214
1-15-73
ANCR-1040

2420
UC-32

FP8
FLOATING POINT ARITHMETIC SOFTWARE FOR
DEC PDP-8 SERIES COMPUTERS

W.R. Myers

Aerojet Nuclear Company

NATIONAL REACTOR TESTING STATION

Idaho Falls, Idaho — 83401

MASTER

DATE PUBLISHED—DECEMBER 1972

PREPARED FOR THE
U. S. ATOMIC ENERGY COMMISSION

IDAHO OPERATIONS OFFICE UNDER CONTRACT AT(10-1)-1375

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.95

LEGAL NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

ANCR-1040

Mathematics and Computers
TID-4500

FP8
FLOATING POINT ARITHMETIC SOFTWARE
FOR DEC PDP-8 SERIES COMPUTERS

W. R. Myers

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

AEROJET NUCLEAR COMPANY

Date Published -- December 1972

PREPARED FOR THE U. S. ATOMIC ENERGY COMMISSION
IDAHO OPERATIONS OFFICE
UNDER CONTRACT NO. AT(10-1)-1375

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

ACKNOWLEDGMENTS

The author is grateful to Mssrs. M. S. Cole and E. W. Killian for many valuable comments and suggestions.

ABSTRACT

FP8 is a floating point arithmetic interpretive program for use in any DEC PDP-8 series computer. It is somewhat smaller and much more versatile than the standard arithmetic package supplied by DEC. The full program requires 1249 locations in one memory field (plus from five to 34 locations in remote fields, for linkage) compared to the 1408 locations required by the DEC software. FP8 implements access across memory field boundaries for both instruction sequences and operand reference. A four-bit operation code is used to obtain an expanded set of floating-mode instructions including: add, subtract, multiply, divide, inverse divide, load, store, three-way compare, and jump and jump-to-subroutine. Operate-class instructions include: immediate load, absolute value, clear, change sign, set data field, and set output format. FP8 includes single- and double-precision fix and float operations, and square, square root, exponential, logarithm, sine, cosine, and arctangent functions, and free-format input and variable format output routines. The program size may be reduced to 663 locations by deletion of the function and input/output routines.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGMENTS	ii
ABSTRACT	iii
I. INTRODUCTION	1
1. FP8 Memory Requirements	1
2. Initiating and Terminating Interpretive Mode	2
3. Interpretive Mode Addressing	2
4. Floating Point Number Format	3
II. INTERPRETIVE MODE INSTRUCTIONS	4
1. Control Instructions	4
2. Register Modification	5
3. Jump Instructions	5
4. Operand Reference	5
5. Functional Instructions	6
6. Input/Output Instructions	7
7. User-Defined Op-Code	8
III. INPUT AND OUTPUT	10
1. The Input Interpreter	10
2. User Character Input Routine	11
3. The Output Controller	12
4. Format Control	12
5. User Character Output Routine	13
APPENDICES	
A. Glossary of Special Memory Locations	A-1
B. Integer Multiply and Divide Routines	B-1
C. FP8 Program Source Listing	C-1

I. INTRODUCTION

FP8 is a system of arithmetic and logical functions which interprets special PDP-8 code to perform a variety of floating-point arithmetic operations. The program will operate in any of the PDP-8 (and PDP-12) series of computers with 2k words (or more) of memory. FP8 has been designed for small size while implementing a versatile instruction set and utilization of extended memory for both operands and instruction sequences. The complete program requires 1249₁₀ words of memory to provide 32₁₀ interpretive-mode instructions including: inverse divide, three-way branch, six functions, free-format input and variable-format output, and single- and double-precision fix and float, as well as the usual arithmetic operations.

1. FP8 MEMORY REQUIREMENTS

The floating point interpretive package should be assembled to span locations 151₈ through 2511 of any PDP-8 memory field, although it is essentially page-relocatable beginning at location 200 provided that the proper linkage is made between the entry point at 176 and the new origin. All discussions and descriptions in this report assume that the normal assembly is made. If the functional operations (exp, log, sqrt, sin, cos, and atn) are not required, the first free location is 2041₈; if the input/output is also deleted, 1400₈ is the first free location.

<u>FP8 FUNCTIONS</u>	<u>SIZE</u>	
Arithmetic	1227 ₈	663 ₁₀
Arithmetic and I/O	1670	952
Arithmetic, I/O and Functions	2341	1249

In order to access FP8 from other memory fields, from five to 34 locations are required for linkage, depending on what facilities are to be used (illustrated in Appendix C, listing page 2). Basic access to the interpreter requires the sequence shown in the appendix, or the sequence:

.LOC	176	/ AT 176 TO CONFORM TO
HLT		/ DEFINITION OF EFM.
CIF+10		
JMS I	+.2	
JMP I	176	/ FP8 ASSUMED IN FIELD "1"
EME		

An additional seven instructions are required for linkage to the single-precision float routine, and another 20 to link to the double-precision float routine and provide for loading and reading the high order floating accumulator.

The user must also supply character-handling routines appropriate to his IO devices for use by the FP8 input and output controllers.

2. INITIATING AND TERMINATING INTERPRETIVE MODE

FP8 is a subroutine that operates on a variable length argument list; the arguments are the interpretive mode instructions. Interpretive mode is initiated by entering the subroutine (by a JMS) and is terminated by a special entry (0000) in the argument list. There are three entries to FP8, directly by executing JMS 176, or through the "single-signed float and enter" instruction JMS I 165, or the "double-precision float and enter" instruction JMS I 164. Provided that the linkage conventions displayed in Appendix C (listing, page 2) are followed, these operations are valid in any PDP-8 memory field.

Interpretive mode is terminated by executing one of the interpretive mode instructions LFM, SFIX, or DFIX. These instructions implement returns to normal mode with the AC clear, or containing the "FIX" (rounded) of the number in the floating accumulator.

3. INTERPRETIVE MODE ADDRESSING

In order to provide an expanded instruction set for floating point operations, bit three of the instruction word, used to indicate indirect addressing in normal PDP-8 coding, is utilized for part of the operation code by FP8. As a consequence, all FP8 memory references are implicitly indirect, and address pointers are required even when both operand and instruction are in the same PDP-8 page; there is no penalty, compared to normal mode coding, when two different pages are involved. The page indicator, bit four, is implemented exactly as in normal mode coding:

FP8 includes an (interpretive mode) instruction to specify the memory field for all data reference instructions (all memory reference instructions except JP, JS). Operands are determined by a pointer, in the current instruction memory field, containing the address of an operand in any other (or the same) memory field. Thus the interpreter, the interpreted coding, and the floating point data may be in the same, or in three different memory fields.

4. FLOATING POINT NUMBER FORMAT

FP8 represents floating point numbers in a three-word format; operands are addressed by the first of the three words. The first word is the binary exponent in two's-complement form, the second and third words contain a positive, normalized binary fraction in bits 1-11 of word two, continued in bits 0-11 of word three. The high-order bit of the second word contains the sign of the number; 0 for positive numbers, 1 for negative numbers:

TAG	Exponent		two's exponent
	S	High Mantissa	sign and high mantissa
	Low Mantissa		low order mantissa

The format for the Floating Accumulator (FAC) and the floating operand register is similar; the only difference is that the sign is separated from the mantissa and held in a separate register:

S		ACS or BCS
	Exponent	ACX or BCX
0	High Mantissa	ACH or BCH
	Low Mantissa	ACL or BCL

Although FP8 normalizes the FAC after each arithmetic operation, so that any data returned to memory by the store instruction (ST) are normalized, numbers loaded to the FAC by the load instruction (LD) are not tested for normalization; all assembled constants should be normalized.

This number format supports a precision greater than six (closer to seven) decimal digits and a magnitude range of 1200 decades. The arithmetic operations are accurate to 23 bits; results are truncated rather than rounded. Results of the "FIX" operations (SFIX, DFIX) are rounded to the nearest integer. The functional operations are generally accurate to 22 or 23 bits with occasional degradation to 21 bits. FP8 will thus reliably support six-decimal-digit computation in any normal application.

II. INTERPRETIVE MODE INSTRUCTIONS

There are 32 standard instructions available to the user in interpretive (floating) mode. The mnemonics used in the description are defined (by direct definition) as part of the FP8 coding. Many of the mnemonics are specifically dependent on the assembly of FP8 beginning at memory address 151.

