 012716        MOV     #42002,@SP      ;FLAG D FORMAT
      042002
05062 010446        MOV     R4,-(SP)        ;SAVE R4
05064 016604        MOV     4(SP),R4        ;GET LOWEST ORDER ARG
      000004
05070 010066        MOV     R0,4(SP)        ;SAVE RETURN
      000004
05074 000412        BR      XC1$9
05076 012700 $ECO:  MOV     #42402,R0       ;FLAG E FORMAT
      042402
05102 012603 XCO$9: MOV     (SP)+,R3        ;SAVE RETURN
```

```
53 05104 012601                MOV     (SP)+,R1          ;GET HIGH ORDER ARG
54 05106 012602                MOV     (SP)+,R2          ;GET LOW ORDER ARG
55 05110 010346                MOV     R3,-(SP)          ;PUSH RETURN
56 05112 010046                MOV     R0,-(SP)          ;PUSH TYPE
57 05114 005003                CLR     R3      ;CLEAR LOW ORDER REGISTERS
58 05116 010446                MOV     R4,-(SP)          ;SAVE R4
59 05120 005004                CLR     R4
60 05122 010546 XC1$9:         MOV     R5,-(SP)          ;SAVE R5 AND CONTINUE ALL TYPES
61 05124 005046                CLR     -(SP)   ;CLEAR EXP
62 05126 005046                CLR     -(SP)   ;CLEAR BEXP
63 05130 024646                CMP     -(SP),-(SP)       ;ROOM FOR POINT AND SIGN
64 05132 066666                ADD     S(SP),L(SP)       ;POINT 1 BEYOND END OF FIELD
         000026
         000024
65 05140 016600                MOV     S(SP),R0
         000026
66 05144 112720 CLE$9:         MOVB    #' ,(R0)+         ;BLANK OUT FIELD
         000040
67 05150 020066                CMP     R0,L(SP)
         000024
68 05154 103773                BLO     CLE$9
69 05156 006101                ROL     R1      ;GET ARG SIGN
70 05160 006116                ROL     @SP     ;SAVE IT
71 05162 000301                SWAB    R1
72 05164 110166                MOVB    R1,BEXP(SP)       ;GET BINARY EXPONENT
         000004
73 05170 001002                BNE     NNZ$9   ;JUMP IF ARG NOT 0
74 05172 005000                CLR     R0;               CLEAR OVERFLOW ACCUMULATOR
75 05174 000502                BR      NOD$9;            GO PRINT THE 0 IN FORMAT
76                      ;
77 05176 000261 NNZ$9:         SEC     ;INSERT NORMAL BIT
78 05200 006001                ROR     R1
79 05202 105001                CLRB    R1      ;LEFT JUSTIFY FRACTION
80 05204 000302                SWAB    R2
81 05206 150201                BISB    R2,R1
82 05210 105002                CLRB    R2
83 05212 000303                SWAB    R3
84 05214 150302                BISB    R3,R2
85 05216 105003                CLRB    R3
86 05220 000304                SWAB    R4
87 05222 150403                BISB    R4,R3
88 05224 105004                CLRB    R4
89 05226 162766                SUB     #200,BEXP(SP)     ;REMOVE EXCESS 128 FROM BINARY E
         000200
         000004
90 05234 002424                BLT     DIV$9   ;JUMP IF BINARY EXPONENT NEG
91 05236 001447                BEQ     NOM$9   ;JUMP IF NO SCALING TO DO
92 05240 005701 MUL$9:         TST     R1      ;BINARY EXPONENT IS POSITIVE
93 05242 002410                BLT     ML1$9   ;JUMP IF FRACTION OVERFLOW IMPENDING
94 05244 006304                ASL     R4      ;DOUBLE FRACTION
95 05246 006103                ROL     R3
96 05250 006102                ROL     R2
97 05252 006101                ROL     R1
98 05254 005366                DEC     BEXP(SP)          ;COMPENSATE EXPONENT
         000004
99 05260 003367                BGT     MUL$9   ;JUMP IF MORE BINARY SCALING TO DO
100 5262 000435                BR      NOM$9
```

```
5264 004767 ML1$9:  JSR     PC,M45$9        ;GET 4/5 FRACTION
     000646
5270 005266         INC     EEXP(SP)        ;MULTIPLY BY 10
     000006
5274 162766         SUB     #3,BEXP(SP)     ;AND DIVIDE BY 8
     000003
     000004
5302 003356         BGT     MUL$9   ;JUMP IF BINARY EXPONENT STILL POS.
5304 001424         BEQ     NOM$9   ;JUMP IF EXPONENT GONE NOW
5306 020127 DIV$9:  CMP     R1,#146314      ;BINARY EXPONENT IS NEGATIVE
     146314
5312 103014         BHIS    DV1$9   ;JUMP IF NO ROOM FOR 5/4 FRACTION
5314 026627         CMP     BEXP(SP),#-3
     000004
     177775
5322 003010         BGT     DV1$9   ;JUMP IF NOT ENOUGH BINARY EXP LEFT
5324 004767         JSR     PC,M54$9        ;MULTIPLY FRACTION BY 5/4
     000520
5330 005366         DEC     EEXP(SP)        ;DIVIDE BY 10
     000006
5334 062766         ADD     #2,BEXP(SP)     ;MULTIPLY BY 4
     000002
     000004
5342 000402         BR      DV2$9
5344 004767 DV1$9:  JSR     PC,RIT$9        ;DIVIDE BY 2
     001264
5350 005266 DV2$9:  INC     BEXP(SP)        ;MULTIPLY BY 2
     000004
5354 001354         BNE     DIV$9   ;HIT IT AGAIN IF BIN.EXP. NOT GONE
            ;       AT THIS POINT THE BINARY EXPONENT IS 0
            ;       AND THE FRACTION IS IN R1, R2, R3 AND R4.
5356 005000 NOM$9:  CLR     R0      ;CLEAR OVERFLOW ACCUMULATOR
5360 004767 NO1$9:  JSR     PC,M54$9        ;MULTIPLY FRACTION BY 5/4
     000464
5364 004767         JSR     PC,ML8$9        ;AND NOW BY 8
     000656
5370 005700         TST     R0
5372 001003         BNE     NOD$9   ;JUMP IF AN INTEGER PART RESULTS
5374 005366         DEC     EEXP(SP)        ;DECREMENT EXPONENT
     000006
5400 000767         BR      NO1$9   ;GO AGAIN TO GET AN INTEGER PART
            ;       AT THIS POINT THE MOST SIGNIFICANT NON ZERO DIGIT IS IN
5402 105766 NOD$9:  TSTB    TYPE(SP)        ;TEST CONVERSON TYPE
     000014
5406 001424         BEQ     FFT$9   ;JUMP IF F FORMAT
5410 106066         RORB    TYPE(SP)
     000014
5414 103114         BCC     EFT$9   ;JUMP IF E FORMAT OR D FORMAT
5416 005766         TST     EEXP(SP)        ;G FORMAT
     000006
5422 002511         BLT     EFT$9   ;JUMP IF RESULT <.1
5424 026666         CMP     EEXP(SP),D(SP)
     000006
     000022
5432 003105         BGT     EFT$9   ;JUMP IF RESULT >10**D
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 166066           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,POINT(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 166005 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
         000016
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 006004 002402           BLT     EX2$9
200 006006 105213           INCB    @R3     ;ACCUMULATE TENS
201 006010 000773           BR      EX3$9
202 006012 062704 EX2$9:    ADD     #72,R4  ;GET POSITIVE UNITS
           000072
203 006016 110463           MOVB    R4,1(R3)        ;MOVE UNITS OUT
           000001
204 006022 062706 ONE$9:    ADD     #8.,SP
           000010
205 006026 012605           MOV     (SP)+,R5
206 006030 012604           MOV     (SP)+,R4
207 006032 012666           MOV     (SP)+,6(SP)     ;MOVE FLAG AND RETURN UP
           000006
208 006036 012666           MOV     (SP)+,6(SP)
           000006
209 006042 022626           CMP     (SP)+,(SP)+     ;FLUSH JUNK
210 006044 006126           ROL     (SP)+   ;SET C BIT IF ERROR
211 006046 000207           RTS     PC      ;RETURN TO CALLER
212
213                   ;     MULTIPLY CONTENTS OF R1 ... R4 BY 5/4.
214                   ;     ANY OVERFLOW GOES INTO R0.
215 006050 010146 M54$9:    MOV     R1,-(SP)        ;5/4X=X+X/4
216 006052 010246           MOV     R2,-(SP)
217 006054 010346           MOV     R3,-(SP)
218 006056 010446           MOV     R4,-(SP)
219 006060 004767           JSR     PC,RIT$9        ;X/2
           000550
220 006064 004767           JSR     PC,RIT$9        ;X/4
           000544
221 006070 005504           ADC     R4      ;ROUND
222 006072 005503           ADC     R3
223 006074 005502           ADC     R2
224 006076 005501           ADC     R1
225 006100 062604           ADD     (SP)+,R4
226 006102 005503           ADC     R3
227 006104 005502           ADC     R2
228 006106 005501           ADC     R1
229 006110 005504           ADC     R0
230 006112 062603           ADD     (SP)+,R3
231 006114 005502           ADC     R2
232 006116 005501           ADC     R1
233 006120 005500           ADC     R0
234 006122 062602           ADD     (SP)+,R2
235 006124 005501           ADC     R1
236 006126 005500           ADC     R0
237 006130 062601           ADD     (SP)+,R1
238 006132 005500           ADC     R0
239 006134 000207           RTS     PC      ;RETURN TO CALLER
240                   ;
241                   ;
242 006136 012705 M45$9:    MOV     #16.,R5 ;MULTIPLY R1...R4 BY 4/5
           000020
243 006142 004767           JSR     PC,RIT$9
           000466
244 006146 010446           MOV     R4,-(SP)
245 006150 010346           MOV     R3,-(SP)
246 006152 010246           MOV     R2,-(SP)
```

```
247 6154 010146                 MOV     R1,-(SP)
248 6156 004767 M51$9:          JSR     PC,RITS9
         000452
249 6162 004767                 JSR     PC,RITS9
         000446
250 6166 012700                 MOV     #2,R0
         000002
251 6172 004767 M52$9:          JSR     PC,RITS9
         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
```

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

```
336 6500 062700 RU3$9:  ADD     #5,R0   ;ROUND MOST SIGNIFICANT DIGIT
         000005
337 6504 000767         BR      RU2$9
338                     ;
339                     ;       INSERT A - IF NECESSARY AND CHECK THAT THE FIELD
340                     ;       IS WIDE ENOUGH TO CONTAIN THE RESULT.
341 6506 020566 ISN$9:  CMP     R5,S-0+2(SP)    ;COMPARE SIGN SLOT WITH FIELD BE
         000030
342 6512 103407         BLO     SPC$9   ;JUMP IF IT MAY NOT FIT
343 6514 006066         ROR     0+2(SP) ;TEST SIGN
         000002
344 6520 103002         BCC     ISR$9   ;JUMP IF +
345 6522 112715         MOVB    #'-,@R5 ;INSERT -
         000055
346 6526 005205 ISR$9:  INC     R5      ;POINT TO LEADING DIGIT SLOT
347 6530 000207         RTS     PC      ;RETURN
348 6532 006066 SPC$9:  ROR     0+2(SP) ;TEST SIGN
         000002
349 6536 103657         BCS     ERR$9   ;JUMP IF IT'S - 'CAUSE THERE ISN'T ROOM
350 6540 005205         INC     R5      ;POINT TO LEADING DIGIT SLOT
351 6542 020566         CMP     R5,S+2(SP)
         000030
352 6546 103653         BLO     ERR$9   ;JUMP IF NO ROOM FOR IT EITHER
353 6550 000207         RTS     PC
354                     ;
355                     ;       EXTRACT LEADING DIGITS FROM R0 ... R4 AND FILL IN
356                     ;       THE AREA STARTING AT THE ADDRESS IN R5 AND
357                     ;       BOUNDED BY THE MODIFIED FIELD END.
358 6552 022700 DG$$9:  CMP     #10.,R0 ;CHECK IF OVERFLOW IN R0
         000012
359 6556 003004         BGT     DG1$9   ;JUMP IF ONLY ONE DIGITS WORTH
360 6560 112725         MOVB    #'1,(R5)+       ;OUTPUT OVERFLOW
         000061
361 6564 162700         SUB     #10.,R0 ;CORRECT R0 FOR NEXT DIGIT
         000012
362 6570 026505 DG1$9:  CMP     POINT+2(SP),R5  ;CHECK FOR . SLOT
         000004
363 6574 001002         BNE     DG2$9
364 6576 112725         MOVB    #'.,(R5)+       ;INSERT THE .
         000056
365 6602 026505 DG2$9:  CMP     L+2(SP),R5      ;CHECK END OF FIELD
         000026
366 6606 101411         BLOS    DIG$9   ;JUMP IF DONE
367 6610 062700 DG3$9:  ADD     #60,R0  ;CONVERT TO ASCII
         000060
368 6614 110025         MOVB    R0,(R5)+        ;PUT IT IN FIELD
369 6616 005000         CLR     R0
370 6620 004767         JSR     PC,M54$9        ;MULTIPLY FRACTION BY 5/4
         177224
371 6624 004767         JSR     PC,ML8$9        ;AND BY 8
         177416
372 6630 000757         BR      DG1$9   ;GO CONVERT TO ASCII
373 6632 000207 DIG$9:  RTS     PC      ;RETURN TO CALLER
374                     ;
375                     ;       SHIFT THE CONTENTS OF R1 .. R4 RIGHT 1.
376 6634 000241 RIT$9:  CLC
377 6636 006001         ROR     R1
```

```
378 6640 006002          ROR     R2
379 6642 006003          ROR     R3
380 6644 006004          ROR     R4
381 6646 000207          RTS     PC
382                      .ENDC
```

```
                           .TITLE  $DLG03
                           .IFDF   CND$10
                  ;
                  ;        DLOG     V003A
                  ;
                  ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                  ;
                           .GLOBL  DLOG,DLOG10,$ERR;
                           .IFNDF  FPU
                           .GLOBL  $POLSH,$ADD,$SBD,$MLD,$DVD,$ID,$POPR4;
                           .ENDC
                  ;        THE FORTRAN DLOG AND DLOG10 FUNCTIONS
                  ;        CALLING SEQUENCE:
                  ;        JSR     R5,DLOG (OR DLOG10)
                  ;        BR      A
                  ;        .WORD   ARGUMENT ADDRESS
                  ;A:
                  ;        RETURNS LN(ARG) (OR LOG10(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
       000007              PC=%7
       000000              F0=%0
       000001              F1=%1
       000002              F2=%2
       000003              F3=%3
                           .IFNDF  FPU
 06650 011746 DLOG10:      MOV     @PC,-(SP)          ;GET 0004XX AS A FLAG
 06652 000401              BR      LOG$10
 06654 005046 DLOG:        CLR     -(SP)    ;FLAG DLOG
 06656 010546 LOG$10:      MOV     R5,-(SP)           ;SAVE RETURN POINTER
 06660 016504              MOV     2(R5),R4           ;GET ARG ADDRESS
       000002
 06664 062704              ADD     #8.,R4  ;POINT TO LEAST SIGNIFICANT PART
       000010
 06670 012746              MOV     #147572,-(SP)
       147572
 06674 012746              MOV     #173721,-(SP)
       173721
 06700 012746              MOV     #071027,-(SP)      ;PUSH -1/2*LN(2)
       071027
 06704 012746              MOV     #137061,-(SP)
       137061
 06710 162706              SUB     #8.,SP  ;GET WORK SPACE
       000010
 06714 014446              MOV     -(R4),-(SP)        ;GET ARG
 06716 014446              MOV     -(R4),-(SP)
 06720 014446              MOV     -(R4),-(SP)
 06722 014446              MOV     -(R4),-(SP)
 06724 003501              BLE     ERR$10  ;JUMP IF NOT POSITIVE
 06726 006316              ASL     @SP
 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,DUP$10,$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'XPU$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
```

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

```
114 7256 016646            MOV     6(SP),-(SP)
         000006
115 7262 016646            MOV     6(SP),-(SP)
         000006
116 7266 000134            JMP     @(R4)+
117                 ;
118 7270 012746 PL2$10:    MOV     #147572,-(SP)
         147572
119 7274 012746            MOV     #173721,-(SP)
         173721
120 7300 012746            MOV     #071027,-(SP)    ;PUSH LN(2)
         071027
121 7304 012746            MOV     #040061,-(SP)
         040061
122 7310 000134            JMP     @(R4)+
123                 ;
124 7312 105366 EXI$10:    DECB    11.(SP) ;CHECK FOR ALOG10
         000013
125 7316 002411            BLT     LGT$10  ;NO, DONE
126 7320 012746            MOV     #024162,-(SP)
         024162
127 7324 012746            MOV     #124467,-(SP)
         124467
128 7330 012746            MOV     #055730,-(SP)    ;PUSH LOG10(E)
         055730
129 7334 012746            MOV     #037736,-(SP)
         037736
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     #FC0$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;
```

```
                     MULD    F1,F1;          Y= W**2
             ;
                     MOV     #6,R1;          COUNT CONSTANTS FOR POLYNOMIAL
                     LDD     (R0)+,F0;               INITIALIZE ACCUMULATOR
             XPD$10: MULD    F1,F0;
                     DEC     R1;             COUNT
                     ADDD    (R0)+,F0;       F0:= Y*F0 + C(I)
                     BGT     XPD$10;         LOOP
                     MULD    F2,F0;
                     ADDD    (R0)+,F0;       F0:= W*F0 - 1/2*LN(2)
                     ADDD    F3,F0;          ADD SCALE FACTOR FOR EXPONENT
                     TST     R4;             TEST DLOG10 FLAG
                     BEQ     LGT$10;
                     MULD    (R0),F0;        DLOG10 = DLOG*LOG10(E)
             ;
             LGT$10: STD     F0,-(SP);       MOVE RESULT TO STACK
                     MOV     (SP)+,R0;       AND TENCE TO R0...R3
                     MOV     (SP)+,R1;
                     MOV     (SP)+,R2;
                     MOV     (SP)+,R3;
                     RTS     R5;             EXIT
             ;
             ERR$10: JSR     R5,$ERR;        ERROR 4,3
                     RTS     R5;             EXIT - NO STACK CLEANUP REQUIRED
                     .BYTE   4
                     .BYTE   3
             ;
             ;       ORDER-DEPENDENT CONSTANTS FOR ROUTINE
             ;       R0 POINTS AT CURRENT CONSTANT IN FPU VERSION
             ;
             FCO$10: .WORD   040061,071027;          LN(2)
                     .WORD   173721,147572;
             ;
                     .WORD   040065,002363;  1/2*SQRT(2)
                     .WORD   031771,137145;
                     .ENDC
             ;
7360 037455          .WORD   037455,106270   ;.16948212488
7362 106270
7364 157166          .WORD   157166,174770
7366 174770
             ;
7370 037471          .WORD   037471,072731   ;.1811136267967
7372 072731
7374 137716          .WORD   137716,117115
7376 117115
             ;
7400 037543          .WORD   037543,111153   ;.22223823332791
7402 111153
7404 060101          .WORD   060101,135465
7406 135465
             ;
7410 037622          .WORD   037622,044436   ;.2857140915904889
7412 044436
7414 007306          .WORD   007306,063062
7416 063062
             ;
```

