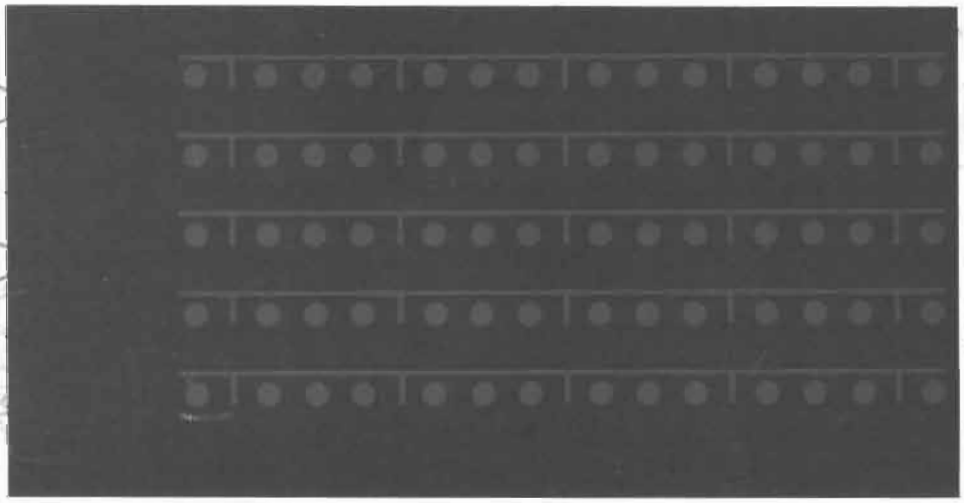
2116A COMPUTER

HEWLETT · PACKARD

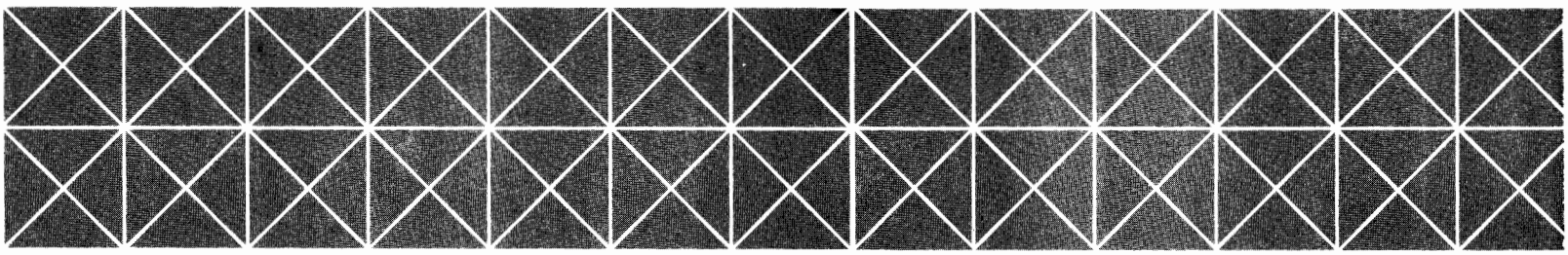


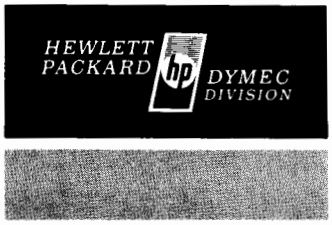
FORTRAN LIBRARY ROUTINES

FORTRAN LIBRARY ROUTINES

reference manual

HEWLETT · PACKARD 2116A COMPUTER





395 Page Mill Road, Palo Alto, California 94306 Area Code 415 326-1755

MARCH 1967

PRINTED IN U.S.A.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

CONTENTS

INTRODUCTION

This manual contains descriptions of the HP 2116A FORTRAN library routines. These routines form three general categories: Math Routines, Floating Point and Exponential Routines, and Service Routines.

The Math Routines may be called from a FORTRAN program by specification of a FORTRAN function of the general form:

$$\text{function } (p_1, p_2, \ldots, p_n)$$

The FORTRAN compiler then generates a call to the library routine.

The Floating Point and Exponential Routines are called from a FORTRAN program as a result of arithmetic expressions in the FORTRAN language. For example, I**K would generate a call to the .ITOI routine.

The Service Routines are those which are called and used by the other two classes of FORTRAN library routines.

All three classes of FORTRAN library routines may be called from an Assembly-language program. The Assembler calling sequence is listed with the description; routine names must be declared as EXTernal points in the program.