1. CONTROL INSTRUCTIONS

<u>EFM</u>	4176	<u>Enter Floating (Interpretive) Mode</u>	JMS 176	Enter the interpreter subroutine to invoke floating point operating mode. Interpretive (floating) mode will remain in effect until one of the instructions LFM, SFIX, or DFIX is encountered. The floating accumulator is not affected.
<u>SFLO</u>	4565	<u>Single (Signed) Float and Enter</u>	JMS I 165	Convert the contents of the AC (a twelve-bit two's-complement integer) to a floating-format number in the FAC and enter interpretive mode.
<u>DFLO</u>	4564	<u>Double-Precision Float and Enter</u>	JMS I 164	Convert the 24-bit positive number in ACH (high order part) and the AC (low-order 12 bits) to a floating point number in the FAC and enter interpretive mode.
<u>FCDF N</u>	606N	<u>Change Floating Mode Data Field</u>		For all interpretive-mode operand-reference instructions, until another FCDF instruction is encountered, all operands will be in PDP-8 memory field N. Operand address pointers are in the instruction field.
<u>LFM</u>	0000	<u>Leave Floating (Interpretive) Mode</u>		Terminate Interpretive Mode coding, leave zero in the AC. The Floating Accumulator (FAC) is not affected.
<u>SFIX</u>	6040	<u>Single (Signed) Fix and Leave</u>		Convert the number in the FAC to a twelve-bit two's-complement signed integer, leave the result in the AC, and leave interpretive mode. The result is rounded to the nearest integer. If the result is greater than or equal 2^{24} or less than $1/2$, zero is returned. If the result is in the range $2^{12} \leq N < 2^{24}$, the low order 12 bits of the "double fix" are returned, two's-complemented if the FAC was negative. The content of the FAC is lost.

DFIX 6030 Double-Precision Fix and Leave Interpreter
Convert the number in the FAC to a 24-bit (magnitude, rounded) integer in ACH (high-order portion) and the AC (low twelve bits). The original content of the FAC is lost, except that the sign remains in ACS. If the result is greater than or equal 2^{24} , or less than $1/2$, zero is returned. Program execution resumes in normal mode.

2. REGISTER MODIFICATION

ILA N ONNN Load N to Floating Accumulator
Immediately load the FAC with the positive number contained in the low-order eight bits of the ILA instruction. Numbers in the range $1 \leq N \leq 377_8$ may be generated directly in the FAC. Note that ILA 0 is the LFM instruction, and not legitimate.

ZAC 6000 Zero the Floating Accumulator
Load zero to the FAC.

ABS 6010 Absolute Value to the Floating Accumulator
Set the sign of the FAC positive, leave the magnitude unchanged.

CHS 6020 Change the Floating Accumulator Sign
Change the sign of the FAC from positive to negative, or from negative to positive, leave the magnitude unchanged.

3. JUMP INSTRUCTIONS

JP L 5400 Jump to Location
JMP I L
Load the address indicated by L to the interpreter instruction location register. Neither the FAC nor the interpretive data field is modified. Jumps are restricted to the current interpretive instruction memory field.

JS L 4400 Jump to Subroutine
JMS I L
Store the current instruction location register, incremented by one, in the location indicated by L, then jump to the following location. The FAC is not modified; the interpretive mode data field is not modified.

4. OPERAND REFERENCE

LD X 1000 Load Operand to the Floating Accumulator
The three-word floating-point number starting in the location indicated by X is loaded to the floating

accumulator (FAC). The operand is assumed properly normalized; the FP8 interpreter does not check for operand normalization in any "fetch" sequence.

- ST X 1400 Store the Floating Accumulator
The number in the FAC is stored in the three words beginning with the address indicated by X.
- AD X 2000 Add to the Floating Accumulator
The operand indicated by X is added to the number in the FAC. The result replaces the content of the FAC. The FAC is normalized after every arithmetic operation.
- SB X 2400 Subtract from the Floating Accumulator
The operand indicated by X is subtracted from the FAC; the difference is normalized and retained in the FAC.
- MP X 3000 Multiply the Floating Accumulator
The product of the number in the FAC and the operand indicated by X replaces the content of the FAC.
- DV X 3400 Divide the Floating Accumulator
The number in the FAC is divided by the operand indicated by X; the quotient replaces the content of the FAC. If either the operand or the number in the FAC is zero, the result is set to zero.
- RDV X 4000 Divide by the Floating Accumulator
The operand indicated by X is divided by the number in the FAC, the quotient replaces the content of the FAC. If either of the operands is zero, the result is zero.
- CMP X 5000 Compare to the Floating Accumulator
The operand is algebraically compared to the number in the FAC and a conditional branch to one of the following three locations is made depending on whether the operand is less than, equal to, or greater than the number in the FAC:

go to L+1; FAC < X
 L+2; FAC = X
 L+3; FAC > X

Neither the operand nor the FAC is modified.

5. FUNCTIONAL INSTRUCTIONS

The function algorithms used by FP8 have been collected from many sources, selected for brevity and relative accuracy rather than for speed in execution. The square is a simple multiplication, the square root is computed by a four-fold Newton iteration with an initial estimate equal to unity. The exponent, sine, and cosine are computed

by direct series expansion.* The logarithm and arctangent algorithms have been adapted from those used in the PDP-4/7/9 Basic Arithmetic Package "DISC" (DEC-9B-SFPA-PA).

<u>FSQR</u>	6070	<u>Square the Floating Accumulator</u> The square of the number in the FAC is computed and left in the FAC.
<u>FSQRT</u>	6100	<u>Square Root of the Floating Accumulator</u> The square root of the absolute value of the number in the floating accumulator replaces the number in the floating accumulator.
<u>FEXP</u>	6110	<u>Natural Exponent</u> The natural exponent of the number in the FAC replaces the content of the FAC.
<u>FLOG</u>	6120	<u>Natural Logarithm</u> The natural logarithm of the absolute value of the number in the FAC replaces the content of the FAC. Zero is returned for a zero argument.
<u>FSIN</u>	6130	<u>Sine of the Floating Accumulator (Radians)</u> The sine of the number in the FAC, assumed to be in radians, replaces the content of the FAC. Accuracy deteriorates as the argument increases beyond 2π .
<u>FCOS</u>	6140	<u>Cosine of the Floating Accumulator (Radians)</u> The cosine of the number in the FAC, assumed to be in radians, replaces the content of the FAC. Accuracy deteriorates as the argument increases beyond 2π .
<u>FATN</u>	6150	<u>Arctangent of the Floating Accumulator</u> The principal value arctangent (in radians) of the number in the FAC replaces the content of the FAC.

6. INPUT/OUTPUT INSTRUCTIONS

The input/output instructions are described in detail in the section devoted to use of the FP8 I/O capabilities, and are listed here for the sake of completeness only.

<u>FREAD</u>	6050	<u>Read to the Floating Accumulator</u>
<u>WRITE</u>	6051	<u>Write from the Floating Accumulator</u>
<u>FORMAT</u>	6400	<u>Set Output Format for "WRITE"</u>
<u>FCRLF</u>	6052	<u>Write CR/LF to Output Device</u>

* The algorithms are described in USAEC Report ANCR-1039, "FP9, Floating Point Arithmetic for DEC PDP-9 and PDP-15 Computers", W. R. Myers, March 1972.

7. USER-DEFINED OP-CODE

FP8 has one unimplemented operation code (7000) that may be utilized for special purpose interpretive mode operations. The user may write a code sequence that contains both normal mode and certain interpretive mode (detailed below) operations, insert the starting address of the sequence in the FP8 branch table, and then execute that code as though it were part of the basic FP8 package. Memory locations of interest to the user who would implement the free operation code are:

<u>LOCATION</u>	<u>NAME</u>	
451	--	Location in the FP8 branch table in which the starting address of the special coding must be inserted. The special code must be in the same memory field as the interpreter.
160	FMQ	Contains the interpretive mode instruction word (in this case, 111 0XX XXX XXX) when control branches to the special code sequence.
401	FDF2	Contains the interpretive mode data field set by "FCDF+N" in the form $CDF+10*N = 62N1$.
206	XIF	Contains the interpretive mode instruction field (from which the instruction in FMQ was fetched) in the form $CIF+10*M = 62M2$.
220	XDF	Same as XIF, but in the form $CDF+10*M = 62M1$.
247 250 251	ADR	Contains the address of the operand as specified in the low eight bits of the operation code; this data may be meaningless, depending on the intent of the address field. Pointers to all three words of the operand are developed in 247-251.
172 173 174 175	BCS BCX BCH BCL	Contains the floating point operand when the branch is made to the user special coding; this data may be meaningless depending on the intent of the instruction address field.
153	ACZ	A page-zero link to a subroutine (ACZA) that tests the FAC for zero: JMS I ACZ returns to call+1 if the FAC is zero, to call+2 if the FAC is not zero.
1240	FXQ	The entry point to a subroutine that FIXes the contents of the FAC, by truncation, to ACH, ACL. The high-order fraction bit is returned in the LINK.