```
209 7420 037714           .WORD   037714,146314   ;.40000000120604 5365
    7422 146314
210 7424 153450           .WORD   153450,165773
    7426 165773
211                  ;
212 7430 040052           .WORD   040052,125252   ;.66666666066633660894
    7432 125252
213 7434 125247           .WORD   125247,004643
    7436 004643
214                  ;
215 7440 040400 CONS10:   .WORD   040400,000000   ;2.00000000000000261
    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
```

```
                               .TITLE  $DNT02
                               .IFDF   CND$11
                        ;
                        ;      $DINT   V002A
                        ;
                               .GLOBL  $DINT
                        ;      COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        ;      OTS INTERNAL FUNCTION TO FIND THE INTEGER
                        ;      PART OF A DOUBLE PRECISION NUMBER.
                        ;      CALLED IN THE POLISH MODE.
        000000                 R0=%0
        000001                 R1=%1
        000002                 R2=%2
        000003                 R3=%3
        000004                 R4=%4
        000005                 R5=%5
        000006                 SP=%6
        177304                 MQ=177304
        177316                 ASH=177316
        000000                 F0=%0
        000001                 F1=%1
                               .IFDF   FPU
                $DINT:         .WORD   170011  ;;SETD
                               .WORD   172426  ;;LDD   (SP)+,F0         ;LOAD ARG
                               .WORD   171467,4        ;;MODD  ONE,F0   ;GET INTEGER PAR
                               .WORD   174146  ;;STD   F1,-(SP)         ;PUSH INTEGER
                               JMP     @(R4)+  ;RETURN TO CALLER
                               .WORD   040200,0,0,0    ;FLOATING 1.
                               .ENDC
                               .IFNDF  FPU
 07450  012600  $DINT:         MOV     (SP)+,R0        ;POP DOUBLE ARG
 07452  012601                 MOV     (SP)+,R1
 07454  012602                 MOV     (SP)+,R2
 07456  012603                 MOV     (SP)+,R3
 07460  010446                 MOV     R4,-(SP)
 07462  010546                 MOV     R5,-(SP)
 07464  010004                 MOV     R0,R4   ;GET EXPONENT
 07466  006104                 ROL     R4
 07470  105004                 CLRB    R4
 07472  000304                 SWAB    R4
 07474  162704                 SUB     #270,R4 ;CONVERT TO -SHIFT COUNT
        000270
 07500  002041                 BGE     DNE$11  ;JUMP IF ARG MUST BE INTEGER ALREADY
 07502  022704                 CMP     #-70,R4
        177710
 07506  002405                 BLT     SHF$11  ;JUMP TO GET INTEGER PART
 07510  005000                 CLR     R0      ;ANSWER IS 0
 07512  005001                 CLR     R1
 07514  005002  C23$11:        CLR     R2
 07516  005003                 CLR     R3
 07520  000431                 BR      DNE$11
 07522          SHF$11:
                               .IFNDF  EAE&MULDIV
 07522  010405                 MOV     R4,R5   ;SAVE A COPY OF SHIFT COUNT
 07524  022704                 CMP     #-32.,R4        ;CHECK LOW OR HIGH TRUNCATION
        177740
 07530  002415                 BLT     RUR$11
```

```
55 07532 001770          BEQ     C23$11  ;GO CLEAR LOW ORDER HALF
56 07534 062704          ADD     #32.,R4 ;DO HIGH ORDER
         000040
57 07540 010405          MOV     R4,R5
58 07542 006000 RR1$11:  ROR     R0       ;SHIFT OUT FRACTION BITS
59 07544 006001          ROR     R1
60 07546 005204          INC     R4
61 07550 002774          BLT     RR1$11
62 07552 006301 AS1$11:  ASL     R1       ;SHIFT IN 0'S
63 07554 006100          ROL     R0
64 07556 005205          INC     R5
65 07560 002774          BLT     AS1$11
66 07562 000754          BR      C23$11  ;GO CLEAR LOW ORDER
67 07564 006002 ROR$11:  ROR     R2       ;MOVE OUT FRACTION BITS
68 07566 006003          ROR     R3
69 07570 005204          INC     R4       ;COUNT LOOP
70 07572 002774          BLT     ROR$11
71 07574 006303 ASL$11:  ASL     R3
72 07576 006102          ROL     R2
73 07600 005205          INC     R5       ;COUNT LOOP
74 07602 002774          BLT     ASL$11
75                       .ENDC
76                       .IFDF   EAE
77                       MOV     #MQ,R5  ;POINT TO MQ
78                       .ENDC
79                       .IFDF   MULDIV!EAE
80                       CMP     #-32.,R4         ;CHECK FOR HIGH OR LOW ORDER TRU
81                       BLT     R23$11  ;LOW
82                       BEQ     C23$11  ;CLEAR LOW ORDER
83              R01$11:  ADD     #32.,R4 ;HIGH ORDER PARTS
84                       .IFDF   MULDIV
85                       .WORD   073004   ;;ASHC  R4,R0   ;SHIFT OUT FRACTION
86                       NEG     R4       ;SET TO SHIFT LEFT
87                       .WORD   073004   ;;ASHC  R4,R0   ;BRING IN THE 0'S
88                       .ENDC
89                       .IFDF   EAE
90                       MOV     R1,@R5  ;HIGH ORDER TO AC,MQ
91                       MOV     R0,-(R5)
92                       MOV     R4,@#ASH         ;SHIFT RIGHT
93                       NEG     R4
94                       MOV     R4,@#ASH         ;SHIFT LEFT
95                       MOV     (R5)+,R0         ;RESULT TO REGS
96                       MOV     @R5,R1
97                       .ENDC
98                       BR      C23$11  ;GO CLEAR LOW ORDER
99                       .IFDF   MULDIV
100             R23$11:  .WORD   073204   ;;ASHC  R4,R2
101                      NEG     R4
102                      .WORD   073204   ;;ASHC  R4,R2   ;SHIFT IN 0'S
103                      .ENDC
104                      .IFDF   EAE
105             R23$11:  MOV     R3,@R5  ;LOW ORDER TO AC,MQ
106                      MOV     R2,-(R5)
107                      MOV     R4,@#ASH         ;DUMP BITS
108                      NEG     R4
109                      MOV     R4,@#ASH         ;BRING IN 0'S
110                      MOV     (R5)+,R2         ;RESULT TO REGS
```

```
111                           MOV     @R5,R3
112                           .ENDC
113                           .ENDC
114 76Ø4 Ø126Ø5 DNES11:       MOV     (SP)+,R5
115 76Ø6 Ø126Ø4               MOV     (SP)+,R4
116 761Ø Ø1Ø346               MOV     R3,-(SP)          ;PUSH RESULT
117 7612 Ø1Ø246               MOV     R2,-(SP)
118 7614 Ø1Ø146               MOV     R1,-(SP)
119 7616 Ø1ØØ46               MOV     RØ,-(SP)
12Ø 762Ø ØØØ134               JMP     @(R4)+   ;RETURN
121                           .ENDC
122                           .ENDC
```

```
                           .TITLE  $DR02
                           .IFDF   CND$12
                           .GLOBL  $DR,$ERR
                   ;       $DR     THE DOUBLE PRECISION TO REAL CONVERTER
                   ;
                   ;       $DR     V002A
                   ;
                   ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                   ;       ROUND THE TOP STACK ITEM TO REAL FORMAT.
         000004            R4=%4
         000005            R5=%5
         000006            SP=%6
         000000            F0=%0
                           .IFDF   FPU
                   $DR:    .WORD   170001  ;;SETF
                           .WORD   177426  ;;LDCDF (SP)+,F0        ;CONVERT ARG
                           .WORD   170000  ;;CFCC  ;GET CONDITION CODES
                           BVS     OV1$12  ;JUMP IF OVERFLOW ON ROUND
                           .WORD   174046  ;;STF   F0,-(SP)
                           JMP     @(R4)+;         !!!
                           .ENDC
                           .IFNDF  FPU
   07622 006166    $DR:    ROL     4(SP)   ;ROUND LOW ORDER PART
         000004
   07626 005566            ADC     2(SP)
         000002
   07632 005516            ADC     @SP
   07634 103406            BCS     OVR$12  ;JUMP IF OVERFLOW
   07636 102405            BVS     OVR$12
   07640 012666    DR1$12: MOV     (SP)+,2(SP)     ;MOVE HIGHEST ORDER PART
         000002
   07644 012666            MOV     (SP)+,2(SP)     ;MOVE LOW ORDER REAL
         000002
   07650 000134            JMP     @(R4)+  ;RETURN
   07652 022626    OVR$12: CMP     (SP)+,(SP)+     ;FLUSH ARG
   07654 022626            CMP     (SP)+,(SP)+
                           .ENDC
   07656 004567    OV1$12: JSR     R5,$ERR ;ERROR 3,23
         012124
   07662 000401            BR      DR2$12
   07664    003            .BYTE   3
   07665    027            .BYTE   23.
   07666 005046    DR2$12: CLR     -(SP)   ;RETURN 0.
   07670 005046            CLR     -(SP)
   07672 000134            JMP     @(R4)+
                           .ENDC
```

```
                     .TITLE  $DSN04
                     .IFDF   CND$13
             ;
             ;       DSINCS  V004A
             ;
             ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
             ;

                     .GLOBL  DSIN,DCOS;
                     .IFNDF  FPU
                     .GLOBL  $ADD,$SBD,$MLD,$DVD,$DINT,$POLSH,$POPR4;
                     .ENDC
             ;       DSIN    DCOS    THE DOUBLE PRECISION SIN AND COS
             ;       FUNCTIONS.
             ;       CALLING SEQUENCE:
             ;       JSR     R5,DSIN (OR DCOS)
             ;       BR      A
             ;       .WORD   ARG ADDRESS
             ;A:
             ;       RETURNS SIN UP COS OF ARG IN R0 - R3.
      000000         R0=%0
      000001         R1=%1
      000002         R2=%2
      000003         R3=%3
      000004         R4=%4
      000005         R5=%5
      000006         SP=%6
      000007         PC=%7
      000000         F0=%0
      000001         F1=%1
      000002         F2=%2
      000003         F3=%3
                     .IFNDF  FPU
07674 010546 DCOS:   MOV     R5,-(SP)          ;SAVE RETURN POINTER
07676 016504         MOV     2(R5),R4          ;GET ARGUMENT ADDRESS
      000002
07702 005046         CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
07704 016446         MOV     6(R4),-(SP)
      000006
07710 016446         MOV     4(R4),-(SP)
      000004
07714 016446         MOV     2(R4),-(SP)       ;PUSH ARGUMENT
      000002
07720 011446         MOV     @R4,-(SP)
07722 012746         MOV     #064302,-(SP)
      064302
07726 012746         MOV     #121041,-(SP)
      121041
07732 012746         MOV     #007732,-(SP)     ;PUSH PI/2
      007732
07736 012746         MOV     #040311,-(SP)
      040311
07742 004467         JSR     R4,$POLSH         ;ENTER POLISH MODE
      011676
07746 000704'        .WORD   $ADD,$NC$13       ;COS(X)=SIN(X+PI/2)
07750 010000'
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 SNC$13:      ASL     @SP      ;CLEAR SIGN AND SAVE IT
55 10002 006066              ROR     8.(SP)   ;IN QUADRANT FLAG
         000010
56 10006 006016              ROR     @SP
57 10010 012746              MOV     #064302,-(SP)
         064302
58 10014 012746              MOV     #121041,-(SP)
         121041
59 10020 012746              MOV     #007732,-(SP)    ;PUSH 2*PI
         007732
60 10024 012746              MOV     #040711,-(SP)
         040711
61 10030 004467              JSR     R4,$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'XPU$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
```

```
10136 010140'          .WORD   RTN$13
10140 005726 RTN$13:   TST     (SP)+    ;POP QUADRANT FLAG
10142 002002           BGE     RT1$13  ;JUMP IF ARGUMENT WAS +
10144 062700           ADD     #100000,R0      ;SIN(-X)=-SIN(X)
      100000
10150 012605 RT1$13:   MOV     (SP)+,R5
10152 000205           RTS     R5      ;BACK TO CALLER
             ;
10154 016646 DUP$13:   MOV     6(SP),-(SP)     ;DUPLICATE STACK ITEM
      000006
10160 016646           MOV     6(SP),-(SP)
      000006
10164 016646           MOV     6(SP),-(SP)
      000006
10170 016646           MOV     6(SP),-(SP)
      000006
10174 000134           JMP     @(R4)+
             ;
10176 005716 X4$13:    TST     @SP     ;CHECK FOR 0 FRACTION
10200 001403           BEQ     ZER$13; QUIT NOW
10202 105266           INCB    1(SP)   ;QUADRUPLE STACK ITEM
      000001
10206 000134           JMP     @(R4)+
10210 012704 ZER$13:   MOV     #PR4$13,R4;      RETURN ZERO RESULT
      010134'
10214 000134           JMP     @(R4)+;          USE POLISH
             ;
 0216 051666 QUD$13:   BIS     @SP,16.(SP)     ;SAVE QUADRANT NUMBER
      000020
 0222 000134           JMP     @(R4)+
             ;
 0224 105766 QST$13:   TSTB    8.(SP)  ;TEST QUADRANT
      000010
 0230 001415           BEQ     Q13$13  ;JUMP IF FIRST OR THIRD QUAD
 0232 062716           ADD     #100000,@SP     ;NEGATE STACK ITEM
      100000
 0236 005046           CLR     -(SP)
 0240 005046           CLR     -(SP)
 0242 005046           CLR     -(SP)   ;PUSH A FLOATING 1.
 0244 012746           MOV     #40200,-(SP)
      040200
 0250 004467           JSR     R4,$POLSH       ;ENTER POLISH
      011370
 0254 000704'          .WORD   $ADD,QSR$13     ;X*1.-X
 0256 010260'
 0260 012704 QSR$13:   MOV     #QSE$13,R4      ;POINT BACK INTO LIST
      010060'
 0264 106266 Q13$13:   ASRB    9.(SP)  ;TEST QUADRANT
      000011
             ;
 0270 103002           BCC     QUT$13  ;JUMP IF FIRST OR SECOND
 0272 062716           ADD     #100000,@SP     ;NEGATE STACK ITEM
      100000
 0276 000134 QUT$13:   JMP     @(R4)+
             ;
 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
```

```
                         BEQ     RTNS13/
                         NEGD    FØ/               SIN(=X) = =SIN(X)
                RTNS13!  STD     FØ,=(SP)/         MOVE RESULT TO STACK
                         MOV     (SP)+,RØ/                 AND THENCE TO RØ...R3
                         MOV     (SP)+,R1/
                         MOV     (SP)+,R2/
                         MOV     (SP)+,R3/
                         RTS     R5/               EXIT
                /
                PI2S13!  .WORD   Ø4Ø311,ØØ7732/    PI/2
                         .WORD   121Ø41,Ø643Ø2/
                /
                /        ORDER=DEPENDENT CONSTANTS
                /
                FCUS13!  .WORD   Ø4Ø711,ØØ7732/    2*PI
                         .WORD   121Ø41,Ø643Ø2/
                         .ENDC
    Ø344 Ø26716          .WORD   Ø26716,1Ø67Ø3     /.5870610981 71E=11
    Ø346 1Ø67Ø3
    Ø35Ø Ø45277          .WORD   Ø45277,146362
    Ø352 146362
                /
    Ø354 13Ø467          .WORD   13Ø467,136273     /=.66843217206396E=9
    Ø356 136273
    Ø36Ø 1Ø3Ø54          .WORD   1Ø3Ø54,123153
    Ø362 123153
                /
    Ø364 Ø32164          .WORD   Ø32164,Ø74657     /.56921348727190 23E=7
    Ø366 Ø74657
    Ø37Ø Ø47254          .WORD   Ø47254,154742
    Ø372 154742
                /
    Ø374 133561          .WORD   133561,1Ø1646     /=.35988430072Ø8693E=5
    Ø376 1Ø1646
    Ø4ØØ 167216          .WORD   167216,134Ø16
    Ø4Ø2 134Ø16
                /
    Ø4Ø4 Ø35Ø5Ø          .WORD   Ø35Ø5Ø,Ø36Ø32     /.16044118470 68221E=3
    Ø4Ø6 Ø36Ø32
    Ø41Ø Ø41214          .WORD   Ø41214,1Ø3131
    Ø412 1Ø3131
                /
    Ø414 136231          .WORD   136231,Ø64546     /=.46817541353Ø2643E=2
    Ø416 Ø64546
    Ø42Ø Ø71423          .WORD   Ø71423,125Ø24
    Ø422 125Ø24
                /
    Ø424 Ø37243          .WORD   Ø37243,Ø32743     /.79692626246 16544E=1
    Ø426 Ø32743
    Ø43Ø Ø35655          .WORD   Ø35655,Ø51557
    Ø432 Ø51557
                /
    Ø434 14ØØ45          .WORD   14ØØ45,Ø56747     /=.64596409750 62462
    Ø436 Ø56747
    Ø44Ø Ø3Ø455          .WORD   Ø3Ø455,171222
    Ø442 171222
                /
```

```
217 0444 040311 CONS13: .WORD   040311,007732   ;1.570796326794897
    0446 007732
218 0450 121041         .WORD   121041,064302
    0452 064302
219               ;
220                     .ENDC
```

```
                              .TITLE  $DSQ03
                              .IFDF   CND$14
                      ;
                      ;       DSQRT   V003A
                      ;
                      ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                      ;

                              .GLOBL  DSQRT,$ERR;
                              .IFNDF  FPU
                              .GLOBL  $ADD,$DVD,$POLSH;
                              .ENDC
                      ;       $DSQRT  THE DOUBLE PRECISION SQUARE ROOT FUNCTION
                      ;       CALLING SEQUENCE:
                      ;       JSR     R5,$DSQRT
                      ;       BR      A
                      ;       #ARG
                      ;A:
                      ;       RETURNS DSQRT IN R0 - R3.
                      ;
       000000                 R0=%0
       000001                 R1=%1
       000002                 R2=%2
       000003                 R3=%3
       000004                 R4=%4
       000005                 R5=%5
       000000                 F0=%0
       000001                 F1=%1
       000002                 F2=%2
       000006                 SP=%6
                              .IFNDF  FPU
10454  010546 DSQRT:          MOV     R5,-(SP)
10456  016505                 MOV     2(R5),R5          ;GET ARGUMENT ADDRESS
       000002
10462  011501                 MOV     @R5,R1  ;GET HIGH ORDER ARGUMENT
10464  100467                 BMI     ERR$14  ;ERROR IF ARGUMENT NEGATIVE
10466  001472                 BEQ     ZER$14  ;FAST EXIT IF ZERO
10470  016502                 MOV     2(R5),R2
       000002
10474  012746                 MOV     #4,-(SP)          ;PUSH ITERATION COUNT
       000004
10500  006201                 ASR     R1      ;FORM INITIAL ESTIMATE
10502  006002                 ROR     R2
10504  062701                 ADD     #20100,R1
       020100
10510  005046                 CLR     -(SP)
10512  005046                 CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
10514  010246                 MOV     R2,-(SP)
10516  010146                 MOV     R1,-(SP)          ;'CAUSE ADD AND DIVIDE ARE
10520  005046                 CLR     -(SP)   ;FASTER THAT WAY
10522  005046                 CLR     -(SP)
10524  016546                 MOV     2(R5),-(SP)
       000002
10530  011546                 MOV     @R5,-(SP)
10532  005046                 CLR     -(SP)
10534  005046                 CLR     -(SP)
10536  010246                 MOV     R2,-(SP)
```