When a FORTRAN or Assembly-language object program which references FORTRAN library routines is loaded for execution, the FORTRAN library must also be loaded. The BCS Relocating Loader will load only those routines called by the user's object program.

The execution times listed in this manual are approximate. The i, j, x, y, and z notations indicate variable names or array element names to be provided by the user in the function or calling sequence. The variables i and j indicate integer, or fixed point, values; x and y indicate real, or floating point, values. z is used to indicate that the value may be integer or real.

CHAPTER 1
MATH ROUTINES

PURPOSE:	To calculate the absolute value of x, where x has a floating point value.
CALLING SEQUENCE:	DLD x JSB ABS
FORTRAN FUNCTION:	ABS(x)
NORMAL RETURN:	Floating point result is left in the A and B registers.
STORAGE:	5 words
EXECUTION TIME:	For x ≥ 0, .006 ms For x < 0, minimum = .3 ms average = .5 ms maximum = .9 ms
ACCURACY:	23 bits

ALOG

PURPOSE: To calculate the natural logarithm of x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD   x
JSB   ALOG
```

FORTRAN FUNCTION: ALOG(x)

NORMAL RETURN: Floating point result is left in the A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 70 words

EXECUTION TIME:

Minimum = 6.12 ms
Average = 7.7 ms
Maximum = 10.65 ms

ACCURACY: 22 bits

ERROR CONDITIONS:

Condition	Error Code
$x \leq 0$	Ø2 UN

ALGORITHM:

Let f = mantissa (x)
i = characteristic (x)

(that is, $x = 2^{i.f}$)

Then answer = $(i + \log_2 f) \cdot (\log_e 2)$

$$= \log_e 2 \left(i + z \left[c_1 + \frac{c_2}{c_3 - z^2} \right] - 1/2 \right)$$

where

$$z = \frac{f - \sqrt{2/2}}{f + \sqrt{2/2}}$$

and

$c_1 = 1.2920070987$
$c_2 = 2.6398577035$
$c_3 = 1.6567626301$

PURPOSE: To calculate the Arctangent of x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD   x
JSB   ATAN
```

FORTRAN FUNCTION: ATAN(x)

NORMAL RETURN: Result is in radians, in the range $-\pi/2$ to $\pi/2$. Floating point result is left in A and B registers.

STORAGE: 87 words

EXECUTION TIME:

Minimum = 15.82 ms
Average = 20.9 ms
Maximum = 28.8 ms

ACCURACY: 21 bits

ALGORITHM:

if abs(X) >1 then U=1/X else U=X
Y=U*Cheby(2*U*U -1)
if abs(X) ≤ 1 then answer = Y
else if X > 0 then answer = $\pi/2$-Y
else answer = $-\pi/2$-Y

COS

PURPOSE: To calculate the cosine of x, where x has a floating point value expressed in radians.

CALLING SEQUENCE:

```
DLD   x
JSB   COS
```

FORTRAN FUNCTION: COS(x)

NORMAL RETURN: Floating point result is left in A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 8 words

EXECUTION TIME:

Minimum = 10.26 ms
Average = 13.8 ms
Maximum = 20.25 ms

ACCURACY: See SIN.

ERROR CONDITIONS:

Condition	Error Code
$ABS(x) > 2^{14}$	Ø5 OR

ALGORITHM: $COS(x) = -SIN(x-\pi/2)$

PURPOSE: To calculate e^x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD   x
JSB   EXP
```

FORTRAN FUNCTION: EXP(x)

NORMAL RETURN: Floating point result is left in the A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 87 words

EXECUTION TIME:

Minimum = 6.12 ms
Average = 7.7 ms
Maximum = 10.65 ms

ACCURACY: $e^{x+y} = e^x * e^y$

Hence, an absolute error of y in the representation of x introduces an error of $(e^{x+y} - e^x)$ into the value of EXP(x).

Thus for values of x with $|x| < 1$, 23 bits of accuracy can be expected, while for values of x with |x| near 100, only 15 bits of accuracy can be expected.

ERROR CONDITIONS:

Condition	Error Code
$x*\log_2 e \geq 124$	Ø7 OF

ALGORITHM: Let i=ENTIER(x), and $f = x*\log_2 e - 1$ (See .IENT)

$$\text{Answer} = 2^i * \left[1 + \frac{2f}{c_4 + c_3 f^2 - f - c_2/(f^2 + c_1)} \right]$$

where

$c_1 = 87.417497202$
$c_2 = 617.9722695$
$c_3 = 0.03465735903$
$c_4 = 9.9545957821$

FLOAT

PURPOSE:	To convert the integer i to floating point format.
CALLING SEQUENCE:	LDA i JSB FLOAT
FORTRAN FUNCTION:	FLOAT(i)
NORMAL RETURN:	Floating point result is left in the A and B registers.
EXECUTION TIME:	Minimum = .1 ms Average = .2 ms Maximum = .3 ms
STORAGE:	10 words

PURPOSE:	To calculate absolute value of i, where i has an integer value.
CALLING SEQUENCE:	LDA i JSB IABS
FORTRAN FUNCTION:	IABS (i)
NORMAL RETURN:	Integer result is left in A register.
EXECUTION TIME:	.0096 ms
STORAGE:	7 words
ACCURACY:	23 bits
ERROR CONDITIONS:	None

IFIX

PURPOSE:	To convert the floating point number x to integer format.
CALLING SEQUENCE:	DLD x JSB IFIX
FORTRAN FUNCTION:	IFIX(x)
NORMAL RETURN:	Integer result is left in A register. Any fractional portion is truncated. If the integer portion is greater than 2^{15}, the A register is set to all 1's.
STORAGE:	29 words

PURPOSE:	To calculate sign of z times |i| where z has an integer or floating point value and i has an integer value.
CALLING SEQUENCE:	JSB ISIGN DEF i DEF z
FORTRAN FUNCTION:	ISIGN (i,z)
NORMAL RETURN:	Integer result is left in A register.
STORAGE:	26 words
EXECUTION TIME:	.05 ms
ACCURACY:	23 bits
ALGORITHM:	Same as SIGN.

SIGN

PURPOSE: To calculate sign of y times |x| , where x has a floating point and z has a floating point or integer value. In the case where z=0, the result of the SIGN function is 0, regardless of the value of |x| .

CALLING SEQUENCE:

```
JSB  SIGN
DEF  x
DEF  z
```

FORTRAN FUNCTION: SIGN (x, z)

NORMAL RETURN: Floating point result is left in A and B registers.

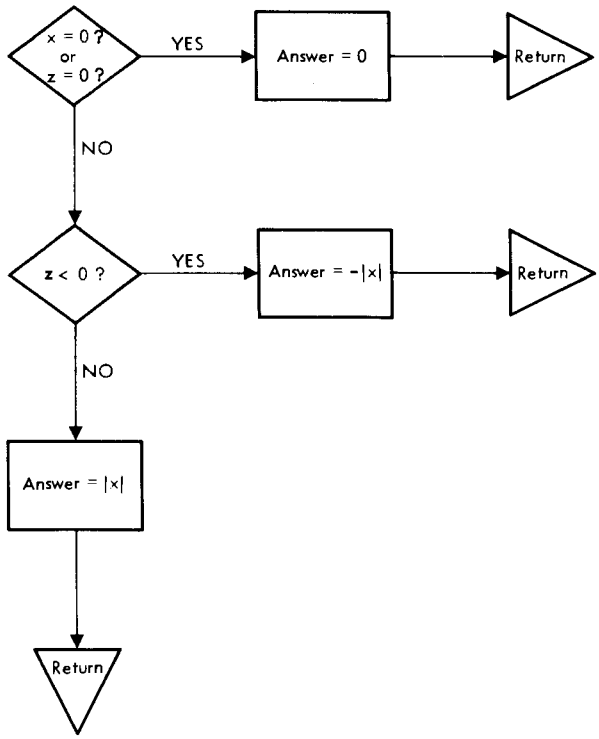
STORAGE: 30 words

EXECUTION TIME:

Minimum = .3 ms
Average = .5 ms
Maximum = .9 ms

ACCURACY: 23 bits

ALGORITHM:



PURPOSE: To calculate sine of x, where x has a floating point value expressed in radians.

CALLING SEQUENCE:

```
DLD   x
JSB   SIN
```

FORTRAN FUNCTION: SIN(x)

NORMAL RETURN: Floating point result is left in the A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 64 words

EXECUTION TIME:

Minimum = 10.26 ms

Average = 13.8 ms

Maximum = 20.25 ms

ACCURACY: For small y, we have $\sin(n\pi + y) = \sin(y) \approx y$

That is, the accuracy of sin(x) for x near a non-zero integer multiple of π is limited by the accuracy of the subtraction $n\pi - x$.

For example, sin(3.14) would have approximately 4 decimal digits of accuracy (the possible 7 minus the 3-digit closeness to π).

ERROR CONDITIONS:

Condition	Error Code
$ABS(x) > 2^{14}$	Ø5 OR

ALGORITHM:

X=X*2/π

X=X-4*ENTIER((X+1)4) (See .IENT)

if X > 1 then X= 2-X

answer = X*Cheby (2*X*X-1)

SQRT

PURPOSE: To calculate the square root of x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD  x
JSB SQRT
```