The user function must be in the interpreter memory field; it must not use any interpretive instructions in the function or input/output

class, or the FCDF instruction. Any normal-mode operations on the FAC or operand registers are allowed, as well as any of the basic FP8 operations (LD, ST, AD, etc., jumps and compare) provided that the operands are in the interpreter memory field. If interpretive operations are not used, the special function is terminated by JMP 177. If interpretive coding is used, interpretive mode is entered by JMS I SVPC which saves the current interpretive instruction location register and data field and then enters interpretive mode with the data field set to the interpreter field. After one execution of SVPC, the EFM and LFM instructions are used to switch modes. The sequence must now be terminated by JMP I RSPC (in either mode) to restore the instruction location and data field values and return to the normal interpretive procedures. The "SQUARE" and "SQUARE ROOT" routines (Appendix C, listing pages 22 and 25) and the "FORMAT" coding FM1 (listing page 22) should provide adequate examples for most purposes.

III. INPUT AND OUTPUT

FP8 includes a free-format input interpreter which converts ASCII character strings to floating point numbers in the floating accumulator and a variable-format output processor to convert a number in the floating accumulator to an ASCII character sequence specified by a format control word. The FP8 input and output routines are linked to devices by user-written character-handling subroutines to provide capability for device-independent or multiple-device IO.

The operation codes associated with input/output are:

<u>FREAD</u>	6050	<u>Read to the Floating Accumulator</u>	An ASCII character string consisting of numeric characters obtained by calling (JMS) a user-written routine is processed, until a non-numeric character is encountered. The input string is converted to a floating-point number and left in the FAC.
<u>WRITE</u>	6051	<u>Write from the Floating Accumulator</u>	The number in the FAC is converted to an ASCII character string, controlled by the contents of memory location FMT; each time a character is ready, a user-written subroutine is called (JMS) with the character in the AC. At the completion of WRITE, the FAC is restored to the absolute value of the initial content.
<u>FORMAT</u>	6400	<u>Set Output Format for Write</u>	The low eight bits of this instruction are moved to memory location FMT to control output format. Bits 4-8 and bits 9-11 are moved to bits 1-5 and bits 9-11 of FMT (with zeroes to the remaining bits) to control the "integer" field width and the "fraction" field width, respectively. The FAC is not modified.
<u>FCRLF</u>	6052	<u>Write CR/LF to Output Device</u>	This operation moves the codes 215 ₈ and 212 ₈ to the user character output routine to facilitate line formatting while in interpretive mode. The FAC is not modified.

1. THE INPUT INTERPRETER

When the interpretive-mode instruction "FREAD" is encountered, FP8 calls a user-written routine which must return a seven- or eight-level ASCII character in the accumulator; the character is masked to seven bits, stored in memory location ICHR, then tested for numeric or non-numeric. If the character is a numeric character (see table below), it is processed into the number currently being developed and the user routine is again called. Any non-numeric character acts as a terminator, causing the current number to be fully developed and left in the floating accumulator; program execution proceeds at the instruction following

the "FREAD" instruction. The terminator character will be in memory location ICHR, and the number of digits (a sub-set of the numeric characters) encountered will be in memory location CHRX. The user may leave interpretive mode at this point and examine ICHR if the terminator code has significance, and CHRX to establish that legitimate (non-trivial) input was processed.

The input character string has the form "SXXX.YYESZZT", where "S" is a sign (+ or -), "." is the decimal point or period, "E" is the character, and "T" is a non-numeric terminator character; "X", "Y", and "Z" are decimal digits. Any of the above fields may be deleted so that input may have any of the forms:

1., 1, 2.5, .3, -2.3, -1E10, -1E-10, 3.67E6

The only limitation is that the mantissa (XXXYY) should not contain more than seven significant digits; it must be smaller in magnitude than 2^{24} ; the exponent (ZZ) must be smaller in magnitude than 2048.

All characters except those listed here as numeric characters act as terminators; numeric characters are:

0-9	60-71	decimal digits
E	105	exponent indicator (10)
-	55	negation (mantissa or exponent)
.	56	decimal point
+	53	(ignored) plus sign
(LF)	12	(ignored) line feed
null	0	(ignored) leader, null
(blank)	40	leading ignored, trailing terminates
(RO)	177	rub-out, erase

The two special numeric characters are the "blank" and "rub-out". Leading blanks, encountered before any digits have been read, are ignored, but a blank encountered subsequent to a digit acts as a terminator. If a rub-out (177) is read, the input interpreter is re-initialized, essentially erasing all characters processed up to that point. FP8 echoes the character "@" (100) to the user output routine on encountering the rub-out; if this echo is not desired, the user may insert 7610₈ in memory location RBO (1452₈).

2. USER CHARACTER INPUT ROUTINE

The input interpreter executes JMS I 163 each time it is ready for a character. The user of FP8 must write a subroutine appropriate to his input device and provide the proper linkage from memory location INR (163). An example of the required linkage is shown in Appendix C (listing pages 4 and 29). The input routine may contain no interpretive mode instructions, but otherwise may be as complex or as simple as required to return a seven- or eight-level ASCII character code in the accumulator.

3. THE OUTPUT CONTROLLER

The output controller, initiated by the interpretive instruction "WRITE", converts the number in the floating accumulator to a series of ASCII characters, controlled by the format word FMT, calling a user-written character-handling routine each time a character has been developed. Program execution continues at the instruction following the "WRITE" instruction when the format has been satisfied. Output characters are passed as eight-level (200+Code) ASCII codes in the accumulator (except for the rub-out echo, see above, which is 7700₈).

4. FORMAT CONTROL

Output format is controlled by the contents of memory location FMT (162) which may be loaded in normal mode or using the interpretive instruction "FORMAT". Both "F" and "I" (Fortran notation) formats may be indicated. The format control word has the form "AABB", where "AA" indicates the number of characters to be written to the left of the decimal point, i.e., the "integer" part of the number, and "BB" indicates the number of characters, including the decimal point, to be written to the right of the "integer" part:

$$\underbrace{\text{XXX.YYY}}_{\text{AA} \quad \text{BB}}$$

AA+BB is the total field width. AA must be large enough for all digits in the "integer" part of the number, for the sign (blank if positive, "-" if negative), and for any desired inter-number spacing. If BB = 0, only the "integer" part of the number is written, effecting an "I" format; if BB = 1, the decimal point is written, but none of the "fraction" digits are transmitted.

<u>"FORTRAN" FORMATS</u>	<u>FP8</u>
1F10.2	0703
1F15.6	1007
1I3	0300
1I10	1200

If the number to be written is too large for the specified field, the field is filled with "periods" (256) to indicate the error while assuring format continuity in multiple-number lines.

The interpretive instruction "FORMAT" facilitates format control without returning to normal mode to set FMT. The low eight bits of the instruction word are moved to the format control word:

The interpretive instruction "FCRLF" is also provided for interpretive formatting convenience. This instruction sends the ASCII chracter sequence "215, 212" to the user output routine to facilitate multiple line output.

5. USER CHARACTER OUTPUT ROUTINE

Each time the output controller has developed a character, it executes JMS I 161 to transmit the character, in the accumulator, to a user-written subroutine appropriate to the output device. The user must provide the linkage from memory location OTR (161) to his output routine; an example of the required linkage is shown in Appendix C (listing pages 4 and 29). The output routine may contain no interpretive instructions and it must return with a clear accumulator; it may otherwise be as simple or as complex as required.

APPENDIX A

GLOSSARY OF SPECIAL MEMORY LOCATIONS

<u>Symbolic Tag</u>	<u>Location</u>	<u>Usage</u>
CHRX	156	Contains a count of the number of input digits immediately* after each read (FREAD) operation.
ICHR	157	Contains the last character processed in an input (FREAD) operation, immediately after that operation.
FMQ	160	Contains the high-order product for integer multiply. The high-order dividend and the remainder for integer divide (see MPY, DIV).
OTR	161	Contains the address of the user-written character output subroutine.
FMT	162	Contains the output format control word.
INR	163	Contains the address of the user-written character input subroutines.
ACS	166	Floating Accumulator Sign.
ACS	167	FAC exponent.
ACH	170	FAC high-order mantissa, also contains the high-order portion of the result of the double-precision fix (DFIX) operation, and the high-order portion of the double-precision quantity for (DFLO) double float.
ACL	171	FAC low-order mantissa.

* Some of these locations are used for scratch variables by FP8 and will not always contain meaningful data.