```
10540 010146                  MOV     R1,-(SP)
10542 004467 LUP$14: JSR     R4,$POLSH           ;ENTER POLISH MODE
      011076
10546 012210'                 .WORD   $DVD,$ADD,UPL$14          ;(X/E+E)
10550 000704'
10552 010554'
10554 162716 UPL$14: SUB     #200,@SP            ;(X/E+E)/2
      000200
10560 005366                  DEC     8.(SP)  ;COUNT LOOP
      000010
10564 001420                  BEQ     OUT$14
10566 016546                  MOV     6(R5),-(SP)
      000006
10572 016546                  MOV     4(R5),-(SP)
      000004
10576 016546                  MOV     2(R5),-(SP)      ;USE LOW ORDER PARTS
      000002
10602 011546                  MOV     @R5,-(SP)        ;TOO FROM NOW ON
10604 016646                  MOV     14.(SP),-(SP)
      000016
10610 016646                  MOV     14.(SP),-(SP)
      000016
10614 016646                  MOV     14.(SP),-(SP)
      000016
10620 016646                  MOV     14.(SP),-(SP)
      000016
10624 000746                  BR      LUP$14  ;GO FOR ANOTHER ITERATION
10626 012600 OUT$14: MOV     (SP)+,R0         ;GET RESULT INTO R0-R3
10630 012601                  MOV     (SP)+,R1
10632 012602                  MOV     (SP)+,R2
10634 012603                  MOV     (SP)+,R3
10636 005726                  TST     (SP)+   ;POP ITERATION COUNTER
10640 012605 RTN$14: MOV     (SP)+,R5
10642 000205                  RTS     R5      ;RETURN TO CALLER
10644 004567 ERR$14: JSR     R5,$ERR ;ERROR 4,4
      011136
10650 000773                  BR      RTN$14
10652    004                  .BYTE   4
10653    004                  .BYTE   4
10654 005000 ZER$14: CLR     R0
10656 005001                  CLR     R1
10660 005002                  CLR     R2
10662 005003                  CLR     R3
10664 000765                  BR      RTN$14
                     ;
                              .ENDC
                     ;
                              .IFDF   FPU
                     DSQRT:   MOV     2(R5),R4;        GET ARGUMENT ADDRESS
                              MOV     @R4,R1;          GET HIGH ORDER ARGUMENT
                              BMI     ERR$14  ;ERROR IF ARGUMENT NEGATIVE
                              BEQ     ZER$14  ;FAST EXIT IF ZERO
                              MOV     2(R4),R2;
                              ASR     R1      ;FORM INITIAL ESTIMATE
                              ROR     R2
                              ADD     #20100,R1
                              CLR     -(SP)
```

```
                CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
                MOV     R2,-(SP)
                MOV     R1,-(SP)        ;'CAUSE ADD AND DIVIDE ARE
                MOV     #4,R0;          ITERATION COUNT
                SETD    ;               DOUBLE PRECISION FP
                LDD     (SP)+,F0;       GET INITIAL ESTIMATE
                LDD     @R4,F2;         GET X
        ;
        LUP$14: LDD     F0,F1;          E=E'
                LDD     F2,F0;          X
                DIVD    F1,F0;          X/E
                ADDD    F1,F0;          X/E+E
                DEC     R0;             COUNT
                DIVD    #2.0,F0;        E'=(X/E+E)/2
                BGT     LUP$14;         LOOP
        ;
                STD     F0,-(SP);       MOVE RESULT TO STACK
                MOV     (SP)+,R0;
                MOV     (SP)+,R1;
                MOV     (SP)+,R2;       AND THENCE TO R0...R3
                MOV     (SP)+,R3;
                RTS     R5;
        ;
        ERR$14: JSR     R5,$ERR;        ERROR 4,4
                RTS     R5;
                .BYTE   4
                .BYTE   4
        ZER$14: CLR     R0;
                CLR     R1;
                CLR     R2;
                CLR     R3;
                RTS     R5;
                .ENDC
                .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
```

```
11076 062706 INF$15: ADD      #36.,SP ;FLUSH STACK
      000044
11102 012700         MOV      #040311,R0      ;ANS = SIGN(ARG1)*PI/2
      040311
11106 012701         MOV      #007732,R1
      007732
11112 012702         MOV      #121041,R2
      121041
11116 012703         MOV      #064301,R3
      064301
11122 005775         TST      @2(R5)  ;TEST ARG1
      000002
11126 002002         BGE      INR$15  ;JUMP IF +PI/2
11130 062700         ADD      #100000,R0      ;-PI/2
      100000
1134 000205 INR$15:  RTS      R5      ;RETURN TO USER
                   ;
1136 010546 DATAN:   MOV      R5,-(SP)
1140 005046          CLR      -(SP)   ;CLEAR SIGN FLAG
1142 005046          CLR      -(SP)   ;CLEAR ATAN2 BIAS
1144 005046          CLR      -(SP)
1146 005046          CLR      -(SP)
1150 005046          CLR      -(SP)
1152 005046          CLR      -(SP)   ;CLEAR QUADRANT BIAS
1154 005046          CLR      -(SP)
1156 005046          CLR      -(SP)
1160 005046          CLR      -(SP)
1162 016504          MOV      2(R5),R4        ;GET ARG ADDRESS
     000002
1166 016446          MOV      6(R4),-(SP)
     000006
1172 016446          MOV      4(R4),-(SP)
     000004
1176 016446          MOV      2(R4),-(SP)     ;GET LOW ORDER ARG
     000002
1202 011446          MOV      @R4,-(SP)       ;GET HIGH ORDER
1204 002004 AT1$15:  BGE      PLU$15  ;JUMP IF QUADRANT 1 OR 3
1206 062716          ADD      #100000,@SP     ;GET ABS VALUE
     100000
1212 005266          INC      24.(SP) ;FLAG -
     000030
1216 021627 PLU$15:  CMP      @SP,#40200      ;CHECK IF <1.
     040200
1222 103455          BLO      LE1$15  ;JUMP IF <1.
1224 003011          BGT      GT1$15  ;>1.
1226 005766          TST      2(SP)   ;CHECK LOW ORDER
     000002
1232 001006          BNE      GT1$15
1234 005766          TST      4(SP)
     000004
1240 001003          BNE      GT1$15
1242 005766          TST      6(SP)
     000006
1246 001443          BEQ      LE1$15  ;=1.
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,$POLSH        ;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 103407             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 103401             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'XPD$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 DP 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
```

```
        A1PS15: LDD     @R4,F1;         GET ARG2
                CFCC
                BLT     A2MS15;
                CLRD    F3;             IF ARG2>0, BIAS=0
        A2MS15: DIVD    F1,F0;          ARG1/ARG2, SET FLOAT CC
                BR      AT1S15;         JOIN MAIN ROUTINE
        ;
        INFS15: LDD     PI2S15,F1;              RESULT=SIGN(ARG1)*PI/2
                TST     @R3;            TEST ARG1
                BGE     EXIS15;         +PI/2
                NEGD    F1;             -PI/2
                BR      EXIS15;
        ;
        DATAN:  SETD    ;               SET DP MODE FOR FPU
                CLRD    F3;             CLEAR ATAN2 BIAS
                LDD     @2(R5),F0;      GET ARGUMENT
        AT1S15: CLR     R4;             CLEAR SIGN FLAG
                CFCC    ;               GET SIGN OF ARGUMENT
                STD     F3,F5;          F5=ATAN2 BIAS
                CLRD    F3;             CLEAR QUADRANT BIAS
                BGE     PLUS15;         JUMP IF QUADRANT 1 OR 3
                ABSD    F0;             ABS(X)
                INC     R4;             FLAG =
        PLUS15: LDD     #1.0,F1;        1.0
                CMPD    F0,F1;          CHECK IF X<=1.0
                CFCC
                BLE     LE1S15;
        GT1S15: DEC     R4;             X>1.0, ADJUST SIGN FLAG
                DIVD    F0,F1;          1.0/X
                LDD     F1,F0;          ATAN(X)=PI/2-ATAN(1/X)
                LDD     PI2S15,F3;              QUADRANT BIAS=PI/2
        ;
        LE1S15: STD     F3,F4;          F4=QUADRANT BIAS
                CLRD    F3;             F3=0.0
                CMPD    T15S15,F0;      COMPARE TAN(15) : X
                CFCC
                BGE     L15S15;         X<= TAN(15)
                LDD     PI6S15,F3;              F3=PI/6
                LDD     F0,F1;
                MULD    RT3S15,F0;
                SUBD    #1.0,F0;        X*ROOT3-1.0
                ADDD    RT3S15,F1;      X+ROOT3
                DIVD    F1,F0;          (X*ROOT3-1.0)/(X+ROOT3)
        ;
        L15S15: LDD     F0,F2;          X
                MULD    F0,F0;          X**2
                MOV     #FCOS15,R0;     POINTER TO POLYNOMIAL CONSTANTS
                MOV     #8.,R1;         COUNT OF COEFFICIENTS
                LDD     (R0)+,F1;       INITIALIZE ACCUMULATOR
        XPDS15: MULD    F0,F1;
                DEC     R1;             COUNT
                ADDD    (R0)+,F1;       F1:= F1* X**2 + C(I)
                BGT     XPDS15; LOOP
                MULD    F2,F1;          F1:= F1*X
                ADDD    F3,F1;          PI/6 OR 0.0
                SUBD    F4,F1;          P(X)-QUAD BIAS
                TST     R4;             TEST SIGN FLAG
```

```
327                     BEQ     SG1$15;         NO ADJUSTMENT
328                     NEGD    F1;             NEGATE RESULT FOR (-1,0)&(1,INF)
329             SG1$15: ADDD    F5,F1;          ATAN2 BIAS
330             ;
331             EXI$15: STD     F1,-(SP);       MOVE RESULT TO STACK
332                     MOV     (SP)+,R0;       AND THEN TO REGISTERS
333                     MOV     (SP)+,R1;
334                     MOV     (SP)+,R2;
335                     MOV     (SP)+,R3;
336                     RTS     R5;             EXIT
337             ;
338             ;
339             PI$15:  .WORD   040511,007732;  PI
340                     .WORD   121041,064301;
341             ;
342             PI2$15: .WORD   040311,007732;  PI/2
343                     .WORD   121041,064301;
344             ;
345             T15$15: .WORD   037611,030242;  TAN(15)
346                     .WORD   172306,065261;
347             ;
348             PI6$15: .WORD   040006,005221;  PI/6
349                     .WORD   140553,115454;
350             ;
351             RT3$15: .WORD   040335,131727;
352                     .WORD   041302,062524;
353                     .ENDC
354 2100 037065 FCU$15: .WORD   037065,150707   ;.0443895157187
    2102 150707
355 2104 162300        .WORD   162300,163030
    2106 163030
356             ;
357 2110 137204        .WORD   137204,143233   ;-.0648319351O303
    2112 143233
358 2114 004010        .WORD   004010,000413
    2116 000413
359             ;
360 2120 037235        .WORD   037235,043002   ;.0767936896066
    2122 043002
361 2124 027154        .WORD   027154,142446
    2126 142446
362             ;
363 2130 137272        .WORD   137272,025671   ;-.0909037114191074
    2132 025671
364 2134 116412        .WORD   116412,065630
    2136 065630
365             ;
366 2140 037343        .WORD   037343,107047   ;.1111109789805I048
    2142 107047
367 2144 023625        .WORD   023625,025401
    2146 025401
368             ;
369 2150 137422        .WORD   137422,044444   ;-.142857141028255 45
    2152 044444
370 2154 071335        .WORD   071335,116151
    2156 116151
371             ;
```

```
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 005566          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 116666          MOVB    D+3(SP),D(SP)
          000013
          000010
72 12362 116666          MOVB    D+2(SP),D+3(SP)
          000012
          000013
73 12370 116666          MOVB    D+5(SP),D+2(SP)
          000015
          000012
74 12376 116666          MOVB    D+4(SP),D+5(SP)
          000014
          000015
75 12404 116666          MOVB    D+7(SP),D+4(SP)
          000017
          000014
76 12412 116666          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 110460            MOVB    R4,Q(SP)         ;SAVE ALL HIGH ORDER Q FRACTION
         000024
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 010460            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
```

```
         000012
164 3012 101034          BHI     NGO$16  ;JUMP IF DIVISOR BIGGER
165 3014 103414          BLO     GO$16   ;JUMP IF DIVISOR SMALLER
166 3016 026001          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 026002          CMP     D+4+2(SP),R2
         000016
170 3032 101024          BHI     NGO$16
171 3034 103404          BLO     GO$16
172 3036 026003          CMP     D+6+2(SP),R3
         000020
173 3042 101020          BHI     NGO$16
174 3044 001422          BEQ     NQO$16  ;JUMP IF NUMERATOR =DENOMINATOR
175 3046 166003 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 166002          SUB     D+4+2(SP),R2
         000016
180 3064 005601          SBC     R1
181 3066 005600          SBC     R0
182 3070 166001          SUB     D+2+2(SP),R1
         000014
183 3074 005600          SBC     R0
184 3076 166000          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 NQO$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   CNU$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
```

```
57 13242 ØØ5Ø16 ZER$17: CLR      @SP      ;RESULT IS Ø
58 13244 ØØØ134         JMP      @(R4)+
59                      .ENDC
60               ;      $DVI FOR THE EAE
61                      .IFDF    EAE
62               $DVI:  MOV      #MQ,RØ   ;POINT TO MQ
63                      MOV      (SP)+,R1          ;GET DIVISOR
64                      BEQ      CHK$17  ;JUMP IF DIVISION BY Ø
65                      MOV      (SP)+,@RØ         ;DIVIDEND TO MQ
66                      TST      -(RØ)    ;SKIP AC
67                      MOV      R1,-(RØ)          ;DIVISOR TO DIV
68                      CMP      (RØ)+,(RØ)+       ;POINT TO MQ
69                      MOV      @RØ,-(SP)         ;GET QUOTIENT
70                      JMP      @(R4)+   ;RETURN TO USER
71                      .ENDC
72               ;      $DVI FOR MUL/DIV
73                      .IFDF    MULDIV
74               $DVI:  MOV      2(SP),R1          ;GET LOW ORDER DIVIDEND
75                      .WORD    ØØ67ØØ   ;;SEX    RØ       ;EXTEND SIGN
76                      .WORD    Ø71Ø26   ;;DIV    (SP)+,RØ          ;DIVIDE
77                      MOV      RØ,@SP   ;PUSH QUOTIENT
78                      BCS      CHK$17   ;JUMP IF ERROR
79                      JMP      @(R4)+
80                      .ENDC
81 13246 ØØ4567 CHK$17: JSR      R5,$ERR ;ERROR 3,5
         ØØ6534
82 13252 ØØØ134         JMP      @(R4)+
83 13254    ØØ3         .BYTE    3
84 13255    ØØ5         .BYTE    5
85                      .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                 G=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+8-2(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 156016                 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     DHIS18    ;JUMP IF DENOMINATOR HIGH
84                          .IFNDF  EAE&MULDIV
85 13414 101034             BHI     DLWS18    ;JUMP IF DENOMINATOR LOW
86 13416 020103             CMP     R1,R3     ;COMPARE LOW ORDER PARTS
87 13420 101032             BHI     DLWS18
88 13422 001034             BNE     DHIS18
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      FLTS18
93                          .ENDC
94                          .IFDF   EAE!MULDIV
95                          BHIS    DLWS18    ;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 DCH$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
```

```
                    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
            NBTS18: .WORD   073427,1        ;;      ASHC    #1,R4   ;SHIFT
                    BMI     NBIS18  ;CHECK FOR NORMAL BIT
                    DEC     @SP     ;COMPENSATE EXPONENT
                    BR      NBTS18  ;GO AGAIN
            NBIS18: .WORD   073427,-7       ;;ASHC  #-8,R4  ;ALIGN FRACTION
                    MOV     R4,Q(SP)        ;STORE HIGH ORDER
                    .ENDC
3550 012604 FLTS18: MOV     (SP)+,R4        ;PUSH UP EXPONENT
3552 062704         ADD     #200,R4 ;ADD IN EXCESS 200
     000200
3556 003741         BLE     UNDS18  ;UNDERFLOW
3560 022704         CMP     #377,R4
     000377
3564 002733         BLT     OVRS18  ;OVERFLOW
3566 110466         MOVB    R4,Q+1-2(SP)    ;INSERT EXPONENT IN RESULT
     000013
3572 006026 SGNS18: 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     OV1S18
3616 102715         BVS     OV1S18
3620 012605 RTNS18: 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 DV1S18: ASL     R5      ;SHIFT QUOTIENT
3632 006301         ASL     R1      ;SHIFT NUMERATOR
3634 006100         ROL     R0
```

```
199 3636 103406          BCS     GOS18    ;GUARANTEED TO GO
200 3640 020200          CMP     R2,R0    ;COMPARE HIGH DIVISOR AND DIVIDEND
201 3642 101010          BHI     NGOS18   ;JUMP IF DIVISOR BIGGER
202 3644 103403          BLO     GOS18    ;JUMP IF DIVISOR SMALLER
203 3646 020301          CMP     R3,R1    ;CHECK THE LOW ORDERS
204 3650 101005          BHI     NGOS18
205 3652 001407          BEQ     NQDS18   ;JUMP IF NUMERATOR =DENOMINATOR
206 3654 160301 GOS18:   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 NGOS18:  DEC     R4       ;COUNT LOOP
211 3666 003360          BGT     DV1S18
212 3670 000207          RTS     PC
213 3672 005205 NQDS18:  INC     R5       ;INSERT LAST 1 BIT IN QUOTIENT
214 3674 000401          BR      EQ1S18
215 3676 006305 EQ2S18:  ASL     R5       ;FINISH OUT QUOTIENT WITH 0'S
216 3700 005304 EQ1S18:  DEC     R4
217 3702 003375          BGT     EQ2S18
218 3704 005204          INC     R4       ;FLAG NO MORE NUMERATOR
219 3706 000207 RTSS18:  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
       013710 010546    DEXP:   MOV     R5,-(SP)        ;SAVE RETURN
       013712 016504            MOV     2(R5),R4        ;GET ARG POINTER
              000002
       013716 011400            MOV     @R4,R0  ;GET HIGH ORDER ARG
                                .ENDC
       013720 003004            BGT     POS$19  ;JUMP IF +
       013722 020027            CMP     R0,#141602      ;ARG IS -
              141602
       013726 101062            BHI     ZER$19  ;JUMP IF ARG <88.7
       013730 000403            BR      SMT$19  ;JUMP TO TEST SMALL MAGNITUDE ARG
       013732 020027    POS$19: CMP     R0,#41660
              041660
       013736 101053            BHI     OVR$19  ;JUMP IF ARG >87
       013740 006300    SMT$19: ASL     R0      ;DUMP SIGN
       013742 020027            CMP     R0,#43000
              043000
       013746 103444            BLO     ONE$19  ;JUMP IF ARG MAGNITUDE <2**-60
                                .IFNDF  FPU
       013750 162706            SUB     #20.,SP ;GET WORK SPACE
              000024
       013754 062704            ADD     #8.,R4  ;POINT TO LOW ORDER ARG
              000010
       013760 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=7.213503410844819083
         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)
```