<u>Symbolic Tag</u>	<u>Location</u>	<u>Usage</u>
BCS	172	Floating Operand Register, format same as ACS ... ACL.
BCX	173	
BCH	174	
BCL	175	
E	176	Interpreter entry point for access from local memory field (field occupied by FP8).
EME	226	Interpreter entry point for access from remote memory fields.
---	451	May contain an address for a user-written floating point operation utilizing operation code '7000'.
DIV	677	Entry point for integer divide subroutine.
MPY	731	Entry point for integer multiply subroutine.
---	2512	First free core location above FP8 interpreter package (exclusive of user written input and output linkage).

APPENDIX B

INTEGER MULTIPLY AND DIVIDE ROUTINES

FP8 is written to operate in any PDP-8 series computer and contains subroutines for integer multiply and divide that may be accessed by the user from the interpreter memory field. The calling sequences for these subroutines are:

JMS I	(MPY)	JMS I	(DIV)
ADRS		ADRS	

where "ADRS" is the address of the multiplier or the divisor. The multiply routine computes the unsigned 24-bit product of the number in the AC and the indicated multiplier. The high-order twelve bits of the product are left in memory location FMQ, the low-order twelve bits in the AC. The divide subroutine divides the 24-bit number contained in FMQ (high 12 bits) and the AC (low 12 bits) by the indicated divisor; the quotient is returned in the AC, and the remainder in FMQ. The divide routine does no error checking; the user must insure that he will not have divide overflow.

FP8 includes coding (Appendix C, listing page 30) to utilize EAE multiply and divide if available. If EAE is not available, this coding must be removed since it overlays the standard multiply and divide subroutines.

APPENDIX C

FP8 PROGRAM SOURCE LISTING*

Instruction Summary and Definition	1
Cross-Bank Access Coding Example	2
FP8 Source Language Listing	3
Linkage Coding for Input/Output	29
Basic Coding for Special Function	29
Multiply and Divide Overlays for EAE	30

* The user may find it necessary to edit the source language text for compatibility with his assembly environment.

** The user must define the instruction "UDF" (default is CDF+10) to correspond to the interpreter memory field to be used.

```

/      INSTRUCTION SUMMARY AND DEFINITION
/
/ MNEMONIC   CODE   CLASS   DESCRIPTION
/
/   EFM      4176   CONTROL   ENTER INTERPRETIVE MODE.
/   SFLO     4565   CONTROL   SINGLE, SIGNED FLOAT AND EFM.
/   DFLO     4564   CONTROL   DOUBLE PRECISION FLOAT AND EFM.
/   FCDF     6060   CONTROL   CHANGE INTERPRETIVE DATA FIELD.
/   LFM      0000   CONTROL   LEAVE INTERPRETIVE MODE.
/   SFIX     6040   CONTROL   SINGLE FIX AND LFM.
/   DFIX     6030   CONTROL   DOUBLE FIX AND LFM.
/   LD       1000   MEM. REF.   LOAD FAC FROM MEMORY.
/   ST       1400   MEM. REF.   STORE FAC TO MEMORY.
/   AD       2000   MEM. REF.   ADD OPERAND TO FAC.
/   SB       2400   MEM. REF.   SUBTRACT OPERAND FROM FAC.
/   MP       3000   MEM. REF.   MULTIPLY FAC BY OPERAND.
/   DV       3400   MEM. REF.   DIVIDE FAC BY OPERAND.
/   RDV      4000   MEM. REF.   DIVIDE OPERAND BY FAC TO FAC.
/   JS       4400   MEM. REF.   JUMP TO SUBROUTINE
/   CMP      5000   MEM. REF.   COMPARE FAC, CONDITIONAL BRANCH.
/   JP       5400   MEM. REF.   JUMP TO NEW LOCATION.
/   ILA      0000   OPERATE    IMMEDIATE LOAD TO FAC.
/   ZAC      6000   OPERATE    SET FAC TO ZERO.
/   ABS      6010   OPERATE    ABSOLUTE VALUE TO FAC.
/   CHS      6020   OPERATE    CHANGE SIGN OF FAC.
/   FREAD    6050   OPERATE    FREE-FORMAT READ TO FAC.
/   WRITE    6051   OPERATE    FORMATTED OUTPUT FROM FAC.
/   FCRLF    6052   OPERATE    NEW LINE TO OUTPUT DEVICE.
/   FORMAT   6400   OPERATE    SET OUTPUT FORMAT.
/   FSQR     6070   FUNCTION   SQUARE OF FAC TO FAC.
/   FSQRT    6100   FUNCTION   SQUARE ROOT OF FAC.
/   FEXP     6110   FUNCTION   NATURAL EXPONENTIAL.
/   FLOG     6120   FUNCTION   NATURAL LOGARITHM.
/   FSIN     6130   FUNCTION   SINE OF FAC (RADIANs).
/   FCOS     6140   FUNCTION   COSINE OF FAC.
/   FATN     6150   FUNCTION   ARCTANGENT.
/
$                                     END DEFINITIONS

```

	FIELD	0	/ EXAMPLE ACCESS FROM '0'
	.LOC	164	
	XDFLO		/ DOUBLE FLOAT LINKAGE.
	XSFLO		/ SINGLE FLOAT LINKAGE.
	XRHO		/ READ HIGH FAC LINKAGE.
	XLHO		/ LOAD HIGH FAC LINKAGE.
	.LOC	175	
	XEE		/ ENTER INTERPRETIVE
	HLT		/ MODE LINKAGE.
	JMP I	.-2	
	.LOC	7400	
XEE	CIF+10		/ ASSUME INTERPRETER
	JMS I	.*2	/ IN MEMORY FIELD #1.
	JMP I	176	
	EME		
XDFLO	HLT		/ DOUBLE FLOAT AND ENTER
	CIF+10		/ FOR CROSS-BANK ACCESS.
	JMS I	.*3	
	EFM		
	JP	XDFLO	
	AFLO		
XSFLO	HLT		/ CROSS-BANK SINGLE, SIGNED
	CIF+10		/ FLOAT AND ENTER INTERPRETER.
	JMS I	.*3	
	EFM		
	JP	XSFLO	
	BFLO		
EXRHA=4566			/ DEFINE READ HIGH FAC.
XRHO	HLT		/ READ HIGH FAC IN FIELD '1'
	CDF+10		/ FROM NORMAL MODE CODE
	TAD I	.*3	/ IN FIELD '0'.
	CDF		
	JMP I	XRHO	
	ACH		
EXLHA=4567			/ DEFINE LOAD HIGH FAC.
XLHO	HLT		/ LOAD HIGH FAC,
	CDF+10		/ CROSS-BANKS '1' AND '0'.
	DCA I	.-3	
	CDF		
	JMP I	XLHO	
\$		END FIELD '0'	

```

/          FP 8 EM
/          PDP-8 FLOATING POINT
/          EXTENDED MEMORY VERSION
/          ANCR 1040
/          W R MYERS
/          FEBRUARY 1972
/ FLOATING MODE INSTRUCTIONS:
UDF=6211          / FP IN FIELD #1
EFM=4176          / ENTER INTERPRETER
LFM=0             / LEAVE FLOATING MODE
ILA=0            / IMMEDIATE LOAD FAC
LD=1000          / LOAD FAC
ST=1400          / STORE FAC
AD=2000          / ADD
SB=2400          / SUBTRACT
MP=3000          / MULTIPLY
DV=3400          / DIVIDE
RDV=4000         / DIVIDE INVERTED
JS=4400          / JUMP TO SUBROUTINE
CMP=5000         / THREE-WAY COMPARE
JP=5400          / JUMP TO LOCATION
SFLO=4565        / SINGLE FLOAT AND EFM
DFLO=4564        / DOUBLE FLOAT AND EFM
ZAC=6000         / ZERO TO FAC
ABS=6010         / ABSOLUTE VALUE
CHS=6020         / CHANGE SIGN
DFIX=6030        / DOUBLE FIX AND LFM
SFIX=6040        / SIGNED FIX AND LFM
FREAD=6050       / INPUT TO FAC
WRITE=6051       / OUTPUT FROM FAC
FCRLF=6052       / NEW OUTPUT LINE
FORMAT=6400      / SET OUTPUT FORMAT
FCDF=6060        / SET FLOATING DATA FIELD
FSQR=6070        / SQUARE FAC
FSQRT=6100       / SQUARE ROOT
FEXP=6110        / EXPONENT
FLOG=6120        / LOGARITHM
FSIN=6130        / SINE
FCOS=6140        / COSINE
FATN=6150        / ARCTANGENT
/ START OF ASSEMBLED CODE
FIELD          1          / EXAMPLE FP 8 IN FIELD '1'
.LOC          151
SVPC          SPC1
RSPC          RPC1
ACZ           ACZA
BCZ           BCZA
NOR           NORA
CHRX          0          / INPUT CHARACTER COUNTER.
ICHR          0          / LAST INPUT CHARACTER.

```

FMQ	0		/ 'MQ' REGISTER.
OTR	UOR		/ OUTPUT ROUTINE.
FMT	1305		/ FORMAT INDICATOR.
INR	UIR		/ INPUT ROUTINE.
	FLOR		/ DOUBLE FLOAT AND ENTER.
	FLOS		/ SINGLE SIGNED FLOAT.
ACS	0		/ FLOATING ACCUMULATOR.
ACX	0		
ACH	0		
ACL	0		
BCS	0		/ FLOATING OPERAND.
BCX	0		
BCH	0		
BCL	0		
E	0		/ INTERPRETER LOCAL ENTRY.
R	CLA CLL		/ INTERPRETIVE RETURN.
	JMS	TENT	
	NOP		
	TAD I	EX	
	SZA		/ TEST FOR LFM.
	JMP	EGO	
	JMS	TENT	/ RESTORE INSTRUCTION
XIF	CIF		/ AND DATA FIELDS
	DCA	E	/ AND RETURN.
	JMP I	EX	
TENT	HLT		/ TEST ENTRY AND
	TAD	E	/ FIX DATA FIELD FOR
	SNA		/ INTERPRETER
	JMP	+.4	/ INSTRUCTION FIELD.
	ISZ	TENT	
	UDF		
	JMP	+.3	
XDF	UDF		
	TAD	EME	
	DCA	EX	
	JMP I	TENT	
EX	0		
	CDF		
EME	0		/ CROSS-BANK ENTRY.
	CLA CLL		
	RDF		/ CALCULATE DATA FIELD
	TAD	EX+1	/ AND RETURN LINKAGE.
	DCA	XDF	
	TAD	XDF	
	IAC		
	DCA	XIF	
	JMP	R	
IFPC	HLT		/ INCREMENT LOCAL OR
	JMS	TENT	/ EXTENDED INSTRUCTION
	JMP	+.3	/ POINTER.

	ISZ	E	
	SKP		
	ISZ	EME	
	UDF		
	JMP I	IFPC	
ADR	0		/ OPERAND ADDRESS
	0		/ POINTERS.
	0		
P1	177		/ PAGE #1 CONSTANTS.
	200		
	7600		
	4000		
	3777		
	7		
EGO	DCA	FMQ	/ SAVE INSTRUCTION.
	TAD	FMQ	
	AND	P1	/ CALCULATE OPERAND ADDRESS.
	DCA	ADR	
	TAD	FMQ	
	AND	P1+1	
	SNA CLA		/ TEST PAGE INDICATOR.
	JMP	.+5	
	TAD	EX	
	AND	P1+2	
	TAD	ADR	
	DCA	ADR	
	TAD I	ADR	/ INDIRECT ADDRESSING
	DCA	ADR	/ IS IMPLICIT.
	TAD	ADR	
	IAC		/ SET ADDRESS POINTERS.
	DCA	ADR+1	
	TAD	ADR+1	
	IAC		
	DCA	ADR+2	
	JMS I	F8DF	/ SET OPERAND DATA FIELD.
	TAD I	ADR	
	DCA	BCX	/ MOVE OPERAND TO
	TAD I	ADR+1	/ FLOATING OPERAND
	AND	P1+3	/ REGISTER.
	DCA	BCS	
	TAD I	ADR+1	
	AND	P1+4	
	DCA	BCH	
	TAD I	ADR+2	
	DCA	BCL	
	JMS	IFPC	
	TAD	FMQ	/ GET OPERATION CODE.
	RTL		
	RTL		
	AND	P1+5	

	RAL		
	SZA		
	JMP	.+5	
	TAD	FMQ	/ IMMEDIATE LOAD FAC,
	JMS I	.+2	/ USE SINGLE FLOAT
	JMP	R	/ ROUTINE.
	BFLO		
	TAD	T1	/ CALCULATE BRANCH
FN2	DCA	ICHR	/ TO OTHER FUNCTIONS.
	TAD I	ICHR	
	DCA	ICHR	
	JMP I	ICHR	
T1	OPT-2		
ST1	JMS I	F8DF	/ STORE FAC TO MEMORY.
	TAD	ACX	
	DCA I	ADR	
	TAD	ACH	
	TAD	ACS	
	DCA I	ADR+1	
	TAD	ACL	
	DCA I	ADR+2	
	JMP	R	
JS1	JMS	TENT	
	JMP	.+6	
	TAD	E	
	DCA I	ADR	
	TAD	ADR+1	
JP2	DCA	E	
	JMP	R	
	TAD	EME	
	DCA I	ADR	
	TAD	ADR+1	
JP3	DCA	EME	
	JMP	R	
JP1	JMS	TENT	
	JMP	.+3	
	TAD	ADR	
	JMP	JP2	
	TAD	ADR	
	JMP	JP3	
F8DF	FDF1		
LD1	JMS I	XAB2	
	JMP	R	
XAB2	XAB1		
FDF1	HLT		
FDF2	UDF		
	JMP I	FDF1	
CDX1	TAD	FMQ	/ FLOATING OPERAND
	AND	SPC1-5	/ DATA FIELD SET.
	CLL RAL		

	RTL		
	TAD	SPC1-3	
	JMP	.+5	
RPC1	LFM		/ RESTORE INSTRUCTION
	TAD	SPC1-2	/ POINTER AND OPERAND
	DCA	E	/ FIELD AND RETURN
	TAD	SPC1-1	/ FROM FUNCTION.
	DCA	FDF2	
	JMP	R	
	7		
	UDF		
	CDF		
	HLT		
	HLT		
SPC1	HLT		/ SAVE LOCAL INSTRUCTION
	TAD	E	/ POINTER AND DATA FIELD
	DCA	SPC1-2	/ FOR FP8 USE OF FP8.
	TAD	FDF2	
	DCA	SPC1-1	
	TAD	SPC1-4	
	DCA	FDF2	
	EFM		
	JP	SPC1	
OPT	LD1		/ OPERATION BRANCH TABLE.
	ST1		
	AD1		
	SB1		
	MP1		
	DV1		
	RD1		
	JS1		
	CM1		
	JP1		
	FN1		
	FM1		
	USER		/ USER-DEFINED FUNCTIONS; OP=7000.
	17		
	OPF		
FN1	TAD	FMQ	
	RAR		
	RTR		
	AND	FN1-2	
	TAD	FN1-1	
	JMP I	.+1	
	FN2		
OPF	ZA1		/ FUNCTION BRANCH TABLE.
	AB1		
	CH1		
	AFIX		
	BFIX		

	IO1		
	CDX1		
	SQAR		
	SQRT		
	EXP		
	LOG		
	SIN		
	COS		
	ATN		
IO1	TAD	FMQ	
	AND	SPC1-5	
	TAD	.+2	
	JMP I	UPF-1	
	.+1		
	INI		/ IO BRANCH TABLE.
	OT1		
	CR1		
ACZA	HLT		/ TEST FAC FOR ZERO.
	TAD	ACH	
	SZA CLA		
	JMP	.+6	
	TAD	ACL	
	SZA CLA		
	JMP	.+3	
	DCA	ACX	
	JMP I	ACZA	
	ISZ	ACZA	
	JMP I	ACZA	
BCZA	HLT		/ TEST OPERAND FOR ZERO.
	TAD	BCH	
	SZA CLA		
	ISZ	BCZA	
	JMP I	BCZA	
RACL	HLT		/ ROTATE FAC LEFT
	TAD	ACL	/ AND DECREMENT
	CLL RAL		/ EXPONENT.
	DCA	ACL	
	TAD	ACH	
	RAL		
	DCA	ACH	
	CMA		
	TAD	ACX	
	DCA	ACX	
	JMP I	RACL	
RACR	HLT		/ ROTATE FAC RIGHT
	TAD	ACH	/ AND INCREMENT
	CLL RAR		/ EXPONENT.
	DCA	ACH	
	TAD	ACL	
	RAR		