```
94 14142 010146             MOV     R1,-(SP)
95 14144 010046             MOV     R0,-(SP)
96                    ;
97 14146 012746             MOV     #153703,-(SP)   ;PUSH P1=.05776113583180192B
          153703
98 14152 012746             MOV     #153011,-(SP)
          153011
99 14156 012746             MOV     #113360,-(SP)
          113360
100 4162 012746             MOV     #037154,-(SP)
          037154
101                   ;
102 4166 010346             MOV     R3,-(SP)
103 4170 010246             MOV     R2,-(SP)
104 4172 010146             MOV     R1,-(SP)
105 4174 010046             MOV     R0,-(SP)
106                   ;
107 4176 012746             MOV     #171042,-(SP)   ;PUSH Q0=20.813771196523036B2973
          171042
108 4202 012746             MOV     #074433,-(SP)
          074433
109 4206 012746             MOV     #101232,-(SP)
          101232
110 4212 012746             MOV     #041246,-(SP)
          041246
111                   ;
112 4216 004467             JSR     R4,$POLSH
          005422
113 4222 000704'            .WORD   $ADD/AUP$19     ;A=E*E+Q0 TO WORK SPACE
    4224 014410'
114 4226 016146'            .WORD   $MLD/$ADD,$MLD  ;B=E*(E*E+P1+P0)
    4230 000704'
    4232 016146'
115 4234 014476'            .WORD   TWC$19  ;DUPLICATE A AND B
116 4236 000704'            .WORD   $ADD/ABP$19     ;A+B TO WORD SPACE
    4240 014432'
117 4242 000700'            .WORD   $SBD/$DVD       ;(A+B)/(A-B)
    4244 012210'
118 4246 014250'            .WORD   SCL$19  ;APPLY SCALE FACTORS
119 4250 012705 SCL$19: MOV     #RT2$19+8.,R5   ;POINT TO POWERS OF 2
          014554'
120 4254 006266 ASR$19: ASR     8.(SP)  ;SHIFT D
          000010
121 4260 103010             BCC     NML$19  ;JUMP IF BIT IS OFF
122 4262 014546             MOV     -(R5),-(SP)     ;PUSH 2**((2**N)*D/16)
123 4264 014546             MOV     -(R5),-(SP)
124 4266 014546             MOV     -(R5),-(SP)
125 4270 014546             MOV     -(R5),-(SP)
126 4272 004467             JSR     R4,$POLSH
          005346
127 4276 016146'            .WORD   $MLD/ASR$19     ;MULTIPLY BY ABOVE FACTOR AND TE
    4300 014254'
128 4302 001403 NML$19: BEQ     SC1$19
129 4304 162705             SUB     #8.,R5  ;POINT TO NEXT POWER OF 2
          000010
130 4310 000761             BR      ASR$19
131 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 TWC$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,R3;           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
```

```
                                .TITLE  $EXP04
                                .IFDF   CND$20
                        ;
                        ;       EXP     V004A
                        ;
                        ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                        ;

                                .GLOBL  EXP,$ERRA;
                                .IFNDF  FPU
                                .GLOBL  $ADR,$SBR,$MLR,$DVR,$IR,$RI,$POLSH;
                                .ENDC
                        ;       EXP     THE REAL EXPONENTIATION ROUTINE
                        ;       CALLING SEQUENCE:
                        ;       JSR     R5,EXP
                        ;       BR      A
                        ;       .WORD   ARG ADDRESS
                        ;A:
                        ;       RETURNS EXPONENTIAL IN R0 AND R1.
                        ;
        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
        000002                  F2=%2
        000003                  F3=%3
 14554  016504  EXP:    MOV     2(R5),R4        ;GET ARGUMENT POINTER
        000002
 14560  011400          MOV     @R4,R0  ;GET HIGH ORDER ARG
 14562  003004          BGT     POS$20  ;JUMP IF ARG +
 14564  020027          CMP     R0,#141662
        141662
 14570  101146          BHI     ZER$20  ;JUMP IF EXPONENT < -88.7
 14572  000403          BR      SMT$20
 14574  020027  POS$20: CMP     R0,#41660
        041660
 14600  101137          BHI     OVR$20  ;JUMP IF EXPONENT > 87
 14602  006300  SMT$20: ASL     R0      ;DUMP SIGN
 14604  020027          CMP     R0,#63000
        063000
 14610  103527          BLO     ONE$20  ;JUMP IF EXPONENT MAGNITUDE < 2**-28
                        .IFNDF  FPU
 14612  005746          TST     -(SP)   ;SAVE SPACE FOR SCALE
 14614  005046          CLR     -(SP)   ;PUSH A 1.
 14616  012746          MOV     #40200,-(SP)
        040200
 14622  016446          MOV     2(R4),-(SP)     ;GET LOW ORDER ARGUMENT
        000002
 14626  011446          MOV     @R4,-(SP)       ;HIGH ORDER
 14630  016446          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   PLES20  ;PUSH LOG2(E)
54 14644 017162'          .WORD   $MLR
55 14646 017650'          .WORD   $RI     ;FIX LOG2(E)*X
56 14650 014744'          .WORD   ESVS20  ;SAVE EXPONENT SCALE
57 14652 016062'          .WORD   $IR     ;FLOAT IT
58 14654 014732'          .WORD   PLES20  ;PUSH LOG2(E)
59 14656 013256'          .WORD   $DVR
60 14660 002004'          .WORD   $SBR
61 14662 014752'          .WORD   CFRS20  ;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   INCS20  ;-2*Y/(A0+Y+A1/(B1+Y*Y))
69 14702 002010'          .WORD   $ADR    ;1-2*Y/.........
70 14704 014720'          .WORD   DUPS20  ;DUPLICATE IT
71 14706 017162'          .WORD   $MLR    ;(1-2*Y/......)**2
72 14710 015046'          .WORD   SCLS20  ;EXIT POLISH MODE AND SCALE RESULT
73 14712 062716 INCS20:   ADD     #100200,@SP      ;MULTIPLY BY -2.0
         100200
74 14716 000134           JMP     @(R4)+  ;GO BACK TO LIST
75                ;
76 14720 016646 DUPS20:   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 PLES20:   MOV     #125073,-(SP)    ;PUSH LOG2(E)
         125073
81 14736 012746           MOV     #40270,-(SP)
         040270
82 14742 000134           JMP     @(R4)+
83                ;
84 14744 011666 ESVS20:   MOV     @SP,10.(SP)      ;SAVE EXPONENT SCALE
         000012
85 14750 000134           JMP     @(R4)+
86                ;
87 14752 006116 CFRS20:   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    ZFRS20  ;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
```

```
15006 010146            MOV     R1,-(SP)
15010 010046            MOV     R0,-(SP)
15012 012746            MOV     #071571,-(SP)   ;PUSH 601.8042667 **************
      071571
 5016 012746            MOV     #042426,-(SP)
      042426
 5022 012746            MOV     #056133,-(SP)   ;PUSH 60.0901907 ***********
      056133
 5026 012746            MOV     #041560,-(SP)
      041560
 5032 010146            MOV     R1,-(SP)
 5034 010046            MOV     R0,-(SP)
 5036 010146            MOV     R1,-(SP)
 5040 010046            MOV     R0,-(SP)
 5042 000134            JMP     @(R4)+
                        .ENDC
                ;
                        .IFDF   FPU
                        SETD    ;               DOUBLE PRECISION ARGUMENT REDUCT
                        SETI    ;               SHORT INTEGERS
                        MOV     #FCO$20,R0;     POINTER TO CONSTANTS
                        LDCFD   @R4,F2;         GET ARGUMENT
                        MODD    (R0)+,F2;       F2=FRACT(X*LOG2(E))
                        STCDI   F3,R4;          R4=INT  (X*LOG2(E))
                        LDD     #1.0,F0;        F0=1.0
                        DIVD    (R0)+,F2;       Y=F2/(2*LOG2(E))
                        SETF    ;
                        LDCDF   F2,F2;          REST IN SINGLE PRECISION
                        CFCC    ;               TEST FOR UNDERFLOW
                        BEQ     SC1$20;         APPROXIMATION RESULT IS 1.0
                        LDF     F2,F3;
                        MULF    F3,F3;          Y*Y
                        ADDF    (R0)+,F3;       B1+Y*Y
                        LDF     (R0)+,F1;
                        DIVF    F3,F1;          A1/(B1+Y*Y)
                        ADDF    F2,F1;
                        ADDF    (R0)+,F1;       A0+Y+A1/(B1+Y*Y)
                        DIVF    F1,F2;          Y/(A0+Y+A1/(B1+Y*Y))
                        MULF    #2.0,F2;
                        SUBF    F2,F0;          1-2*Y/. . .
                        MULF    F0,F0;          (1-2*Y/. . . )**2
                SC1$20: STF     F0,-(SP);       MOVE APPROXIMATION TO STACK
                        .ENDC
                ;
                        .IFNDF  FPU
 5044 022626    ZFR$20: CMP     (SP)+,(SP)+     ;FLUSH CFRACT ARG
                ;                         RESULT IS 1.
                        .ENDC
 5046 012600    SCL$20: MOV     (SP)+,R0;       GET APPROXIMATION RESULT
 5050 012601            MOV     (SP)+,R1;
                        .IFNDF  FPU
 5052 012604            MOV     (SP)+,R4;       GET INT(X*LOG2(E))
                        .ENDC
 5054 000304            SWAB    R4;             MAKE INTO EXPONENT MODIFIER
 5056 105004            CLRB    R4;
 5060 006204            ASR     R4;
 5062 060400            ADD     R4,R0;          ADD IN EXPONENT MODIFIER
```

```
5064 100405             BMI     OVR$20;            TEST OVERFLOW
5066 000205             RTS     R5;
                ;
5070 005001 ONE$20: CLR     R1
5072 012700             MOV     #40200,R0          ;EXP(TINY) = 1.
     040200
5076 000205             RTS     R5
5100 012700 OVR$20: MOV     #2404,R0           ;ERROR 4,5
     002404
5104 000402             BR      ECL$20
5106 012700 ZER$20: MOV     #2405,R0           ;ERROR 5,5
     002405
5112 004567 ECL$20: JSR     R5,$ERRA
     004700
5116 005000             CLR     R0      ;RETURN 0
5120 005001             CLR     R1
5122 000205             RTS     R5
                ;
                        .IFDF   FPU
                ;       ORDER-DEPENDENT CONSTANTS
                ;
                FCU$20: .WORD   040270,125073;  LOG2(E) DOUBLE PRECISION
                        .WORD   024534,013761;
                ;
                        .WORD   040470,125073;  2*LOG2(E) DOUBLE PRECISION
                        .WORD   024534,013761;
                ;
                        .WORD   041560,056133;  B1=60.0901907
                ;
                        .WORD   042426,071571;  A1=601.8042667
                ;
                        .WORD   141100,036602;  A0=-12.01501675
                        .ENDC
                        .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
```

```
                             .TITLE  $FIXØ3
                             .IFDF   CNDS22
                      ;
                      ;      IFIX    VØØ3A
                      ;
                      ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                      ;
                             .GLOBL  IFIX,$RI,$POLSH
                      ;      THE FORTRAN IFIX FUNCTION
                      ;      CALLING SEQUENCE:
                      ;      JSR     R5,IFIX
                      ;      BR      A
                      ;      .WORD   ARGUMENT ADDRESS
                      ;A:
                      ;      RETURNS THE TRUNCATED AND FIXED REAL
                      ;      ARGUMENT AS AN INTEGER IN RØ.
                      ;
        ØØØØØØ               RØ=%Ø
        ØØØØØ4               R4=%4
        ØØØØØ5               R5=%5
        ØØØØØ6               SP=%6
  15154 Ø165Ø4 IFIX:  MOV     2(R5),R4         ;GET ARG ADDRESS
        ØØØØØ2
  1516Ø Ø16446        MOV     2(R4),-(SP)      ;PUSH ARG
        ØØØØØ2
  15164 Ø11446        MOV     @R4,-(SP)
  15166 ØØ4467 RND$22: JSR    R4,$POLSH        ;ENTER POLISH MODE
        ØØ4452
  15172 Ø1765Ø'       .WORD   $RI,UPL$22       ;TRUNCATE AND FIX
  15174 Ø15176'
  15176 Ø126ØØ UPL$22: MOV    (SP)+,RØ         ;POP INTEGER RESULT
  152ØØ ØØØ2Ø5        RTS     R5       ;RETURN TO CALLER
                      .ENDC
```

```
                     .TITLE  $FLTØ2
                     .IFDF   CND$23
             ;
             ;       FLOAT   VØØ2A
             ;
             ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
             ;
                     .GLOBL  FLOAT,$IR,$POLSH,$POPR3
             ;       FLOAT   THE FORTRAN FLOAT FUNCTION
             ;       CALLING SEQUENCE:
             ;       JSR     R5,FLOAT
             ;       BR      A
             ;       .WORD   ADDRESS OF INTEGER
             ;A:
             ;       RETURNS REAL EQUIVALENT IN RØ AND R1.
             ;       USES $IR.
             ;
      ØØØØØØ         RØ=%Ø
      ØØØØØ1         R1=%1
      ØØØØØ4         R4=%4
      ØØØØØ5         R5=%5
      ØØØØØ6         SP=%6
15202 Ø17546 FLOAT:  MOV     @2(R5),-(SP)    ;GET ARGUMENT ON STACK
      ØØØØØ2
15206 ØØ4467         JSR     R4,$POLSH       ;ENTER POLISH MODE
      ØØ4432
15212 Ø16Ø62'        .WORD   $IR     ;CALL $IR TO CONVERT TO REAL
15214 Ø1761Ø'        .WORD   $POPR3  ;POP RESULT TO REGS
15216 Ø1522Ø'        .WORD   UPL$23
1522Ø ØØØ2Ø5 UPL$23: RTS     R5      ;RETURN TO CALLER
                     .ENDC
```

```
                                .TITLE  $ICI02
                                .IFDF   CND$24
                        ;
                        ;       $ICI    V002A
                        ;
                        ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                        ;
                                .GLOBL  $ICI,$OCI
                        ;       $OCI    ASCII TO OCTAL CONVERSION
                        ;       $ICI    ASCII TO INTEGER CONVERSION
                        ;       CALLING SEQUENCE:
                        ;       PUSH    CHARACTER FIELD START
                        ;       PUSH    CHARACTER FIELD LENGTH
                        ;       JSR     PC,$ICI   OR $OCI
                        ;       RETURNS WITH INTEGER RESULT ON TOP OF STACK.
         000000                 R0=%0
         000001                 R1=%1
         000002                 R2=%2
         000006                 SP=%6
         000007                 PC=%7
   15222 012746 $OCI:   MOV     #67,-(SP)       ;SET OCTAL FLAGS
         000067
   15226 000402         BR      GO$24
   15230 012746 $ICI:   MOV     #471,-(SP)      ;SET DECIMAL FLAGS
         000471
   15234 010146 GO$24:  MOV     R1,-(SP)        ;SAVE R1
   15236 016601         MOV     8.(SP),R1       ;GET STRING START
         000010
   15242 066666         ADD     6(SP),8.(SP)    ;GET END+1
         000006
         000010
   15250 016666         MOV     4(SP),6(SP)     ;FIDDLE RETURN POINTER
         000004
         000006
   15256 010066         MOV     R0,4(SP)        ;SAVE R0
         000004
   15262 010246         MOV     R2,-(SP)        ;SAVE R2
   15264 005046         CLR     -(SP)   ;CLEAR SIGN
   15266 005000         CLR     R0      ;CLEAR WORK SPACE
   15270 112102 STT$24: MOVB    (R1)+,R2        ;GET NEXT CHAR.
   15272 042702         BIC     #177600,R2
         177600
   15276 120227         CMPB    R2,#' ;
         000040
   15302 001004         BNE     SGS$24  ;JUMP IF NOT BLANK
   15304 020166         CMP     R1,12.(SP)
         000014
   15310 002767         BLT     STT$24  ;JUMP IF MORE TO SCAN
   15312 000454         BR      SGN$24  ;DONE
   15314 105766 SGS$24: TSTB    7(SP);          IF OCTAL CONVERSION
         000007
   15320 001002         BNE     SN1$24;         DO NOT PERMIT SIGNS
   15322 005216         INC     @SP;            OCTAL - FAKE THE SIGN BIT
   15324 000420         BR      NCK$24;         GO PROCESS THE DIGIT
   15326 120227 SN1$24: CMPB    R2,#'+;
         000053
   15332 001441         BEQ     FLD$24  ;JUMP IF +
15334 120227            CMPB    R2,#'-
      000055
15340 001012            BNE     NCK$24  ;JUMP IF NOT -
15342 005216            INC     @SP     ;SET SIGN -
15344 000434            BR      FLD$24
15346 112102 NXT$24:    MOVB    (R1)+,R2        ;GET NEXT CHAR.
15350 042702            BIC     #177600,R2
      177600
15354 120227            CMPB    R2,#' ;
      000040
15360 001002            BNE     NCK$24  ;JUMP IF NOT BLANK
15362 112702            MOVB    #60,R2  ;BLANK =ZERO
      000060
15366 120227 NCK$24:    CMPB    R2,#'0
      000060
15372 002440            BLT     ERR$24  ;JUMP IF TOO SMALL
15374 120266            CMPB    R2,6(SP)
      000006
15400 003035            BGT     ERR$24  ;JUMP IF TOO BIG
15402 162702            SUB     #60,R2  ;MAKE NUMERIC
      000060
15406 105766            TSTB    7(SP)   ;OCTAL OR BINARY
      000007
15412 001435            BEQ     OCL$24
15414 006300            ASL     R0      ;R0=BASE*R0+R2
15416 102426            BVS     ERR$24
15420 160002            SUB     R0,R2
15422 006300            ASL     R0
15424 102423            BVS     ERR$24
15426 006300            ASL     R0
15430 102421            BVS     ERR$24
15432 160200            SUB     R2,R0
15434 102417            BVS     ERR$24
15436 020166 FLD$24:    CMP     R1,12,(SP)
      000014
15442 002741            BLT     NXT$24  ;JUMP IF MORE FIELD TO SCAN
15444 006026 SGN$24:    ROR     (SP)+   ;TEST SIGN
15446 103403            BCS     DNE$24  ;JUMP IF -
15450 005400            NEG     R0      ;MAKE +
15452 102411            BVS     NGM$24  ;JUMP IF -NEGMAX
15454 000241            CLC     ;SET SUCCESS FLAG
15456 012602 DNE$24:    MOV     (SP)+,R2        ;RESTORE R2
15460 012601            MOV     (SP)+,R1        ;RESTORE R1
15462 006126            ROL     (SP)+   ;FLUSH FLAG AND SET C BIT IF ERROR
15464 010066            MOV     R0,4(SP)        ;RETURN RESULT
      000004
15470 012600            MOV     (SP)+,R0
15472 000207            RTS     PC
15474 005726 ERR$24:    TST     (SP)+   ;FLUSH SIGN
15476 005000 NGM$24:    CLR     R0
15500 005166            COM     4(SP)   ;SET ERROR FLAG
      000004
15504 000764            BR      DNE$24
             ;
15506 006100 OCL$24:    ROL     R0;     SHIFT 3 BITS LEFT,
15510 103771            BCS     ERR$24; CHECKING AS YOU GO
15512 006100            ROL     R0;
```

```
   15514 103767          BCS     ERRS24;
   15516 006100          ROL     R0;
   15520 103765          BCS     ERRS24;
   15522 060200          ADD     R2,R0;  ADD IN THE DIGIT
   15524 000744          BR      FLDS24; DO NEXT
                         .ENDC
```