	DCA	ACL	
	ISZ	ACX	
	JMP I	RACR	
	JMP I	RACR	
NORA	HLT		/ NORMALIZE FAC.
	JMS	ACZA	
	JMP I	NORA	
	TAD	ACH	
	SMA CLA		
	JMP	+.3	
	JMS	RACR	
	JMP I	NORA	
	JMS	RACL	
	JMP	-.6	
ZAI	DCA	ACL	/ CLEAR FAC.
	DCA	ACH	
	DCA	ACX	
	JMP	AB1	
CHI	STL RAR		/ CHANGE SIGN.
	TAD	ACS	
AB1	DCA	ACS	/ ABSOLUTE VALUE.
	JMP	R	
DTAD	HLT		/ DOUBLE PRECISION ADD
	CLA CLL		/ AC AND BC MANTISSAS.
	TAD	ACL	
	TAD	BCL	
	DCA	ACL	
	CLA RAL		
	TAD	ACH	
	TAD	BCH	
	DCA	ACH	
	JMP I	DTAD	
MPTN	HLT		/ MULTIPLY FAC BY TEN.
	TAD	ACH	
	DCA	BCH	
	TAD	ACL	
	DCA	BCL	
	JMS I	RR3	
	JMS I	RR3	
	JMS	DTAD	
	ISZ	ACX	
	CLL		
	JMS I	NOR	
	JMP I	MPTN	
DVTN	HLT		/ DIVIDE FAC BY TEN.
	TAD	ACH	
	DCA	FMO	
	TAD	ACL	
	JMS	DIV	
	RR3+2		

	DCA	ACH	
	JMS	DIV	
	RR3+2		
	DCA	ACL	
	TAD	RR3+1	
	TAD	ACX	
	DCA	ACX	
	JMS I	NOR	
	JMP I	DVTN	
XAB1	HLT		/ EXCHANGE FAC
	TAD	ACL	/ AND OPERAND.
	DCA	DTAD	
	TAD	ACH	
	DCA	MPTN	
	TAD	ACX	
	DCA	DVTN	
	TAD	ACS	
	CLL RAL		
	TAD	BCS	
	DCA	ACS	
	RAR		
	DCA	BCS	
	TAD	BCX	
	DCA	ACX	
	TAD	BCH	
	DCA	ACH	
	TAD	BCL	
	DCA	ACL	
	TAD	DTAD	
	DCA	BCL	
	TAD	MPTN	
	DCA	BCH	
	TAD	DVTN	
	DCA	BCX	
	JMP I	XAB1	
DIV	HLT		/ DIVIDE FMQ, AC BY
	DCA	DTAD	/ ADDRESSED NUMBER.
	TAD I	DIV	
	DCA	MPTN	/ QUOTIENT IN AC AND
	TAD I	MPTN	/ REMAINDER IN FMQ.
	CIA CLL		
	DCA	MPTN	
	TAD	RR3+3	
	DCA	XAB1	
	ISZ	DIV	
	JMP	+.11	
	TAD	FMQ	
	RAL		
	DCA	FMQ	
	TAD	FMQ	

	TAD	MPTN	
	SZL		
	DCA	FMQ	
	CLA		
	TAD	DTAD	
	RAL		
	DCA	DTAD	
	ISZ	XAB1	
	JMP	.-14	
	TAD	DTAD	
	JMP I	DIV	
MPY	HLT		/ MULTIPLY AC BY
	DCA	DTAD	/ ADDRESSED NUMBER.
	DCA	FMQ	
	TAD	RR3+4	/ HIGH PRODUCT IN FMQ,
	DCA	MPTN	/ LOW PRODUCT IN AC.
	TAD I	MPY	
	DCA	DIV	
	TAD I	DIV	
	DCA	DIV	
	ISZ	MPY	
	CLL		
	TAD	DTAD	
	RAR		
	DCA	DTAD	
	TAD	FMQ	
	SNL		
	JMP	.-+3	
	CLL		
	TAD	DIV	
	RAR		
	DCA	FMQ	
	ISZ	MPTN	
	JMP	.-13	
	TAD	DTAD	
	RAR		
	CLL		
	JMP I	MPY	
RR3	RACR		
	7774		
	5000		
	7763		
	7764		
	RACL		
NP1	JMS I	RR3+5	/ MULTIPLY FAC
	TAD	ACX	/ BY OPERAND.
	TAD	BCX	
	IAC		
	DCA	ACX	
	TAD	ACS	

	TAD	BCS	
	DCA	ACS	
	TAD	ACL	
	JMS I	MP4	
	BCL		
	CLA CLL		
	TAD	FMQ	
	DCA	BCS	
	TAD	ACL	
	JMS I	MP4	
	BCH		
	TAD	BCS	
	DCA	BCS	
	GLK		
	TAD	FMQ	
	DCA	ACL	
	GLK		
	DCA	BCX	
	TAD	ACH	
	JMS I	MP4	
	BCL		
	TAD	BCS	
	GLK		
	TAD	ACL	
	DCA	ACL	
	GLK		
	TAD	BCX	
	DCA	BCX	
	TAD	FMQ	
	TAD	ACL	
	DCA	ACL	
	GLK		
	TAD	BCX	
	DCA	BCX	
	TAD	ACH	
	JMS I	MP4	
	BCH		
	TAD	ACL	
	DCA	ACL	
	GLK		
	TAD	BCX	
	TAD	FMQ	
	DCA	ACH	
	JMS I	NOR	
	JMP	R	
SB1	STL RAR		/ SUBTRACT.
	TAD	BCS	
	DCA	BCS	
AD1	JMS I	BCZ	/ ADD OPERAND TO FAC.
	JMP	R	

	JMS I	ACZ	
	JMP I	MP4+1	
	JMS I	MP4+2	
	CLA SMA SZA		
	JMS I	MP4+3	
	TAD	BCX	
	CIA		
	TAD	ACX	
	SMA CLA		
	JMP	.+3	
	JMS I	MP4+4	
	JMP	.-6	
	TAD	ACS	
	TAD	BCS	
	SNA CLA		
	JMP	.+11	
	TAD	ACL	
	CLL CIA		
	DCA	ACL	
	TAD	ACH	
	CMA		
	SZL		
	IAC		
	DCA	ACH	
	TAD	BCS	
	DCA	ACS	
AR3	JMS I	MP4+5	
	JMS I	NOR	
	JMP	R	
RD1	JMS I	MP4+3	/ INVERSE DIVIDE.
DV1	JMS I	ACZ	/ NORMAL DIVIDE.
	JMP	R	
	JMS I	BCZ	
	JMP I	MP4+6	
	TAD	ACS	
	TAD	BCS	
	DCA	ACS	
	TAD	BCX	
	CIA		
	TAD	ACX	
	DCA	ACX	
	TAD	BCL	
	CLL RAL		
	DCA	BCL	
	TAD	BCH	
	RAL		
	DCA	BCH	
	TAD	ACH	
	JMS I	MP4	
	BCL		

	JMS I	MP4+7	
	BCH		
	CLL CIA		
	TAD	ACL	
	DCA	ACL	
	SNL		
	CMA		
	TAD	ACH	
	DCA	FMQ	
	TAD	ACL	
	JMS I	MP4+7	
	BCH		
	DCA	ACH	
	JMS I	MP4+7	
	BCH		
	DCA	ACL	
	JMP	AR3	
MP4	MPY		
	LDI		
	CPQ		
	XAB1		
	RACR		
	DTAD		
	ZAI		
	DIV		
SCR1	HLT		
	HLT		
	HLT		
FLOS	HLT		/ SINGLE, SIGNED
	JMS	BFLO	/ FLOAT AND ENTER.
	EFM		
	JP	FLOS	
FLOR	HLT		/ DOUBLE PRECISION
	JMS	AFLO	/ FLOAT AND ENTER.
	EFM		
	JP	FLOR	
AFLO	HLT		
	DCA	ACL	
	DCA	ACS	
	TAD	AFLO	
	DCA	BFLO	
	JMP	CFLO	
BFLO	HLT		
	SMA		
	JMP	.+5	
	CIA		
	DCA	ACL	
	STL RAR		
	SKP		
	DCA	ACL	

	DCA	ACS	
	DCA	ACH	
CFLO	TAD	T5	
	DCA	ACX	
	JMS I	NOR	
	RDF		
	TAD	T5+1	
	DCA	.+1	
	HLT		
	JMP I	BFLO	
FXQ	HLT		
	TAD	ACX	
	SMA		
	JMP	.+5	
	CLA CLL		
	DCA	ACH	
	DCA	ACL	
	JMP I	FXQ	
	TAD	T5+2	
	SMA		
	JMP	.-6	
	DCA	AFLO	
	JMS I	T5+3	
	ISZ	AFLO	
	JMP	.-2	
	TAD	ACL	
	JMP I	FXQ	
CFIX	HLT		
	JMS	FXQ	
	SNL CLA		
	JMP I	CFIX	
	ISZ	ACL	
	JMP I	CFIX	
	ISZ	ACH	
	JMP I	CFIX	
BFIX	JMS	CFIX	
	TAD	ACS	/ SINGLE, SIGNED FIX
	CLL RAL		/ AND LEAVE.
	TAD	ACL	
	SZL		
	CIA		
	DCA	ACL	
	SKP		
AFIX	JMS	CFIX	/ DOUBLE PRECISION
	TAD I	T5+4	/ FIX AND LEAVE.
	IAC CLL		
	DCA	CFIX	
	TAD I	T5+5	
	DCA	.+2	
	JMS I	T5+6	