```
                              .TITLE  $IC002
                              .IFDF   CND$25
                      ;
                      ;       $ICO    V002A
                      ;
                      ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                      ;

                              .GLOBL  $ICO,$OCO
                      ;       $OCO    OCTAL TO ASCII CONVERSION
                      ;       $ICO    INTEGER TO ASCII CONVERSION
                      ;       CALLING SEQUENCE:
                      ;       PUSH    FIELD START LOCATION
                      ;       PUSH    FIELD LENGTH
                      ;       PUSH VALUE
                      ;       JSR     PC,$ICO (OR $OCO)
                      ;       ERROR WILL RETURN WITH C BIT SET ON
                      ;       R0, R1, R2, R3 ARE DESTROYED
        000000                R0=%0
        000001                R1=%1
        000002                R2=%2
        000003                R3=%3
        000004                R4=%4
        000006                SP=%6
        000007                PC=%7
 15526  012700 $OCO:          MOV     #OCT$25-REL$25,R0       ;POINT TO OCTAL TABLE
        000166
 15532  000402                BR      GO$25
 15534  012700 $ICO:          MOV     #DEC$25-REL$25,R0       ;POINT TO DECIMAL TABLE
        000154
 15540  010446 GO$25:         MOV     R4,-(SP)
 15542  016603                MOV     8.(SP),R3       ;GET FIELD START
        000010
 15546  016602                MOV     6.(SP),R2       ;GET FIELD LENGTH
        000006
 15552  002003                BGE     LPS$25  ;JUMP IF LENGTH NOT NEG
 15554  005002                CLR     R2
 15556  005066                CLR     6(SP)
        000006
 15562  016604 LPS$25:        MOV     4.(SP),R4       ;GET VALUE TO BE CONVERTED
        000004
 15566  012746                MOV     #' ,-(SP)       ;CLEAR SIGN
        000040
 15572  020027                CMP     R0,#OCT$25-REL$25       ;CHECK IF DOING OCTAL
        000166
 15576  001405                BEQ     POS$25  ;YES, GIVE MAGNITUDE RESULT
 15600  005704                TST     R4
 15602  002003                BGE     POS$25  ;JUMP IF +
 15604  005404                NEG     R4      ;GET ABSOLUTE VALUE
 15606  012716                MOV     #'-,@SP ;SAVE -
        000055
 15612  005046 POS$25:        CLR     -(SP)   ;SET FENCE
 15614  060700                ADD     PC,R0
 15616         REL$25:
 15616  005710 TST$25:        TST     @R0
 15620  001416                BEQ     MOV$25  ;JUMP IF ALL POWERS DONE
 15622  005001                CLR     R1
```

```
15624 161004 SUB$25: SUB     @R0,R4  ;SEE IF CURRENT POWER WILL GO AGAIN
15626 103402         BLO     BAC$25
15630 005201         INC     R1      ;BUMP DIGIT
15632 000774         BR      SUB$25
15634 062004 BAC$25: ADD     (R0)+,R4        ;TOO MUCH, BACK UP
15636 005701         TST     R1
15640 001002         BNE     NZE$25  ;JUMP IF DIGIT NOT 0
15642 005716         TST     @SP
15644 001764         BEQ     TST$25  ;JUMP IF NO NON-ZERO DIGITS YET
15646 062701 NZE$25: ADD     #60,R1  ;CONVERT TO ASCII
      000060
15652 010146         MOV     R1,-(SP)
15654 000760         BR      TST$25
15656 060203 MOV$25: ADD     R2,R3   ;POINT TO FIELD END
15660 062704         ADD     #60,R4  ;CONVERT LEAST SIGNIFICANT DIGIT
      000060
15664 110443         MOVB    R4,-(R3)
15666 005302 DCR$25: DEC     R2
15670 003410         BLE     FUL$25  ;JUMP IF COUNT EXHAUSTED
15672 112643         MOVB    (SP)+,-(R3)     ;MOVE DIGIT
15674 001374         BNE     DCR$25  ;JUMP IF NOT THE FENCE
15676 112613         MOVB    (SP)+,@R3       ;MOVE OUT THE SIGN
15700 005302 FIL$25: DEC     R2
15702 001410         BEQ     DNE$25  ;JUMP IF FIELD FILLED
15704 112743         MOVB    #' ,-(R3)       ;MOVE IN LEADING BLANKS
      000040
15710 000773         BR      FIL$25
15712 005726 FUL$25: TST     (SP)+
15714 001011         BNE     ERR$25  ;NUMBER TOO BIG FOR FIELD
15716 022726         CMP     #' ,(SP)+
      000040
15722 001011         BNE     STS$25-4.       ;JUMP IF NO ROOM FOR -
15724 012604 DNE$25: MOV     (SP)+,R4
15726 012666         MOV     (SP)+,4(SP)     ;MOVE RETURN UP
      000004
15732 005726         TST     (SP)+   ;FLUSH VALUE
15734 006126         ROL     (SP)+   ;FLUSH FLAG AND SET C BIT ON IF ERROR
15736 000207         RTS     PC
15740 005726 ERR$25: TST     (SP)+
15742 001376         BNE     ERR$25
15744 005726         TST     (SP)+   ;FLUSH SIGN
15746 016603         MOV     8.(SP),R3
      000010
15752 112723 STS$25: MOVB    #'*,(R3)+       ;FILL FIELD WITH *
      000052
15756 005366         DEC     6(SP)
      000006
15762 003373         BGT     STS$25  ;JUMP IF MORE TO DO
15764 005166         COM     6(SP)   ;FLAG ERROR
      000006
15770 000755         BR      DNE$25
15772 023420 DEC$25: .WORD   10000.,1000.,100.,10.,0
15774 001750
15776 000144
16000 000012
16002 000000
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  SINT02
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 012600 UPL$26: MOV     (SP)+,R0        ;POP INTEGER RESULT
29 16044 000205         RTS     R5
30                      .ENDC
```

```
                                .TITLE  $IR04
                                .IFDF   CND$27
                                .GLOBL  $IR,$ID
                        ;       INTEGER TO REAL CONVERSION.
                        ;
                        ;       $IR     V004A
                        ;
                        ;       COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORP., MAYNARD, MA
                        ;       ARGUMENT IS A FULL WORD ON THE TOP OF THE STACK
                        ;       CONVERT IT TO REAL FORMAT AND RETURN IT AS THE TOP
                        ;       TWO WORDS ON THE STACK.
         000000                 R0=%0
         000001                 R1=%1
         000002                 R2=%2
         000003                 R3=%3
         000004                 R4=%4
         000006                 SP=%6
         177304                 MQ=177304
         177312                 NOR=177312
         000000                 F0=%0
                                .IFDF   FPU
                        $ID:    SETD;
                                BR      IDI$27
                        $IR:    SETF    ;
                        IDI$27: SETI    ;               SHORT INTEGERS
                                LDCIF   (SP)+,F0;       CONVERT
                                STF     F0,-(SP);       RESULT TO STACK
                                JMP     @(R4)+
                                .ENDC
                                .IFNDF  FPU
  16046  011646         $ID:    MOV     @SP,-(SP)       ;PUSH ARGUMENT DOWN
  16050  011646                 MOV     @SP,-(SP)
  16052  005066                 CLR     2(SP)   ;CLEAR LOWEST ORDER DOUBLE
         000002
  16056  005066                 CLR     4(SP)
         000004
  16062  005046         $IR:    CLR     -(SP)   ;MAKE ROOM FOR RESULT
  16064  016601                 MOV     2(SP),R1        ;GET INTEGER ARGUMENT
         000002
  16070  003002                 BGT     POS$27
  16072  001424                 BEQ     ZER$27
  16074  005401                 NEG     R1      ;GET ABSOLUTE VALUE
  16076  006146         POS$27: ROL     -(SP)   ;SAVE SIGN
                                .IFNDF  EAE
  16100  012702                 MOV     #220,R2 ;GET MAX. POSSIBLE EXPONENT +1
         000220
                                .ENDC
                        ;       EAE CODE
                                .IFDF   EAE
                                MOV     #217,R2 ;GET MAX. POSSIBLE EXPONENT
                                .ENDC
  16104  105066                 CLRB    4(SP)   ;CLEAR LOWEST ORDER FRACTION
         000004
  16110                 NOM$27:
                                .IFNDF  EAE
  16110  006101                 ROL     R1      ;LOOK FOR NORMAL BIT
  16112  103402                 BCS     NOD$27  ;JUMP IF FOUND
```

```
16114 005302            DEC     R2      ;DECREASE EXPONENT
16116 000774            BR      NOM$27  ;TRY AGAIN
                        .ENDC
                ;       EAE CODE
                        .IFDF   EAE
                        MOV     #MQ,R3  ;POINT TO MQ
                        CLR     @R3
                        MOV     R1,-(R3)        ;MOVE ARG
                        MOV     #NOR,R0 ;POINT TO NOR IN EAE
                        CLR     @R0     ;NORMALIZE FRACTION
                        SUB     (R0)+,R2        ;TELL EXPONENT
                        MOV     #2,@R0  ;SHIFT OUT NORMAL BIT BY LSH
                        MOV     @R3,R1  ;RESULT TO R1
                        .ENDC
16120 110166 NOU$27:    MOVB    R1,5(SP)        ;SAVE LOW ORDER FRACTION
      000005
16124 105001            CLRB    R1
16126 150201            BISB    R2,R1   ;COMBINE EXPONENT AND HIGH ORDER FRACTIO
16130 000301            SWAB    R1
16132 006026            ROR     (SP)+   ;GET SIGN
16134 006001            ROR     R1      ;INSERT SIGN IN RESULT
16136 106066            RORB    3(SP)
      000003
16142 010116            MOV     R1,@SP  ;OUTPUT RESULT
16144 000134 ZER$27:    JMP     @(R4)+
                        .ENDC
                        .ENDC
```

```
                     .TITLE  $MLD05
                     .IFDF   CND$28
                     .GLOBL  $MLD,$ERRA
             ;       $MLD      THE DOUBLE MULTIPLY ROUTINE
             ;
             ;       $MLD    V005A
             ;
             ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
             ;       CALLED IN POLISH MODE.
             ;       REPLACES THE TOP TWO DOUBLES ON THE STACK
             ;       WITH THEIR PRODUCT.
      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
      000010         A=8.
      000020         B=16.
      000014         RESLT=12.
      000002         SIGN=2
      000000         F0=%0
                     .IFDF   FPU
             $MLD:   .WORD   170011  ;;SETD
                     .WORD   172426  ;;LDD    (SP)+,F0          ;GET OPERAND
                     .WORD   171026  ;;MULD   (SP)+,F0          ;PRODUCT
                     .WORD   174046  ;;STD    F0,-(SP)          ;PRODUCT TO STAC
                     JMP     @(R4)+
                     .ENDC
                     .IFNDF  FPU
16146 010446 $MLD:   MOV     R4,-(SP)
16150 010546         MOV     R5,-(SP)
16152 006366         ASL     A+0-4(SP)          ;SHIFT MULTIPLICAND
      000004
16156 006146         ROL     -(SP)   ;KEEP SIGN
16160 005046         CLR     -(SP)   ;CLEAR EXPONENT
16162 116616         MOVB    A+1(SP),@SP        ;KEEP MULTIPLICAND EXPONENT
      000011
16166 001436         BEQ     ZER$28  ;JUMP IF ANSWER IS ZERO
16170 116666         MOVB    A(SP),A+1(SP)   ;SHIFT FRACTION LEFT
      000010
      000011
16176 000261         SEC             ;INSERT NORMAL BIT
16200 006066         ROR     A(SP)
      000010
16204 116666         MOVB    A+3(SP),A(SP)
      000013
      000010
16212 000366         SWAB    A+2(SP)
      000012
16216 116666         MOVB    A+5(SP),A+2(SP)
      000015
      000012
16224 000366         SWAB    A+4(SP)
```

```
         000014
48 16230 116666                 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 ZEN$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
```

```
            000230
83 16410 016604 B24$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
```

```
        ADC     R1
        MOV     R2,R3   ;DIVIDE BY 2**16
        MOV     R1,R2
        CLR     R1
        MOV     A(SP),-(SP)
        MOV     B+4+2(SP),@R4   ;GET A1*B3
        JSR     R5,EMUS28
        ADD     (SP)+,R2
        ADC     R1
        ADD     (SP)+,R3
        ADC     R2
        ADC     R1
        MOV     A+2(SP),-(SP)
        MOV     B+2+2(SP),@R4   ;GET A2*B2
        JSR     R5,EMUS28
        ADD     (SP)+,R2
        ADC     R1
        ADD     (SP)+,R3
        ADC     R2
        ADC     R1
        MOV     A+4(SP),-(SP)
        MOV     B+0+2(SP),@R4   ;GET A3*B1
        JSR     R5,EMUS28
        ADD     (SP)+,R2
        ADC     R1
        ADD     (SP)+,R3
        ADC     R2
        ADC     R1
        MOV     A(SP),-(SP)
        MOV     B+2+2(SP),@R4   ;GET A1*B2
        JSR     R5,EMUS28
        ADD     (SP)+,R1
        ADC     R0
        ADD     (SP)+,R2
        ADC     R1
        ADC     R0
        MOV     A+2(SP),-(SP)
        MOV     B+0+2(SP),@R4   ;GET A2*B1
        JSR     R5,EMUS28
        ADD     (SP)+,R1
        ADC     R0
        ADD     (SP)+,R2
        ADC     R1
        ADC     R0
        MOV     A(SP),-(SP)
        MOV     B+0+2(SP),@R4   ;GET A1*B1
        JSR     R5,EMUS28
        ADD     (SP)+,R0
        ADD     (SP)+,R1
        ADC     R0
        MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
        .ENDC
        .IFDF   MULDIV
        MOV     A(SP),-(SP)
        MOV     B+6+2(SP),R4    ;GET A1*B4
        JSR     PC,EMUS28
        MOV     R4,R2   ;RESULT TO PRODUCT
        MOV     R5,R3
        MOV     A+2(SP),-(SP)
        MOV     B+4+2(SP),R4    ;GET A2*B3
        JSR     PC,EMUS2Ø
        ADD     R4,R2   ;ADD TO PRODUCT
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     A+4(SP),-(SP)
        MOV     B+2+2(SP),R4    ;GET A3*B2
        JSR     PC,EMUS2Ø
        ADD     R4,R2
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     A+6(SP),-(SP)
        MOV     B+Ø+2(SP),R4    ;GET A4*B1
        JSR     PC,EMUS2Ø
        ADD     R4,R2
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     R2,R3   ;DIVIDE BY 2**16
        MOV     R1,R2
        CLR     R1
        MOV     A(SP),-(SP)
        MOV     B+4+2(SP),R4    ;GET A1*B3
        JSR     PC,EMUS2Ø
        ADD     R4,R2
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     A+2(SP),-(SP)
        MOV     B+2+2(SP),R4    ;GET A2*B2
        JSR     PC,EMUS2Ø
        ADD     R4,R2
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     A+4(SP),-(SP)
        MOV     B+Ø+2(SP),R4    ;GET A3*B1
        JSR     PC,EMUS2Ø
        ADD     R4,R2
        ADC     R1
        ADD     R5,R3
        ADC     R2
        ADC     R1
        MOV     A(SP),-(SP)
        MOV     B+2+2(SP),R4    ;GET A1*B2
        JSR     PC,EMUS2Ø
        ADD     R4,R1
        ADC     RØ
```

```
249                         ADD     R5,R2
250                         ADC     R1
251                         ADC     R0
252                         MOV     A+2(SP),-(SP)
253                         MOV     B+0+2(SP),R4     ;GET A2*B1
254                         JSR     PC,EMU$28
255                         ADD     R4,R1
256                         ADC     R0
257                         ADD     R5,R2
258                         ADC     R1
259                         ADC     R0
260                         MOV     A(SP),-(SP)
261                         MOV     B+0+2(SP),R4     ;GET A1*B1
262                         JSR     PC,EMU$28
263                         ADD     R4,R0
264                         ADD     R5,R1
265                         ADC     R0
266                         MOV     (SP)+,R4         ;GET SUM OF EXPONENTS
267                         .ENDC
268 6434 006303             ASL     R3       ;SHIFT OUT NORMAL BIT
269 6436 006102             ROL     R2
270 6440 006101             ROL     R1
271 6442 006100             ROL     R0
272 6444 103405             BCS     NOM$28   ;JUMP IF IT WAS FOUND
273 6446 006303             ASL     R3
274 6450 006102             ROL     R2
275 6452 006101             ROL     R1
276 6454 006100             ROL     R0       ;MUST HAVE GOT IT NOW
277 6456 005304             DEC     R4       ;ADJUST EXPONENT
278 6460 162704 NOM$28:     SUB     #200,R4 ;TAKE OUT ONE OF THE EXCESS 128'S
         000200
279 6464 003453             BLE     UND$28   ;JUMP IF UNDERFLOW
280 6466 022704             CMP     #377,R4
         000377
281 6472 002445             BLT     OVR$28   ;JUMP IF OVERFLOW
282 6474 105003             CLRB    R3
283 6476 150203             BISB    R2,R3    ;SHIFT FRACTION RIGHT
284 6500 000303             SWAB    R3
285 6502 105002             CLRB    R2
286 6504 150102             BISB    R1,R2
287 6506 000302             SWAB    R2
288 6510 105001             CLRB    R1
289 6512 150001             BISB    R0,R1
290 6514 000301             SWAB    R1
291 6516 105000             CLRB    R0
292 6520 150400             BISB    R4,R0
293 6522 000300             SWAB    R0
294 6524 006026             ROR     (SP)+    ;GET PRODUCT SIGN
295 6526 006000             ROR     R0       ;INSERT IT IN RESULT
296 6530 006001             ROR     R1
297 6532 006002             ROR     R2
298 6534 006003             ROR     R3
299 6536 005503             ADC     R3       ;ROUND RESULT
300 6540 005502             ADC     R2
301 6542 005501             ADC     R1
302 6544 005500             ADC     R0
303 6546 103410             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 ZER$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
```

```
6730 ØØØ2Ø7          RTS      PC        /RETURN TO CALLER
6732 ØØ5366 MT2$28!  DEC      2(SP)     /DO ONLY 15 BITS THIS PASS
     ØØØØØ2
6736 ØØ62Ø4 MTØ$28!  ASR      R4        /TEST NEXT MULTIPLIER BIT
674Ø 1Ø3ØØ7          BCC      XØØ$28    /JUMP IF Ø
6742 Ø666Ø1          ADD      A+2+4(SP),R1      /USE ONLY HIGH ORDER MULTIPLICAN
     ØØØØ16
6746 ØØ55ØØ          ADC      RØ
675Ø ØØ55Ø5          ADC      R5
6752 Ø666ØØ          ADD      A+Ø+4(SP),RØ
     ØØØØ14
6756 ØØ55Ø5          ADC      R5
676Ø ØØ62Ø5 XØØ$28!  ASR      R5        /RECOVER ANY OVERFLOW
6762 ØØ6ØØØ          ROR      RØ
6764 ØØ6ØØ1          ROR      R1
6766 ØØ6ØØ2          ROR      R2
677Ø ØØ6ØØ3          ROR      R3
6772 ØØ5366          DEC      2(SP)     /COUNT LOOP
     ØØØØØ2
6776 ØØ3357          BGT      MTØ$28
7ØØØ ØØØ2Ø7          RTS      PC        /RETURN TO CALLER
                     .ENDC
                     .IFDF    EAE
            EMU$28!  CLR      @SP       /CLEAR PRODUCT
                     TST      @R4
                     BEQ      MZ$28     /JUMP IF MULTIPLIER Ø
                     BGT      MPL$28
                     TST      2(SP)     /TEST MULTIPLICAND
                     BEQ      MZ$28     /JUMP IF Ø
                     BGT      MNG$28    /JUMP IF +*=
                     ADD      (R4)+,@SP         /CORRECT 2'S COMPLEMENT
                     ADD      2(SP),@SP
                     BR       EML$28
            MPL$28!  TST      2(SP)     /TEST MULTIPLICAND
                     BEQ      MZ$28     /JUMP IF Ø
                     BGT      MLQ$28    /JUMP IF +
                     ADD      (R4)+,@SP
                     BR       EML$28
            MNG$28!  ADD      2(SP),@SP
            MLQ$28!  TST      (R4)+     /POINT TO MUL
            EML$28!  MOV      2(SP),@R4         /MULTIPLY
                     MOV      =(R4),2(SP)       /GET PRODUCT
                     ADD      =(R4),@SP
                     TST      (R4)+     /POINT TO MQ
                     JMP      @R5       /RETURN
            MZ$28!   CLR      2(SP)     /RETURN Ø
                     JMP      @R5
                     .ENDC
                     .IFDF    MULDIV
            EMU$28!  CLR      =(SP)     /CLEAR HIGH PRODUCT
                     TST      R4        /TEST MULTIPLICAND
                     BEQ      MZ$28     /JUMP IF Ø
                     BGT      MPL$28    /+
                     TST      4(SP)     /TEST MULTIPLIER
                     BEQ      MZ$28     /JUMP IF Ø
                     BGT      MN1$28    /+
                     BR       MNG$28
```

```
        MPL$28: TST     4(SP)   ;TEST MULTIPLIER
                BEQ     MZ$28   ;JUMP IF Ø
                BGT     MLQ$28  ;+
                ADD     R4,@SP
                BR      MLQ$28
        MNG$28: ADD     R4,@SP
        MN1$28: ADD     4(SP),@SP
        MLQ$28: .WORD   Ø7Ø466,4        ;;MUL   4(SP),R4        ;GET PRO
        MDN$28: ADD     (SP)+,R4        ;ADD IN HIGH ORDER PARTS
                MOV     (SP)+,@SP       ;FLUSH MULTIPLIER
                RTS     PC      ;RETURN
        MZ$28:  CLR     R4      ;RESULT IS Ø
                CLR     R5
                BR      MDN$28
                .ENDC
                .ENDC
                .ENDC
```

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

```
57 1711Ø ØØ54Ø3                 NEG     R3        ;MAKE -
58 17114 1ØØØ15                 BPL     OVR$29    ;JUMP IF TOO BIG
59 17116 ØØ6ØØØ                 ROR     RØ        ;GET PRODUCT SIGN
6Ø 1712Ø 1Ø34Ø2                 BCS     OUT$29    ;JUMP IF -
61 17122 ØØ54Ø3                 NEG     R3        ;MAKE +
62 17124 1Ø2411                 BVS     OVR$29
63 17126 Ø1Ø316 OUT$29: MOV     R3,@SP    ;MOVE OUT RESULT
64 1713Ø ØØØ134                 JMP     @(R4)+    ;RETURN
65 17132 ØØ54Ø3 NGM$29: NEG     R3        ;TEST FOR OCTAL 1ØØØØØ
66 17134 1Ø2ØØ5                 BVC     OVR$29    ;JUMP IF NOT
67 17136 ØØ6ØØØ                 ROR     RØ        ;TEST FOR NEGATIVE RESULT
68 1714Ø 1Ø3772                 BCS     OUT$29    ;YES, WE CAN HANDLE THIS
69 17142 ØØØ4Ø2                 BR      OVR$29    ;OVERFLOW
7Ø 17144 ØØ5Ø16 ZER$29: CLR     @SP       ;CLEAR PRODUCT
71 17146 ØØØ134                 JMP     @(R4)+    ;RETURN
72                              .ENDC
73                      ;       $MLI CODE FOR THE EAE
74                              .IFDF   EAE
75                      $MLI:   MOV     #MQ,RØ  ;GET MQ ADDRESS
76                              MOV     (SP)+,(RØ)+     ;MULTIPLIER TO MQ
77                              MOV     (SP)+,@RØ       ;MULTIPLICAND TO MUL
78                              MOV     -(RØ),-(SP)     ;PRODUCT TO STACK
79                              BITB    #2,SR$29
8Ø                              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)+,RØ        ;MOVE MULTIPLIER
86                              .WORD   Ø7ØØ26  ;;MUL   (SP)+,RØ        ;MULTIPLY
87                              MOV     R1,-(SP)        ;PUSH PRODUCT
88                              BCS     OVR$29  ;JUMP IF OVERFLOW
89                              JMP     @(R4)+
9Ø                              .ENDC
91 1715Ø ØØ5Ø16 OVR$29: CLR     (SP)    ;RETURN Ø
92 17152 ØØ4567                 JSR     R5,$ERR ;ERROR 3,14
         ØØ263Ø
93 17156 ØØØ134                 JMP     @(R4)+
94 1716Ø    ØØ3                 .BYTE   3
95 17161    Ø16                 .BYTE   14.
96                              .ENDC
```

```
                                .TITLE  $MLR05
                                .IFDF   CND$30
                                .GLOBL  $MLR,$ERRA
                        ;       $MLR    THE REAL MULTIPLY ROUTINE
                        ;
                        ;
                        ;       $MLR    V005A
                        ;
                        ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        ;
                        ;       CALLED IN POLISH MODE.
                        ;       REPLACES THE TOP TWO REALS ON THE STACK
                        ;       WITH THEIR PRODUCT.
         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
         177311                 SR=177311
         177314                 LSH=177314
         000000                 F0=%0
         000010 A=8.
         000014 B=12.
         000010 RESLT=8.
         000002 SIGN=2
                                .IFDF   FPU
                        $MLR:   .WORD   170001  ;;SETF
                                .WORD   172426  ;;LDF   (SP)+,F0          ;GET MULTIPLICAN
                                .WORD   171026  ;;MULF  (SP)+,F0          ;MULTIPLY
                                .WORD   174046  ;;STF   F0,-(SP)          ;PRODUCT TO STAC
                                JMP     @(R4)+
                                .ENDC
                                .IFNDF  FPU
   17102 010446 $MLR:   MOV     R4,-(SP)
   17104 010546                 MOV     R5,-(SP)
                                .IFNDF  EAE&MULDIV
   17106 016602                 MOV     A+0-4(SP),R2
         000004
   17172 006302                 ASL     R2      ;SHIFT MULTIPLICAND
   17174 006146                 ROL     -(SP)   ;KEEP SIGN
   17176 005046                 CLR     -(SP)   ;CLEAR EXPONENT
   17200 000302                 SWAB    R2
   17202 110216                 MOVB    R2,@SP  ;KEEP MULTIPLICAND EXPONENT
   17204 001507                 BEQ     ZER$30  ;JUMP IF ANSWER IS ZERO
   17206 000261                 SEC             ;INSERT NORMAL BIT
   17210 006002                 ROR     R2
   17212 105002                 CLRB    R2
   17214 156602                 BISB    A+3(SP),R2
         000013
   17220 005003                 CLR     R3
   17222 156603                 BISB    A+2(SP),R3
         000012
   17226 000303                 SWAB    R3
   17230 006366            ASL     B(SP)   ;SHIFT HIGH MULTIPLIER
         000014
   17234 005566            ADC     SIGN(SP)        ;GET PRODUCT SIGN
         000002
   17240 105766            TSTB    B+1(SP)
         000015
   17244 001407            BEQ     ZE1$30  ;JUMP IF ZERO
   17246 006066            ROR     B(SP)   ;SIGN IS NOW ZERO
         000014
   17252 005000            CLR     R0      ;CLEAR PRODUCT
   17254 005001            CLR     R1
   17256 016604            MOV     B+2(SP),R4      ;GET LOW ORDER MULTIPLIER
         000016
   17262 001406            BEQ     B2Z$30
   17264 012705 B2N$30:    MOV     #15.,R5
         000017
   17270 004767            JSR     PC,MT0$30
         000220
   17274 004767            JSR     PC,MLT$30       ;DO LAST LOW BIT FULL PRECISION
         000160
   17300 016604 B2Z$30:    MOV     B(SP),R4        ;GET HIGH ORDER BITS
         000014
   17304 012705            MOV     #7,R5   ;THERE ARE ONLY SEVEN OF THEM
         000007
   17310 004767            JSR     PC,MLT$30
         000144
   17314 004767            JSR     PC,MT1$30       ;GO DO THE NORMAL BIT
         000144
   17320 062604            ADD     (SP)+,R4        ;ADD EXPONENTS
                           .ENDC
                        ;  EAE CODE
                           .IFDF   EAE
                        ;  (A1+A2*2**-16)*(B1+B2*2**-16)
                           MOV     #MQ,R4  ;POINT TO MQ
                           MOV     #100000,R5      ;GET LEADING BIT
                           MOV     B+2-4(SP),@R4   ;LOW ORDER B TO MQ
                           MOV     B+0-4(SP),-(R4) ;HIGH TO AC
                           BEQ     ZER$30  ;JUMP IF 0
                           INC     @#LSH   ;GET SIGN
                           RORB    @#SR
                           ROL     -(SP)   ;SAVE IT
                           MOV     (R4)+,-(SP)     ;SAVE EXPONENT
                           CLRB    @SP     ;RIGHT JUSTIFY IT
                           SWAB    @SP
                           MOV     #7,@#LSH        ;MOVE FRACTION LEFT
                           MOV     @R4,-(SP)       ;SAVE B2
                           BIS     R5,-(R4)        ;INSERT NORMAL BIT
                           MOV     (R4)+,-(SP)     ;SAVE B1
                           MOV     A+2+4(SP),@R4   ;LOW ORDER A TO MQ
                           MOV     A+0+4(SP),-(R4) ;HIGH TO AC
                           BEQ     ZE2$30  ;JUMP IF 0
                           INC     @#LSH   ;GET SIGN
                           RORB    @#SR
                           ADC     6(SP)   ;GET RESULT SIGN
                           MOV     @R4,R3  ;GET EXPONENT
                           CLRB    R3
                           SWAB    R3
                        ADD     R3,4(SP)            ;GET SUM OF EXPONENTS
                        MOV     #7,@#LSH            ;LEFT JUSTIFY FRACTION
                        MOV     (R4)+,R2            ;SAVE A1
                        BIS     R5,R2   ;INSERT NORMAL BIT
                        CLR     R0      ;CLEAR PRODUCT
                        CLR     R1
                        MOV     (R4)+,R3            ;SAVE A2
                        BNE     A2N$30
                        TST     -(R4)   ;POINT TO MQ
                        BR      A2Z$30  ;SHORT CUT
                A2N$30: MOV     @SP,@R4 ;GET B1*A2
                        CMP     -(R4),-(R4)         ;POINT TO AC
                        ADD     R3,@R4  ;A2, 2'S COMP CORRECTION
                        TST     R3
                        BPL     A2P$30
                        ADD     @SP,@R4 ;B1, CORRECTION
                A2P$30: MOV     (R4)+,R1            ;HIGH PRODUCT TO R1
                A2Z$30: MOV     2(SP),(R4)+         ;B2 TO MQ
                        BNE     B2N$30
                        TST     -(R4)   ;POINT TO MQ
                        BR      B2Z$30  ;SHORT CUT
                B2N$30: MOV     R2,@R4  ;GET B2*A1
                        CMP     -(R4),-(R4)         ;POINT TO AC
                        ADD     2(SP),@R4           ;B2, CORRECTION
                        TST     2(SP)
                        BPL     B2P$30  ;JUMP IF B2 +
                        ADD     R2,@R4  ;A1, CORRECTION
                B2P$30: ADD     (R4)+,R1            ;HIGH PRODUCT TO R1
                        ADC     R0
                B2Z$30: MOV     R2,(R4)+            ;A1 TO MQ
                        ADD     R2,R0
                        MOV     @SP,@R4 ;GET A1*B1
                        ADD     (SP)+,R0
                        ADD     -(R4),R1
                        ADC     R0
                        ADD     -(R4),R0            ;AC+R0
                        TST     (SP)+   ;POP B2
                        MOV     (SP)+,R4            ;GET SUM OF EXPONENTS
                        .ENDC
                ;       MUL/DIV CODE
                        .IFDF   MULDIV
                ;       (A1+A2*2**-16)*(B1+B2*2**-16)
                        MOV     B+2-4(SP),R5    ;LOW ORDER B
                        MOV     B+0-4(SP),R4    ;HIGH ORDER
                        BEQ     ZER$30
                        .WORD   073427,1            ;;      ASHC    #1,R4   ;GET SIG
                        ROL     -(SP)   ;SAVE IT
                        MOV     R4,-(SP)            ;SAVE EXPONENT
                        CLRB    @SP
                        SWAB    @SP     ;RIGHT JUSTIFY
                        .WORD   073427,7            ;;      ASHC    #7,R4   ;LEFT JU
                        MOV     R5,-(SP)            ;SAVE B2
                        BIS     #100000,R4          ;INSERT NORMAL BIT
                        MOV     R4,-(SP)            ;SAVE B1
                        MOV     A+2+4(SP),R3    ;GET A2
                        MOV     A+0+4(SP),R2    ;GET A1
                        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 003430                  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 ØØ6ØØ1            ROR     R1
213 7374 ØØ55Ø1            ADC     R1
214 7376 ØØ55ØØ            ADC     RØ
215 74ØØ 1Ø3414            BCS     OV1$3Ø  ;JUMP IF OVERFLOW ON ROUND
216 74Ø2 1Ø2413            BVS     OV1$3Ø
217 74Ø4 Ø1ØØ66 OUT$3Ø:    MOV     RØ,RESLT(SP)     ;PUT OUT ANSWER
         ØØØØ1Ø
218 741Ø Ø1Ø166            MOV     R1,RESLT+2(SP)
         ØØØØ12
219 7414 Ø126Ø5            MOV     (SP)+,R5
22Ø 7416 Ø126Ø4            MOV     (SP)+,R4
221 742Ø Ø22626            CMP     (SP)+,(SP)+      ;FLUSH TOP ARGUMENT
222 7422 ØØØ134            JMP     @(R4)+  ;RETURN
223                        .IFDF   EAE!MULDIV
224             ZE2$3Ø:    CMP     (SP)+,(SP)+      ;POP B1,B2
225                        .ENDC
226 7424 Ø22626 ZE1$3Ø:    CMP     (SP)+,(SP)+      ;POP SIGN AND EXPONENT
227 7426 ØØØ411            BR      ZER$3Ø
228 743Ø ØØ5726 OVR$3Ø:    TST     (SP)+   ;FLUSH SIGN
229 7432 Ø127ØØ OV1$3Ø:    MOV     #6ØØ3,RØ         ;ERROR 3,12
         ØØ6ØØ3
23Ø 7436 ØØØ4Ø3            BR      ECL$3Ø
231 744Ø Ø127ØØ UNU$3Ø:    MOV     #34Ø5,RØ         ;ERROR 5,7
         ØØ34Ø5
232 7444 ØØ5726            TST     (SP)+   ;FLUSH SIGN
233 7446 ØØ4567 ECL$3Ø:    JSR     R5,$ERRA         ;CALL ERROR
         ØØ2344
234 7452 ØØ5ØØØ ZER$3Ø:    CLR     RØ      ;CLEAR RESULT
235 7454 ØØ5ØØ1            CLR     R1
236 7456 ØØØ752            BR      OUT$3Ø
237                        .IFNDF  EAE&MULDIV
238 746Ø ØØ62Ø4 MLT$3Ø:    ASR     R4      ;TEST NEXT MULTIPLIER BIT
239 7462 1Ø3ØØ4            BCC     XØ$3Ø   ;JUMP IF IT IS Ø
24Ø 7464 Ø6Ø3Ø1 MT1$3Ø:    ADD     R3,R1
241 7466 ØØ55ØØ            ADC     RØ
242 747Ø 1Ø34Ø6            BCS     COV$3Ø
243 7472 Ø6Ø2ØØ            ADD     R2,RØ
244 7474 ØØ6ØØØ XØ$3Ø:     ROR     RØ      ;NOW SHIFT PRODUCT
245 7476 ØØ6ØØ1            ROR     R1
246 75ØØ ØØ53Ø5            DEC     R5      ;COUNT LOOP
247 75Ø2 ØØ3366            BGT     MLT$3Ø  ;AGAIN PLEASE
248 75Ø4 ØØØ2Ø7            RTS     PC      ;RETURN TO CALLER
249 75Ø6 Ø6Ø2ØØ COV$3Ø:    ADD     R2,RØ   ;FIRST ADD OVERFLOWED RØ
25Ø 751Ø ØØØ261            SEC             ;SHOW THIS OVERFLOW TO SHIFT
251 7512 ØØØ77Ø            BR      XØ$3Ø
252 7514 ØØ62Ø4 MTØ$3Ø:    ASR     R4      ;REDUCED PRECISION MULTIPLY
253 7516 1Ø3ØØ1            BCC     XØØ$3Ø
254 752Ø Ø6Ø2ØØ            ADD     R2,RØ   ;USE ONLY HIGH ORDER MULTIPLICAND
255 7522 ØØ6ØØØ XØØ$3Ø:    ROR     RØ
256 7524 ØØ6ØØ1            ROR     R1
257 7526 ØØ53Ø5            DEC     R5
258 753Ø ØØ3371            BGT     MTØ$3Ø
259 7532 ØØØ2Ø7            RTS     PC
26Ø                        .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
```

```
                            .TITLE  $PPRØ4
                            .IFDF   CNDS$Ø3
                    ;
                    ;       $POPR5  VØØ4A
                    ;
                    ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                    ;
                    ;
             ØØØØØØ RØ=%Ø
             ØØØØØ1 R1=%1
             ØØØØØ2 R2=%2
             ØØØØØ3 R3=%3
             ØØØØØ4 R4=%4
             ØØØØØ5 R5=%5
             ØØØØØ6 SP=%6
             ØØØØØ7 PC=%7
                    ;
                    ; THIS ROUTINE REMOVES TWO OR FOUR ITEMS FROM THE STACK
                    ;       AND PLACES THEM IN REGISTERS RØ-R3.  IT IS USED IN EXTER
                    ;       FUNCTIONS TO RETURN THE FUNCTION VALUE IN THE REGISTERS
                    ;
                            .GLOBL  $POPR5,$POPR4,$POPR3
                    ;
       17576        $POPR5:
       17576 Ø126ØØ $POPR4: MOV     (SP)+,RØ        ;POP FOUR WORDS
       176ØØ Ø126Ø1         MOV     (SP)+,R1
       176Ø2 Ø126Ø2         MOV     (SP)+,R2
       176Ø4 Ø126Ø3         MOV     (SP)+,R3
       176Ø6 ØØØ134         JMP     @(R4)+
       1761Ø Ø126ØØ $POPR3: MOV     (SP)+,RØ        ;POP TWO WORDS
       17612 Ø126Ø1         MOV     (SP)+,R1
       17614 ØØØ134         JMP     @(R4)+
                    ;
                            .ENDC
```