	HLT		
	DCA	E	
	TAD	ACL	
	JMP I	CFIX	
T5	27		
	CIF		
	7751		
	RACR		
	EX		
	XIF		
	TENT		
	IFPC		
CPQ	HLT		/ COMPARE OPERAND
	TAD	BCX	/ MAGNITUDES.
	CIA		
	TAD	ACX	
	SZA		
	JMP I	CPQ	
	TAD	BCH	
	CIA		
	TAD	ACH	
	SZA		
	JMP I	CPQ	
	TAD	BCL	
	CIA CLL		
	TAD	ACL	
	CML		
	SZA		
	RAR		
	JMP I	CPQ	
CZ	TAD	ACS	
	SZA CLA		
	JMP	R	
	JMS I	T5+7	
	JMS I	T5+7	
	JMP	R	
CM1	JMS I	ACZ	/ THREE-WAY BRANCH.
	JMP	RAZ	
	JMS I	BCZ	
	JMP	CZ	
	TAD	ACS	
	TAD	BCS	
	SZA CLA		
	JMP	CZ	
	JMS	CPQ	
	SNA		
	JMP	CZ+4	
	SMA CLA		
	JMP	CZ	
	TAD	BCS	

	SZA	CLA	
	JMP	CZ+3	
	JMP	R	
RAZ	JMS I	BCZ	
	JMP	CZ+4	
	JMP	-6	
CHAR	HLT		/ INPUT CHARACTER PROCESSOR.
	JMS I	INR	
	AND	K5	/ MASK TO SEVEN BITS.
	SNA		/ IGNORE NULL.
	JMP	CHAR+1	
	DCA	ICHR	/ SAVE CHARACTER.
	TAD	ICHR	
	TAD	K5+1	/ TEST FOR RUB-OUT.
	SNA		
	JMP	RBO	
	TAD	K5+2	/ TEST FOR 'E'.
	SNA		
	JMP	EXPN	
	TAD	K5+3	/ TEST FOR DECIMAL POINT.
	SNA		
	JMP	DC5	
	IAC		/ TEST FOR MINUS.
	SNA		
	JMP	MN5	
	TAD	K5+4	/ IGNORE PLUS.
	SNA		
	JMP	CHAR+1	
	TAD	K5+5	/ BLANK MAY TERMINATE.
	SNA		
	JMP	BK5	
	TAD	K5+6	/ IGNORE LINE-FEED.
	SNA		
	JMP	CHAR+1	
	TAD	K5+7	/ TERMINATE IF NOT
	SMA SZA		/ DECIMAL DIGIT.
	JMP	INAN	
	TAD	K5+10	
	SPA		
	JMP	INAN	/ INCREMENT COUNTER,
	DCA	ACX	/ RETURN FOUR-BIT
	ISZ	CHRX	/ DIGIT IN ACX.
	JMP I	CHAR	
MN5	STL	CLA RAR	
	ISZ	CHAR	
	JMP I	CHAR	
DC5	CLA	CMA	
	JMP	IN2-1	
RBO	TAD	ROC	
	JMS I	OTR	

BK5	JMP	IN1	
	TAD	CHRX	
	SNA CLA		
	JMP	CHAR+1	
INAN	CLA CLL		/ INTERPRET AND NORMALIZE
	TAD	K5+3	/ INPUT NUMBER.
	DCA	ACX	
	JMS I	NOR	
	TAD	BCL	
	CLL RAL		
	TAD	BCH	
	SZL		
	CIA CLL		
	TAD	BCS	
	SNA		
	JMP	R	
	DCA	BCS	
	TAD	BCS	
	SMA		
	CIA		
	DCA	BCX	
	TAD	BCS	
ROC	SMA CLA		
	JMP	.+3	
	TAD	K5+12	
	SKP		
	TAD	K5+11	
	DCA	CHAR	
	JMS I	CHAR	
	ISZ	BCX	
	JMP	.-2	
	JMP	R	
K5	177		
	7601		
	72		
	27		
	2		
	13		
	26		
	7720		
	12		
	MPTN		
	DVTN		
	MPY		
IN1	DCA	CHRX	
	DCA	ACL	
	DCA	ACH	
	DCA	ACS	
	DCA	BCL	
	DCA	BCH	

	DCA	BCS	
	DCA	BCX	
IN2	JMS	CHAR	
	JMP	.+3	
	DCA	ACS	
	JMP	IN2	
	TAD	BCS	
	TAD	BCX	
	DCA	BCS	
	TAD	ACH	
	JMS I	K5+13	
	K5+10		
	DCA	ACH	
	TAD	ACL	
	JMS I	K5+13	
	K5+10		
	TAD	ACX	
	DCA	ACL	
	CLA RAL		
	TAD	ACH	
	TAD	FMQ	
	DCA	ACH	
	JMP	IN2	
EXPN	JMS	CHAR	
	JMP	.+3	
	DCA	BCL	
	JMP	EXPN	
	TAD	BCH	
	JMS I	K5+13	
	K5+10		
	TAD	ACX	
	DCA	BCH	
	JMP	EXPN	
	HLT		
OT1	TAD	FMT	/ FIXED-POINT OUTPUT.
	AND	T6A	
	CIA		
	DCA	FRWD	
	TAD	FMT	
	RTR		
	RTR		
	RTR		
	AND	T6A	
	CIA		
	DCA	INWD	
	TAD	ACS	
	SMA CLA		
	JMP	.+3	
	TAD	T6A+4	
	SKP		

	TAD	T6A+3
	DCA	T6A+7
	DCA	ACS
	JMS I	SVPC
	ST	T6B
	LFM	
	JMS I	ACZ
	JMP	ZOT
	TAD	FRWD
	SMA	
	CLA CMA	
	DCA	SPAC
	EFM	
	ILA	5
	LFM	
	JMS I	T6B+1
	ISZ	SPAC
	JMP	.-2
	EFM	
	AD	T6B
	LFM	
	TAD	ACX
	SMA SZA CLA	
	JMP	IGOT
	JMP	.-3
ZOT	TAD	T6A+6
	DCA	T6A+7
	IAC	
	TAD	INWD
	SPA	
	JMS	SPAC
	TAD	T6A+7
	JMS I	OTR
FRO	TAD	FRWD
	SMA CLA	
	JMP	ODN
	TAD	T6A+5
	JMS I	OTR
	ISZ	FRWD
	SKP CLA	
	JMP	ODN
	JMS I	T6B+2
	JMP	.-4
IGOT	DCA	CHRX
	JMS I	T6B+1
	ISZ	CHRX
	TAD	ACX
	SMA SZA CLA	
	JMP	.-4
	TAD	INWD

	TAD	CHRX
	SPA	
	JMP	OCN
	CLA	
	TAD	FRWD
	TAD	INWD
	DCA	CHRX
	TAD	T6A+3
	SKP	
	TAD	T6A+5
	JMS I	OTR
	ISZ	CHRX
ODN	JMP	.-3
	EFM	
	LD	T6B
OCN	JP	RSPC
	IAC	
	SPA	
	JMS	SPAC
	TAD	CHRX
	CMA	
	DCA	CHRX
	TAD	T6A+7
	JMS I	OTR
	ISZ	CHRX
	SKP CLA	
	JMP	FRO
	JMS I	T6B+2
	JMP	.-4
T6B	SCR1	
	DVTN	
	DGO	
FRWD	HLT	
INWD	HLT	
T6A	77	
	212	
	215	
	240	
	255	
	256	
	260	
	HLT	
	HLT	
SPAC	HLT	
	DCA	T6A+10
	TAD	T6A+3
	JMS I	OTR
	ISZ	T6A+10
	JMP	.-3
	JMP I	SPAC