```
                               .TITLE  $RDØ2
                               .IFDF   CND$34
                               .GLOBL  $RD
                        ;      $RD     THE REAL TO DOUBLE PRECISION CONVERTER
                        ;
                        ;      $RD     VØØ2A
                        ;
                        ;      COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        ;      APPEND ZEROS TO THE TOP STACK ITEM TO
                        ;      MAKE IT DOUBLE PRECISION FORMAT
         ØØØØØ4                R4=%4
         ØØØØØ6                SP=%6
         ØØØØØØ                FØ=%Ø
         ØØØØØ1                F1=%1
                               .IFDF   FPU
                        $RD:   .WORD   170011  ;;SETD
                               .WORD   177426  ;;LDCFD (SP)+,FØ        ;CONVERT ARG
                               .WORD   174046  ;;STD   FØ,-(SP)
                               JMP     @(R4)+
                               .ENDC
                               .IFNDF  FPU
   17616 Ø16646 $RD:   MOV     2(SP),-(SP)      ;MOVE LOW ORDER PART
         ØØØØØ2
   17622 Ø16646                MOV     2(SP),-(SP)      ;MOVE HIGH ORDER PART
         ØØØØØ2
   17626 ØØ5Ø66                CLR     4(SP)    ;INSERT TRAILING ZEROS
         ØØØØØ4
   17632 ØØ5Ø66                CLR     6(SP)
         ØØØØØ6
   17636 ØØØ134                JMP     @(R4)+
                               .ENDC
                               .ENDC
```

```
                              .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     DNE$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:
```

```
                     .IFNDF  EAE&MULDIV
17720 006103         ROL     R3      ;GET NEXT BIT
17722 006102         ROL     R2
17724 005301 DEC$35: DEC     R1      ;DECREASE EXPONENT
17726 003374         BGT     SFT$35  ;GO AGAIN IF NOT DONE
                     .ENDC
             ;
             ;       EAE CODE
                     .IFDF   EAE
                     MOV     #MQ,R0  ;POINT TO MQ
                     MOV     R3,@R0  ;INSERT FRACTION
                     MOV     R2,-(R0)
                     MOV     R1,@#LSH        ;SHIFT LEFT
                     MOV     @R0,R2  ;RESULT TO REG
                     .ENDC
             ;       MULDIV CODE
                     .IFDF   MULDIV
                     .WORD   073201  ;;ASHC  R1,R2
                     .ENDC
17730 005402 DNE$35: NEG     R2      ;MAKE -
17732 102406         BVS     NGM$35  ;JUMP IF POSSIBLE NEGMAX
17734 003007         BGT     OVR$35  ;JUMP IF MORE THAN 15 BITS
17736 006026 SGN$35: ROR     (SP)+   ;GET SIGN
17740 103401         BCS     OUT$35  ;JUMP IF -
17742 005402         NEG     R2      ;- RESULT
17744 010216 OUT$35: MOV     R2,@SP  ;STORE INTEGER RESULT
17746 000134         JMP     @(R4)+  ;RETURN TO CALLER
17750 006026 NGM$35: ROR     (SP)+
17752 103774         BCS     OUT$35  ;OK IF RESULT TO BE -
17754 005746 OVR$35: TST     -(SP)   ;FAKE SIGN
17756 004567         JSR     R5,$ERR ;ERROR 3,22
      002024
17762 000401         BR      ZER$35
17764    003         .BYTE   3
17765    026         .BYTE   22.
17766 005002 ZER$35: CLR     R2      ;ANSWER IS 0
17770 000762         BR      SGN$35
                     .ENDC
                     .ENDC
```

```
                               .TITLE  $SGL02
                               .IFDF   CND$36
                       ;
                       ;       SNGL    V002A
                       ;
                       ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                       ;
                               .GLOBL  SNGL,$ERR
                       ;       THE FORTRAN SNGL FUNCTION
                       ;       CALLING SEQUENCE:
                       ;       JSR     R5,SNGL
                       ;       BR      A
                       ;       .WORD   ARGUMENT ADDRESS
                       ;A:
                       ;       RETURNS THE ARGUMENT ROUNDED TO SINGLE
                       ;       PRECISON REAL FORMAT IN R0, R1.
                       ;
        000000                 R0=%0
        000001                 R1=%1
        000004                 R4=%4
        000005                 R5=%5
 17772  016504 SNGL:   MOV     2(R5),R4         ;GET ADDRESS
        000002
 17776  012400         MOV     (R4)+,R0         ;GET HIGH ORDER
 20000  012401         MOV     (R4)+,R1         ;GET LOW ORDER
 20002  011404         MOV     @R4,R4   ;GET NEXT WORD
 20004  006104         ROL     R4       ;GET ROUND BIT
 20006  005501         ADC     R1       ;ROUND REAL
 20010  005500         ADC     R0
 20012  103402         BCS     OVR$36   ;JUMP IF OVERFLOW ON ROUND
 20014  102401         BVS     OVR$36
 20016  000205         RTS     R5       ;RETURN TO CALLER
 20020  004567 OVR$36: JSR     R5,$ERR  ;ERROR 4,12
        001762
 20024  000205         RTS     R5
 20026     004         .BYTE   4
 20027     014         .BYTE   12.
                       .ENDC
```

```
                                .TITLE  $SIN04
                                .IFDF   CND$37
                        ;
                        ;       SINCOS  V004A
                        ;
                        ; COPYRIGHT 1971,1972 DIGITAL EQUIPMENT CORPORATION, MAYNARD,MAS
                        ;
                                .GLOBL  SIN,COS;
                                .IFNDF  FPU
                                .GLOBL  $ADR,$MLR,$SBR,$DVR,$INTR,$POLSH
                                .ENDC
                        ;       SIN     COS     THE REAL SIN AND COSINE FUNCTIONS
                        ;       CALLING SEQUENCE:
                        ;       JSR     R5,SIN  (OR COS)
                        ;       BR      A
                        ;       .WORD   ARG ADDRESS
                        ;A:
                        ;       RETURNS SIN OR COS OF ARG IN R0 AND R1
        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
        000002                  F2=%2
        000003                  F3=%3
                                .IFNDF  FPU
 20030  016504  COS:            MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
        000002
 20034  005046                  CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
 20036  016446                  MOV     2(R4),-(SP)     ;PUSH ARGUMENT
        000002
 20042  011446                  MOV     @R4,-(SP)
 20044  012746                  MOV     #007733,-(SP)   ;PUSH PI/2
        007733
 20050  012746                  MOV     #040311,-(SP)
        040311
 20054  004467                  JSR     R4,$POLSH       ;ENTER POLISH MODE
        001564
 20060  002010'                 .WORD   $ADR,SNC$37     ;COS(X)=SIN(X+PI/2)
 20062  020100'
 20064  016504  SIN:            MOV     2(R5),R4        ;GET ARGUMENT ADDRESS
        000002
 20070  005046                  CLR     -(SP)   ;MAKE ROOM FOR QUADRANT FLAG
 20072  016446                  MOV     2(R4),-(SP)
        000002
 20076  011446                  MOV     @R4,-(SP)       ;PUSH ARGUMENT
 20100  006316  SNC$37:         ASL     @SP     ;REMOVE AND SAVE SIGN
 20102  006066                  ROR     4(SP)   ;IN QUADRANT FLAG
        000004
 20106  006016                  ROR     @SP
 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 005726            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 QTS$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       SNC$37;
127                SIN:      SETD     ;                  DOUBLE PRECISON FP
128                          LDCFD    @2(R5),F0;         GET ARGUMENT
129                SNC$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
                    XPD$37:  MULF     F2,F1;
                             DEC      R1;               COUN
                             ADDF     (R0)+,F1;         F1:= Z*F1 + C(I)
                             BGT      XPD$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    121041,064302;
                    ;
                    ;        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
```

```
                                 .TITLE  $TNH02
                                 .IFDF   CND$38
                                 .GLOBL  TANH,EXP,$ADR,$SBR,$MLR,$DVR,$FCALL
                                 .GLOBL  $POLSH,$PSHR3
                        ;        THE FORTRAN TANH FUNCTION
                        ;        TANH    V002A
                        ;        COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        ;        CALLING SEQUNCE:
                        ;        JSR     R5,TANH
                        ;        BR      A
                        ;        .WORD   ARGUMENT ADDRESS
                        ;A:
                        ;        RETURNS (EXP(2*ARG) -1)/(EXP(2*ARG)+1) IN R0,R1.
                        ;
       000000                    R0=%0
       000001                    R1=%1
       000004                    R4=%4
       000005                    R5=%5
       000006                    SP=%6
       000007                    PC=%7
 20404 010546 TANH:      MOV     R5,-(SP)         ;SAVE RETURN POINTER
 20406 016505            MOV     2(R5),R5         ;GET ARG ADDRESS
       000002
 20412 011500            MOV     @R5,R0  ;GET HIGH ORDER ARG
 20414 001554            BEQ     ZER$38  ;JUMP IF ARG=0
 20416 006300            ASL     R0
 20420 105000            CLRB    R0
 20422 000300            SWAB    R0      ;GET EXPONENT
 20424 020027            CMP     R0,#205
       000205
 20430 002410            BLT     STE$38  ;JUMP IF ABS(ARG) <16.
 20432 012700            MOV     #40200,R0        ;ANSWER IS 1.*SIGN(ARG)
       040200
 20436 005001            CLR     R1
 20440 005715            TST     @R5     ;TEST ARG SIGN
 20442 002052            BGE     OUT$38
 20444 062700            ADD     #100000,R0       ;MAKE -1.
       100000
 20450 000447            BR      OUT$38
 20452 020027 STE$38:    CMP     R0,#177
       000177
 20456 003007            BGT     TAN$38  ;JUMP IF >1/2
 20460 020027            CMP     R0,#164
       000164
 20464 002043            BGE     SML$38  ;USE CONTINUED FRACTION FOR THIS RANGE
 20466 016501            MOV     2(R5),R1
       000002
 20472 011500            MOV     @R5,R0  ;IF ABS(X)<2**-12, LET TANH=X
 20474 000435            BR      OUT$38
 20476 016546 TAN$38:    MOV     2(R5),-(SP)      ;PUSH 2*ARG ON STACK
       000002
 20502 011546            MOV     @R5,-(SP)
 20504 062716            ADD     #200,@SP         ;DOUBLE ARG
       000200
 20510 010605            MOV     SP,R5   ;SET UP CALL TO EXP, ARG POINTER
 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 016000 ONE$38: MOV     4(SP),R0          ;GET XSQUARE AGAIN
         000004
76 20672 016001         MOV     6(SP),R1
```

```
             ØØØØØ6
77 2Ø676 ØØ5Ø66          CLR     6(SP)    ;INSERT A 1.
             ØØØØØ6
78 2Ø7Ø2 Ø12766          MOV     #4Ø2ØØ,4(SP)
             Ø4Ø2ØØ
             ØØØØØ4
79 2Ø71Ø ØØØ134          JMP     @(R4)+
8Ø 2Ø712 Ø12746 P45$38:  MOV     #136237,-(SP)     ;PUSH 45.1642
             136237
81 2Ø716 Ø12746          MOV     #414Ø4,-(SP)
             Ø41464
82 2Ø722 Ø12746 P15$38:  MOV     #1657Ø7,-(SP)     ;PUSH 1Ø5.46Ø5
             1657Ø7
83 2Ø726 Ø12746          MOV     #41722,-(SP)
             Ø41722
84 2Ø732 ØØØ134          JMP     @(R4)+
85 2Ø734 Ø12746 P35$38:  MOV     #116457,-(SP)     ;PUSH 35.1535
             116457
86 2Ø74Ø Ø12746          MOV     #41414,-(SP)
             Ø41414
87 2Ø744 ØØØ134          JMP     @(R4)+
88 2Ø746 ØØ5ØØØ ZER$38:  CLR     RØ
89 2Ø75Ø ØØ5ØØ1          CLR     R1
9Ø 2Ø752 ØØØ7Ø5          BR      OUT$38
91                ;
92 2Ø754 Ø12Ø66 UP$38:   MOV     (SP)+,1Ø.(SP)     ;MOVE STACK ITEM UP
             ØØØØ12
93 2Ø76Ø Ø12Ø66          MOV     (SP)+,1Ø.(SP)
             ØØØØ12
94 2Ø764 ØØØ134          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 21Ø14 Ø165Ø4              MOV      4(R5),R4         ;GET SECOND ARG ADDRESS
         ØØØØØ4
57 21Ø2Ø Ø1644Ø              MOV      2(R4),-(SP)      ;GET SECOND ARG
         ØØØØØ2
58 21Ø24 Ø1144Ø              MOV      @R4,-(SP)
59 21Ø26 Ø116Ø1              MOV      @SP,R1   ;ARG2 TO R1
6Ø 21Ø3Ø ØØ1437              BEQ      INF$39   ;JUMP IF DENOMINATOR IS Ø
61 21Ø32 ØØ63ØØ              ASL      RØ       ;GET ABS VAL ARG1
62 21Ø34 1Ø5ØØØ              CLRB     RØ       ;GET EXPONENT
63 21Ø36 ØØØ3ØØ              SWAB     RØ
64 21Ø4Ø ØØ63Ø1              ASL      R1
65 21Ø42 1Ø5ØØ1              CLRB     R1       ;GET EXPONENT ARG2
66 21Ø44 ØØØ3Ø1              SWAB     R1
67 21Ø46 16Ø1ØØ              SUB      R1,RØ    ;GET EXPONENT DIFFERENCE
68 21Ø5Ø Ø227ØØ              CMP      #26.,RØ ;CHECK MAGNITUDE
         ØØØØ32
69 21Ø54 ØØ2425              BLT      INF$39   ;TREAT AS INFINITY
7Ø 21Ø56 ØØ4467 DIV$39: JSR       R4,$POLSH
         ØØØ562
71 21Ø62 Ø13256'             .WORD    $DVR,UPL$39      ;GET ARG1/ARG2
   21Ø64 Ø21Ø66'
72 21Ø66 ØØ5775 UPL$39: TST       @4(R5)   ;IF ARG2 >Ø, BIAS =Ø
         ØØØØØ4
73 21Ø72 ØØ2Ø14              BGE      ATE$39   ;IF ARG2<Ø, BIAS=SIGN(ARG1)*PI
74 21Ø74 Ø12766              MOV      #Ø4Ø511,8.(SP)   ;PI
         Ø4Ø511
         ØØØØ1Ø
75 211Ø2 Ø12766              MOV      #ØØ7733,1Ø.(SP)
         ØØ7733
         ØØØØ12
76 2111Ø ØØ5775              TST      @2(R5)   ;TEST ARG1
         ØØØØØ2
77 21114 ØØ2ØØ3              BGE      ATE$39
78 21116 Ø62766              ADD      #1ØØØØØ,8.(SP)   ;-PI
         1ØØØØØ
         ØØØØ1Ø
79 21124 ØØ571Ø ATE$39: TST       @SP      ;SET CODES
8Ø 21126 ØØØ426              BR       AT1$39   ;JOIN MAIN ROUTINE
81 2113Ø Ø627Ø6 INF$39: ADD       #18.,SP ;FLUSH STACK
         ØØØØ22
82 21134 Ø127ØØ              MOV      #Ø4Ø511,RØ       ;ANS = SIGN(ARG1)*PI/2
         Ø4Ø511
83 2114Ø Ø127Ø1              MOV      #ØØ7733,R1
         ØØ7733
84 21144 ØØ5775              TST      @2(R5)   ;TEST ARG1
         ØØØØØ2
85 2115Ø ØØ2ØØ2              BGE      INR$39   ;JUMP IF +PI/2
86 21152 Ø627ØØ              ADD      #1ØØØØØ,RØ       ;-PI/2
         1ØØØØØ
87 21156 ØØØ2Ø5 INR$39: RTS       R5       ;RETURN TO USER
88                      ;
89 2116Ø ØØ5Ø46 ATAN:   CLR       -(SP)    ;CLEAR SIGN FLAG
9Ø 21162 ØØ5Ø46              CLR      -(SP)    ;CLEAR ATAN2 BIAS
91 21164 ØØ5Ø46              CLR      -(SP)
92 21166 ØØ5Ø46              CLR      -(SP)    ;CLEAR QUADRANT BIAS
93 2117Ø ØØ5Ø46              CLR      -(SP)
94 21172 Ø165Ø4              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 021027 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
```

```
                     ;
     1536 012666 UP$39:  MOV     (SP)+,10.(SP)      ;MOVE STACK ITEM UP
          000012
     1542 012666         MOV     (SP)+,10.(SP)
          000012
     1546 000134         JMP     @(R4)+
                     ;
     1550 012600 PLY$39: MOV     (SP)+,R0           ;POP POLY ARG
     1552 012601         MOV     (SP)+,R1
     1554 012702         MOV     #CON$39+4,R2       ;POINT TO COEFFICIENT TABLE
          021644'
     1560 012703         MOV     #5,R3   ;LOOP 5
          000005
     1564 000402         BR      PY1$39
     1566 010146 PY2$39: MOV     R1,-(SP)           ;PUSH ARG
     1570 010046         MOV     R0,-(SP)
     1572 014246 PY1$39: MOV     -(R2),-(SP)        ;PUSH CONSTANT
     1574 014246         MOV     -(R2),-(SP)
     1576 005303         DEC     R3      ;COUNT
     1600 003372         BGT     PY2$39
     1602 000134         JMP     @(R4)+
                     ;
     1604 005766 SGN$39: TST     8.(SP)  ;CHECK SIGN FLAG
          000010
     1610 001402         BEQ     SG1$39
     1612 062716         ADD     #100000,@SP        ;NEGATE RESULT FOR (-1,0) & (1,I
          100000
     1616 000134 SG1$39: JMP     @(R4)+
                         .ENDC
                     ;
                         .IFDF   FPU
                 ATAN2:  SETF    ;                  SET FP MODE FOR FPU
                         MOV     2(R5),R3;          ADDRESS OF ARG1
                         MOV     4(R5),R4;          ADDRESS OF ARG2
                         MOV     @R3,R0;            HIGH ORDER ARG1
                         MOV     @R4,R1;            HIGH ORDER ARG2
                         BEQ     INF$39;            JUMP IF DENOMINATOR 0
                         ASL     R0;
                         CLRB    R0;
                         SWAB    R0;                EXPONENT OF ARG1
                         ASL     R1;
                         CLRB    R1;
                         SWAB    R1;                EXPONENT OF ARG2
                         SUB     R1,R0;             GET EXPONENT DIFFERENCE
                         CMP     #26.,R0;                CHECK MAGNITUDE
                         BLT     INF$39;            TREAT AS INFINITE
                         LDF     PI$39,F3;               INITIALIZE BIAS=PI
                         LDF     @R3,F0;            GET ARG1
                         CFCC
                         BGE     A1P$39;            JUMP IF ARG1>0
                         NEGF    F3;                BIAS=SIGN(ARG1)*PI
                 A1P$39: LDF     @R4,F1;            GET ARG2
                         CFCC
                         BLT     A2M$39;
                         CLRF    F3;                IF ARG2>0, BIAS=0
                 A2M$39: DIVF    F1,F0;             ARG1/ARG2, SET FLOAT CC
                         BR      AT1$39;            JOIN MAIN ROUTINE
```