CR1	TAD	T6A+2	
	JMS I	OTR	
	TAD	T6A+1	
	JMS I	OTR	
	JMP	R	
SQAR	JMS I	SVPC	
	ST	T6B	
	MP	T6B	
	JP	RSPC	
SCR2	HLT		
	HLT		
	HLT		
DGO	HLT		/ PRINT FAC HIGH
	JMS I	T7A	/ DECIMAL DIGIT.
	EFM		
	ST	T7A+2	
	LFM		
	JMS I	T7A+1	
	DCA	T7B	
	TAD	T7B	
	TAD	T7B+1	
	JMS I	OTR	
	TAD	T7B	
	CIA		
	SFLO		
	AD	T7A+2	
	LFM		
	JMP I	DGO	
FM1	TAD	FMQ	/ SET OUTPUT FORMAT.
	AND	T7B+2	
	DCA	T7B	
	TAD	FMQ	
	AND	T7B+3	
	CLL RAL		
	RTL		
	TAD	T7B	
	DCA	FMT	
	JMP	R	
T7A	MPTN		
	FXQ		
	SCR2		
T7B	HLT		
	260		
	7		
	370		
SCR3	HLT		
	HLT		
	HLT		
PIOT	1		
	3110		

	3755		
ONE	1		
	2000		
HALF	0		
	2000		
ZERO	0		
	0		
	0		
XK2	0		
	2000		
	0		
XTA	ONE		
	XK1		
	SCR1		
	XK1+3		
	XK2		
	SCR2		
	SCR3		
	ZERO		
	EXPS		
	EXPT		
EXP	JMS I	SVPC	/ NATURAL EXPONENT.
	MP	XTA+1	
	ST	XTA+2	
	SFIX		
	DCA	T7B	
	TAD	T7B	
	IAC		
	DCA	XK2	
	TAD	T7B	
	CIA		
	SFLO		
	AD	XTA+2	
	MP	XTA+3	
	ST	XTA+2	
	LD	XTA	
	ST	XTA+5	
	ILA	10	
EXPT	ST	XTA+6	
	RDV	XTA+2	
	MP	XTA+5	
	AD	XTA	
	ST	XTA+5	
	LD	XTA+6	
	SB	XTA	
	CMP	XTA+7	
	JP	XTA+10	
	JP	XTA+10	
	JP	XTA+11	
EXPS	LD	XTA+5	

	MP	XTA+4	
	JP	RSPC	
LGK	LGC		
	LGC+3		
	LGC+6		
	LGC+11		
	LGC+14		
	LGC+17		
	LGC+22		
	HALF		
LOG	TAD	ACX	/ NATURAL LOGARITHM.
	DCA	XK2	
	IAC		
	DCA	ACX	
	DCA	ACS	
	JMS I	ACZ	
	JMP	R	
	JMS I	SVPC	
	AD	LGK+1	
	ST	XTA+5	
	SB	LGK	
	DV	XTA+5	
	ST	XTA+5	
	MP	XTA+5	
	ST	XTA+2	
	MP	LGK+3	
	AD	LGK+4	
	MP	XTA+2	
	AD	LGK+5	
	MP	XTA+2	
	AD	LGK+6	
	MP	XTA+5	
	ST	XTA+5	
	LFM		
	TAD	XK2	
	SFLO		
	SB	LGK+7	
	AD	XTA+5	
	MP	LGK+2	
	JP	RSPC	
LGK	2		
	2650		
	1172		
	1		
	2650		
	1172		
	0		
	2613		
	4414		
	-1		

	3362		
	5646		
	0		
	2234		
	6604		
	0		
	3661		
	6112		
	2		
	2705		
	2436		
XK1	1		
	2705		
	2436		
	0		
	2613		
TSQ	4414		
	SCR1		
	SCR2		
	0		
	HALF		
	ONE		
	7774		
	SQRTL		
	0		
SQRT	DCA	ACS	/ SQUARE ROOT.
	JMS I	ACZ	
	JMP	R	
	TAD	ACX	
	SPA		
	CML		
	RAR		
	DCA	SQRT-1	
	GLK		
	DCA	ACX	
	TAD	TSQ+5	
	DCA	TSQ+2	
	JMS I	SVPC	
	ST	TSQ	
	AD	TSQ+4	
	JP	TSQ+6	
	EFM		
	ST	TSQ+1	
	RDV	TSQ	
	AD	TSQ+1	
SQRTL	MP	TSQ+3	
	LFM		
	ISZ	TSQ+2	
	JMP	.-7	
	TAD	ACX	

	TAD	SQRT-1	
	DCA	ACX	
	JMP I	RSPC	
ATN	TAD	ACS	/ ARCTANGENT.
	DCA	TSQ+2	
	DCA	ACS	
	TAD	ACX	
	DCA	TSQ+7	
	JMS I	SVPC	
	CMP	TSQ+4	
	JP	KAT+1	
	JP	KAT+1	
ATL	RDV	TSQ+4	
	ST	KAT+11	
	MP	KAT+11	
	ST	KAT+10	
	DV	KAT+7	
	AD	KAT+6	
	RDV	KAT+10	
	AD	KAT+5	
	RDV	KAT+10	
	AD	KAT+4	
	RDV	KAT+10	
	AD	KAT+3	
	RDV	KAT+10	
	AD	KAT+2	
	RDV	KAT+10	
	AD	TSQ+4	
	MP	KAT+11	
	LFM		
	TAD	TSQ+7	
	SPA SNA CLA		
	JMP	.+5	
	EFM		
	CHS		
	AD	KAT	
	LFM		
	TAD	TSQ+2	
	DCA	ACS	
	JMP I	RSPC	
KAT	PIOT		
	ATL		
	XAT		
	XAT+5		
	XAT+10		
	XAT+13		
	XAT+2		
	XAT+16		
	SCR1		
	SCR2		

XAT	2		
	7000		
	6		
	6622		
	2322		
	0		
	6161		
	6351		
	4		
	7743		
	0460		
	7776		
	6456		
	2636		
	7775		
	7353		
	1611		
COS	DCA	ACS	/ COSINE.
	JMS I	SVPC	
	AD	KR1	
	JP	.+1	
	SIN+1		
KS1	SCR1		
	SCR2		
	SCR3		
KR1	PIOT		
	FXQ		
	ONE		
	0		
SCOS	.+1		
SCOX	HLT		
	RDV	KS1+1	
	MP	KS1+2	
	CHS		
	AD	KR1+2	
	ST	KS1+2	
	JP	SCOX	
SIN	JMS I	SVPC	/ SINE.
	LFM		
	TAD	ACS	
	DCA	KR1+3	
	DCA	ACS	
	EFM		
	DV	KR1	
	ST	KS1	
	LFM		
	JMS I	KR1+1	
	RTR		
	DCA	SCOX	
	CLA RAR		

TAD	KR1+3
DCA	KR1+3
DCA	ACH
TAD	ACL
DFLO	
CHS	
AD	KS1
ST	KS1
LFM	
TAD	SCOX
SMA	
JMP	.+5
EFM	
LD	KR1+2
SB	KS1
LFM	
EFM	
MP	KR1
ST	KS1
MP	KS1
ST	KS1+1
LD	KR1+2
ST	KS1+2
322	
JS	SCOS
234	
JS	SCOS
156	
JS	SCOS
110	
JS	SCOS
52	
JS	SCOS
24	
JS	SCOS
6	
JS	SCOS
MP	KS1
LFM	
TAD	KR1+3
DCA	ACS
JMP I	RSPC
	END FP8 MAIN CODE

\$

```

/      SAMPLE USER OUTPUT ROUTINE
/      USES 'PRINTA' ROUTINE IN FIELD '0'
UOR    HLT
        CIF+00
        JMS I      .+3
        CLA CLL
        JMP I      UOR
        PRINTA
/      SAMPLE USER INPUT ROUTINE
/      USES 'READA' ROUTINE IN FIELD '0'
UIR    HLT
        CIF+00
        JMS I      .+2
        JMP I      UIR
        READA
/      SAMPLE USER-DEFINED FUNCTION
USER   JMS I      SVPC      / SAVE FPC AND EFM.
/      ANY CODE (SEE TEXT)
        JMS I      RSPC      / RESTORE AND RETURN.
$      END USER FUNCTIONS

```

/

/ THE FOLLOWING CODE OVERLAYS THE SOFTWARE

/ MULTIPLY AND DIVIDE ROUTINES.

/ REMOVE IF EAE NOT AVAILABLE.

/

```

      .LOC      DIV
      HLT                               / FP8 EAE DIVIDE.
      7421
      TAD I      DIV
      ISZ        DIV
      DCA        MPTN
      TAD I      MPTN
      DCA        .+3
      TAD        FMQ
      7407
      HLT
      DCA        FMQ
      7501
      JMP I      DIV
      .LOC      MPY
      HLT
      DCA        .+6      / FP8 EAE MULTIPLY.
      TAD I      MPY
      ISZ        MPY
      DCA        MPTN
      TAD I      MPTN
      7425
      HLT
      DCA        FMQ
      7501
      JMP I      MPY
      .END

```

\$ END FIELD '1'