```
                ;
                INF$39: LDF     PI2$39,F1;              RESULT=SIGN(ARG1)*PI/2
                        TST     @R3;            TEST ARG1
                        BGE     EXI$39;         +PI/2
                        NEGF    F1;             -PI/2
                        BR      EXI$39;
                ;
                ATAN:   SETF    ;               SET FP MODE FOR FPU
                        CLRF    F3;             CLEAR ATAN2 BIAS
                        LDF     @2(R5),F0;      GET ARGUMENT
                AT1$39: CLR     R4;             CLEAR SIGN FLAG
                        CFCC    ;               GET SIGN OF ARGUMENT
                        STF     F3,F5;          F5=ATAN2 BIAS
                        CLRF    F3;             CLEAR QUADRANT BIAS
                        BGE     PLUS39;         JUMP IF QUADRANT 1 OR 3
                        ABSF    F0;             ABS(X)
                        INC     R4;             FLAG -
                PLUS39: LDF     #1.0,F1;        1.0
                        CMPF    F0,F1;          CHECK IF X<=1.0
                        CFCC
                        BLE     LE1$39;
                GT1$39: DEC     R4;             X>1.0, ADJUST SIGN FLAG
                        DIVF    F0,F1;          1.0/X
                        LDF     F1,F0;          ATAN(X)=PI/2-ATAN(1/X)
                        LDF     PI2$39,F3;              QUADRANT BIAS=PI/2
                ;
                LE1$39: STF     F3,F4;          F4=QUADRANT BIAS
                        CLRF    F3;             F3=0.0
                        CMPF    T15$39,F0;      COMPARE TAN(15) : X
                        CFCC
                        BGE     L15$39;         X<= TAN(15)
                        LDF     PI6$39,F3;              F3=PI/6
                        LDF     F0,F1;
                        MULF    RT3$39,F0;
                        SUBF    #1.0,F0;        X*ROOT3-1.0
                        ADDF    RT3$39,F1;      X+ROOT3
                        DIVF    F1,F0;          (X*ROOT3-1.0)/(X+ROOT3)
                ;
                L15$39: LDF     F0,F2;          X
                        MULF    F0,F0;          X**2
                        MOV     #FCO$39,R0;     POINTER TO POLYNOMIAL CONSTANTS
                        MOV     #4,R1;          COUNT OF COEFFICIENTS
                        LDF     (R0)+,F1;       INITIALIZE ACCUMULATOR
                XPD$39: MULF    F0,F1;
                        DEC     R1;             COUNT
                        ADDF    (R0)+,F1;       F1:= F1* X**2 + C(I)
                        BGT     XPD$39; LOOP
                        MULF    F2,F1;          F1:= F1*X
                        ADDF    F3,F1;          PI/6 OR 0.0
                        SUBF    F4,F1;          P(X)-QUAD BIAS
                        TST     R4;             TEST SIGN FLAG
                        BEQ     SG1$39;         NO ADJUSTMENT
                        NEGF    F1;             NEGATE RESULT FOR (-1,0)&(1,INF)
                SG1$39: ADDF    F5,F1;          ATAN2 BIAS
                ;
                EXI$39: STF     F1,-(SP);       MOVE RESULT TO STACK
                        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  SSQT03
                                .IFDF   CND$41
                        ;
                        ;       SQRT    V003A
                        ;
                        ; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
                        ;

                                .GLOBL  SQRT,$ERR;
                                .IFNDF  FPU
                                .GLOBL  $ADR,$DVR,$POLSH;
                                .ENDC
                        ;       SQRT    THE REAL SQUARE ROOT FUNCTION
                        ;       CALLING SEQUENCE:
                        ;       JSR     R5,SQRT
                        ;       BR      A
                        ;       #ARG
                        ;A:
                        ;       RETURNS THE SQUARE ROOT IN R0 AND R1.
                        ;
        000000                  R0=%0
        000001                  R1=%1
        000004                  R4=%4
        000005                  R5=%5
        000006                  SP=%6
        000000                  F0=%0
        000001                  F1=%1
        000002                  F2=%2
                                .IFDF   FPU
                        SQRT:   MOV     @2(R5),R1;        GET HIGH ORDER ARGUMENT
                                .ENDC
                                .IFNDF  FPU
 21650  010546          SQRT:   MOV     R5,-(SP)
 21652  016505                  MOV     2(R5),R5          ;GET ARGUMENT ADDRESS
        000002
 21656  011501                  MOV     @R5,R1  ;GET HIGH ORDER ARGUMENT
                                .ENDC
 21660  100443                  BMI     ERR$41  ;ERROR IF ARGUMENT NEGATIVE
 21662  001446                  BEQ     ZER$41  ;FAST EXIT IF ZERO
                                .IFNDF  FPU
 21664  012746                  MOV     #3,-(SP)          ;PUSH ITERATION COUNT
        000003
                                .ENDC
 21670  006201                  ASR     R1      ;FORM INITIAL ESTIMATE
 21672  062701                  ADD     #20100,R1
        020100
 21676  005046                  CLR     -(SP)   ;USE ONLY HIGH ORDER PARTS FIRST
 21700  010146                  MOV     R1,-(SP)          ;'CAUSE ADD AND DIVIDE ARE
                                .IFNDF  FPU
 21702  005046                  CLR     -(SP)   ;FASTER THAT WAY
 21704  011546                  MOV     @R5,-(SP)
 21706  005046                  CLR     -(SP)
 21710  010146                  MOV     R1,-(SP)
 21712  004467          LUP$41: JSR     R4,$POLSH         ;ENTER POLISH MODE
        177726
 21716  013256'                 .WORD   $DVR,$ADR,UPL$41          ;(X/E+E)
 21720  002010'
```

```
21722 021724'
21724 162716 UPL$41: SUB     #200,@SP        ;(X/E+E)/2
      000200
21730 005366         DEC     4(SP)   ;COUNT LOOP
      000004
21734 001410         BEQ     OUT$41
21736 016546         MOV     2(R5),-(SP)     ;USE LOW ORDER PARTS
      000002
21742 011546         MOV     @R5,-(SP)       ;TOO FROM NOW ON
21744 016646         MOV     6(SP),-(SP)
      000006
21750 016646         MOV     6(SP),-(SP)
      000006
21754 000756         BR      LUP$41  ;GO FOR ANOTHER ITERATION
21756 012600 OUT$41: MOV     (SP)+,R0        ;GET RESULT INTO R0,R1
21760 012601         MOV     (SP)+,R1
21762 005726         TST     (SP)+   ;POP ITERATION COUNTER
21764 012605 RTN$41: MOV     (SP)+,R5
21766 000205         RTS     R5      ;RETURN TO CALLER
21770 004567 ERR$41: JSR     R5,$ERR ;ERROR 4,11
      000012
21774 000773         BR      RTN$41
21776    004         .BYTE   4
21777    013         .BYTE   11.
22000 005000 ZER$41: CLR     R0
22002 005001         CLR     R1
22004 000767         BR      RTN$41
                     .ENDC
                     .IFDF   FPU
                     MOV     #3,R0;          ITERATION COUNT
                     SETF    ;               SINGLE PRECISION FP
                     LDF     (SP)+,F0;       GET INITIAL ESTIMATE
                     LDF     @2(R5),F2;      GET X
             ;
             LUP$41: LDF     F0,F1;          E=E'
                     LDF     F2,F0;          X
                     DIVF    F1,F0;          X/E
                     ADDF    F1,F0;          X/E+E
                     DEC     R0;             COUNT
                     DIVF    #2.0,F0;        E'=(X/E+E)/2
                     BGT     LUP$41;
             ;
                     STF     F0,-(SP);               RESULT TO STACK
                     MOV     (SP)+,R0;       AND THENCE TO R0,R1
                     MOV     (SP)+,R1;
                     RTS     R5;             EXIT
             ;
             ERR$41: JSR     R5,$ERR;        ERROR 4,11
                     RTS     R5;             EXIT
                     .BYTE   4
                     .BYTE   11.
             ZER$41: CLR     R0;
                     CLR     R1;
                     RTS     R5;
                     .ENDC
                     .ENDC
```

```
                           .TITLE  $ERR01
                           .GLOBL  $ERR,$ERRA,$ERVEC
       000000              R0 = %0
       000005              R5 = %5
       000006              SP = %6
       000007              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          MOV     2(R5),R0;         GET ERROR CLASS/NUMBER
      000002
22014 000401          BR      ERB$43
22016 010046 $ERRA:   MOV     R0,-(SP);         SAVE R0
22020        ERB$43:  .IFNDF  CLASS5;           DEFINE TO GET WARNINGS
22020 120027          CMPB    R0,#5;            CLASS 5 (WARNING)?
      000005
22024 001402          BEQ     IGN$43;           IGNORE IF SO
                      .ENDC
22026 004777          JSR     PC,@$ERVEC;       CALL USER ERR ROUTINE
      000004
22032 012600 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
```

```
                                .TITLE  $LDR01
                                .IFDF   CND$44&CND$42
          000004                R4=%4
          000006                SP=%6

                        ;       LOAD FLAC - SINGLE PRECISION

   022044 012666 $LDR:          MOV     (SP)+,2(SP);     MOVE OPERAND TO RESULT LOC
          000002
   022050 012666                MOV     (SP)+,2(SP)
          000002
   22054 000134                 JMP     @(R4)+;          POLISH RETURN
                                .ENDC
```

```
                                  .TITLE  $LDD01
                                  .IFDF   CND$45&CND$42
         000000                   R0=%0
         000004                   R4=%4
         000006                   SP=%6

                         ;        LOAD FLAC - DOUBLE PRECISION

 022056  010600  $LDD:            MOV     SP,R0;            COPY STACK POINTER
  22060  062700                   ADD     #8.,R0;           CALC ADDR OF RESULT
         000010
  22064  012620                   MOV     (SP)+,(R0)+;      MOVE OPRAND TO RESULT LOC
  22066  012620                   MOV     (SP)+,(R0)+
  22070  012620                   MOV     (SP)+,(R0)+
  22072  012620                   MOV     (SP)+,(R0)+
  22074  000134                   JMP     @(R4)+;           POLISH RETURN
                                  .ENDC
```

```
                                .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 001413         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     #13,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
```

```
1                               .TITLE  $STD01
2                               .IFDF   CND$47&CND$42
3             000000            R0=%0
4             000001            R1=%1
5             000002            R2=%2
6             000003            R3=%3
7             000004            R4=%4
8             000005            R5=%5
9             000006            SP=%6
10            000007            PC=%7
11
12 22152 012705 $STD1:  MOV     #FAC$42,R5
      000432'
13 22156 005766         TST     34(SP);         TEST FOR STACK MODE
      000034
14 22162 001420         BEQ     STK$47;         BRANCH IF NOT
15 22164 005066         CLR     34(SP)
      000034
16 22170 010600         MOV     SP,R0
17 22172 010601         MOV     SP,R1
18 22174 062701         ADD     #10,R1
      000010
19 22200 012702         MOV     #13,R2
      000013
20 22204 012120 LP$47:  MOV     (R1)+,(R0)+
21 22206 005302         DEC     R2
22 22210 001375         BNE     LP$47
23 22212 012520         MOV     (R5)+,(R0)+;    STORE THE FLAC
24 22214 012520         MOV     (R5)+,(R0)+
25 22216 012520         MOV     (R5)+,(R0)+
26 22220 012520         MOV     (R5)+,(R0)+
27 22222 000134         JMP     @(R4)+;         RETURN
28 22224 012520 STK$47: MOV     (R5)+,(R0)+
29 22226 012520         MOV     (R5)+,(R0)+
30 22230 012520         MOV     (R5)+,(R0)+
31 22232 012520         MOV     (R5)+,(R0)+
32 22234 062706         ADD     #10,SP;         POP OPERAND
      000010
33 22240 000134         JMP     @(R4)+
34                      .ENDC
35                      .TITLE  FPMP11 FLOATING POINT & MATH PACKAGE
36
37            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      021160RG      ATAN2     020766RG       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    016362R        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    021640R       COS       020030RG       COV$30    017506R
CYC$29    017062R       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    012564R        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         DPK$8     004610R
DR1$12    007640R       DR2$12    007666R        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
```

```
FPMP11 FLOATING POINT & MATH PA MACRO VR04-14 07-SEP-72 11:43 PAGE 52+
SYMBOL TABLE

D16$19  014370R         D2     = 000024         D6R$19  014400R
ECK$1   001232R         ECK$2   002212R         ECL$16  012520R
ECL$18  013406R         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    = 000032         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   005526R         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  022032R         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   003060R         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   005264R         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   004762R         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         NGM$35  017750R         NGO$16  013104R
NGO$18  013664R         NML$19  014302R         NNZ$28  016272R
NNZ$8   003644R         NNZ$9   005176R         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
```

```
FPMP11 FLOATING POINT & MATH PA MACRO VRØ4-14 Ø7-SEP-72 11:43 PAGE 52+
SYMBOL TABLE

NTØ$18  Ø13540R                NUMEND= ØØØØØØ               NXT$24  Ø15346R
NXT$8   ØØ3574R                NZE$25  Ø15646R              OCL$24  Ø15506R
OCT$25  Ø16ØØ4R                ONE$19  Ø14Ø6ØR              ONE$2Ø  Ø15Ø7ØR
ONE$38  Ø2Ø666R                UT2$42  ØØØ17ØR              OUT$1   ØØ1552R
OUT$14  Ø1Ø626R                OUT$19  Ø1434ØR              OUT$2   ØØ2426R
OUT$28  Ø16552R                OUT$29  Ø17126R              OUT$3Ø  Ø174Ø4R
OUT$35  Ø17744R                OUT$38  Ø2Ø57ØR              OUT$41  Ø21756R
OUT$5   ØØ3322R                OUT$6   ØØ34ØØR              OVF$1   ØØ1564R
OVR$1   ØØ1566R                OVR$12  ØØ7652R              OVR$16  Ø12764R
OVR$18  Ø13454R                OVR$19  Ø14Ø66R              OVR$2   ØØ2436R
OVR$2Ø  Ø151ØØR                OVR$28  Ø166Ø6R              OVR$29  Ø1715ØR
OVR$3Ø  Ø1743ØR                OVR$31  Ø17554R              OVR$35  Ø17754R
OVR$36  Ø2ØØ2ØR                OVR$8   ØØ467ØR              OV1$12  ØØ7656R
OV1$16  Ø12762R                OV1$18  Ø13452R              OV1$28  Ø166Ø4R
OV1$3Ø  Ø17432R                P     = ØØØØ2Ø               PC    =%ØØØØØ7
PCK$8   ØØ4Ø52R                PLE$2Ø  Ø14732R              PLM$42  ØØØ354R
PLS$42  ØØØ262R                PLS$5   ØØ333ØR              PLS$6   ØØ34Ø6R
PLU$15  Ø11216R                PLU$39  Ø21216R              PLY$13  Ø1Ø3ØØR
PLY$15  Ø12Ø2ØR                PLY$37  Ø2Ø324R              PLY$39  Ø2155ØR
PL1$42  ØØØ372R                PL2$1Ø  ØØ727ØR              PL2$3   ØØ3Ø26R
PMODE = Ø4ØØØØ                 PNT$8   ØØ3766R              POINT = ØØØØØ2
POINTL= ØØØØØ2                 POS$19  Ø13732R              POS$2Ø  Ø14574R
POS$25  Ø15612R                POS$27  Ø16Ø76R              PR4$13  Ø1Ø134R
PTF$8   ØØ4Ø6ØR                PT$42   ØØØØ76R              PY1$13  Ø1Ø322R
PY1$15  Ø12Ø42R                PY1$37  Ø2Ø346R              PY1$39  Ø21572R
PY2$13  Ø1Ø312R                PY2$15  Ø12Ø32R              PY2$37  Ø2Ø342R
PY2$39  Ø21566R                P1$17   Ø13144R              P1$29   Ø17Ø16R
P15$38  Ø2Ø722R                P2$17   Ø13156R              P2$29   Ø17Ø3ØR
P3$17   Ø13236R                P35$38  Ø2Ø734R              P45$38  Ø2Ø712R
Q     = ØØØØ14                 QSE$13  Ø1ØØ6ØR              QSE$37  Ø2Ø15ØR
QSR$13  Ø1Ø26ØR                QSR$37  Ø2Ø3Ø4R              QST$13  Ø1Ø224R
QST$37  Ø2Ø254R                QUD$13  Ø1Ø216R              QUD$37  Ø2Ø246R
QUT$13  Ø1Ø276R                QUT$37  Ø2Ø322R              Q13$13  Ø1Ø264R
Q13$37  Ø2Ø31ØR                RDD$9   ØØ6432R              RDF$9   ØØ64Ø2R
REG$1Ø  ØØ7144R                REG$3   ØØ2742R              REL$25  Ø15616R
RESLT = ØØØØ1Ø                 RESULT= ØØØØ36               RET$21  Ø15146R
RET$42  ØØØ22ØR                RIT$8   ØØ5Ø22R              RIT$9   ØØ6634R
RND$22  Ø15166R                ROR$11  ØØ7564R              ROR$4   ØØ32Ø4R
RRN$8   ØØ4664R                RR1$11  ØØ7542R              RTI$42  ØØØ332R
RTN$13  Ø1Ø14ØR                RTN$14  Ø1Ø64ØR              RTN$16  Ø1275ØR
RTN$18  Ø1362ØR                RTN$37  Ø2Ø2Ø4R              RTN$41  Ø21764R
RTS$16  Ø13126R                RTS$18  Ø137Ø6R              RT1$13  Ø1Ø15ØR
RT1$37  Ø2Ø22ØR                RT2$19  Ø14544R              RT2$42  ØØØ322R
RUD$9   ØØ634ØR                RU1$9   ØØ6476R              RU2$9   ØØ6464R
RU3$9   ØØ65ØØR                RØ    =%ØØØØØØ               R1    =%ØØØØØ1
R2    =%ØØØØØ2                 R3    =%ØØØØØ3               R4    =%ØØØØØ4
R5    =%ØØØØØ5                 S     = ØØØØ26               SCK$1   ØØ12ØØR
SCK$2   ØØ2176R                SCL$1Ø  ØØ7232R              SCL$19  Ø1425ØR
SCL$2Ø  Ø15Ø46R                SCL$3   ØØ3ØØØR              SCL$8   ØØ3722R
SCN$8   ØØ3524R                SC1$19  Ø14312R              SC1$8   ØØ3742R
SFD$1   ØØ1414R                SFD$2   ØØ2336R              SFL$2   ØØ23Ø6R
SFR$1   ØØ1266R                SFR$2   ØØ2236R              SFT$1   ØØ1236R
SFT$2   ØØ2216R                SFT$35  Ø1772ØR              SFØ$2   ØØ2326R
SF1$1   ØØ13Ø4R                SGN$15  Ø12Ø64R              SGN$16  Ø127Ø2R
SGN$18  Ø13572R                SGN$24  Ø15444R              SGN$35  Ø17736R
SGN$39  Ø216Ø4R                SGS$24  Ø15314R              SGS$8   ØØ3552R
SG1$15  Ø12Ø76R                SG1$39  Ø21616R              SHF$11  ØØ7522R
```

```
SHF$4   003202R         SIGN   = 000002         SIGNS  = 000000
SIN     020064RG        SINGLE= 000001          SL8$1    001346R
SME$5   003266R         SME$6    003364R        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$16  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   016614R
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   016760R
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

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

FPMP-11 User's Manual
DEC-11-NFPMA-A-D

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

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

How can this manual be improved?

Other comments?

Please state your position. ______________________ Date: __________

Name: ______________________ Organization: ______________________

Street: ______________________ Department: ______________________

City: ______________ State: ______________ Zip or Country ____________

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

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754