FORTRAN FUNCTION: SQRT(x)

NORMAL RETURN: Floating point result is left in A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 77 words

EXECUTION TIME:

Minimum = 3.94 ms
Average = 4.8 ms
Maximum = 6.45 ms

ACCURACY: 22 bits

ERROR CONDITIONS:

Condition	Error Code
$x < 0$	Ø3 UN

ALGORITHM:

Choose f such that $x=2^{2b}(f)$, $.25 \le f < 1$

Then $\sqrt{x}=2^{b}\sqrt{f}$

$\sqrt{f}$ is approximated by $p_1 = c_1 f + c_2$, where for

$.25 \le f < .5$, $c_1 = .875$, $c_2 = .27863$

and for $.5 \le f < 1$, $c_1 = .578125$, $c_2 = .421875$

This approximation is improved by two Newton iterations:

$p_2 = (p_1 + f/p_1)/2$
$p_3 = (p_2 + f/p_2)/2$
p_3 is the final result

PURPOSE: To calculate tangent of x, where x has a floating point value expressed in radians.

CALLING SEQUENCE:

```
DLD   x
JSB   TAN
```

FORTRAN FUNCTION: TAN(x)

NORMAL RETURN: Floating point result is left in the A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 87 words

EXECUTION TIME:

Minimum = 14.28 msec.

Average = 19.4 msec.

Maximum = 28.8 msec.

ACCURACY: Approximately as accurate as SIN(x)/COS(x).

ERROR CONDITIONS:

Condition	Error Code
$x > 2^{14}$	Ø9 OR
$\tan(x) > 2^{128}$	(floating point overflow)

ALGORITHM:

$X=4*X/\pi$

X=X-4*ENTIER((X+1)/4) (See .IENT)

if X > 1 then Y=2-X else Y=X.

Y=Y*CHEBY (2*Y*Y-1)

if X > 1 then answer = 1/Y else answer = Y.

TANH

PURPOSE: To calculate hyperbolic tangent of x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD     x
JSB     TANH
```

FORTRAN FUNCTION: TANH(x)

NORMAL RETURN: Floating point result is left in the A and B registers.

STORAGE: 99 words

EXECUTION TIME: Depends on the range in which x lies. Always less than twice as long as EXP.

ACCURACY: At least 20 bits correct.

ALGORITHM:

1. $x \geq 0$
   a. $x \geq 16$
      $TANH(x) = 1$
   b. $.125 \leq x < 16$
      $TANH(x) = (EXP(2*x) - 1)/(EXP(2*x) + 1)$
   c. $.00005 \leq x < .125$

$$TANH(x) = \left\{ c_1 + f^2 \left[c_2 + c_3 \left(c_4 + f^2 \right)^{-1} \right] \right\}^{-1}$$

   where:

$$f = 4*x*\log_2 e$$
$$c_1 = 5.7707801636$$
$$c_2 = .01732867951$$
$$c_3 = 14.1384114018$$
$$c_4 = 349.6699888$$

   d. $x < .00005$
      $TANH(x) = x$
2. $x < 0$
   $TANH(x) = -TANH(-x)$

CHAPTER 2
FLOATING POINT AND EXPONENTIAL ROUTINES

PURPOSE:	To divide the two-word integer quantity i by the integer quantity j.
CALLING SEQUENCE:	JSB .DIV DEF j
ASSEMBLY LANGUAGE:	DIV j (i is a two-word quantity in the B and A registers)
NORMAL RETURN:	Result is left in the A and B registers; the dividend is left in A, the remainder in B.
STORAGE:	49 words
EXECUTION TIME:	Minimum = .27 ms Average = .3 ms Maximum = .31 ms

.DLD

PURPOSE: To load x, a two-word quantity, into the A and B registers.

CALLING SEQUENCE:

```
JSB    .DLD
DEF    x        (x= address of the first word of the
                 two-word quantity)
```

ASSEMBLY LANGUAGE:

```
DLD    x        (x= address of the first word of the
                 two-word quantity)
```

NORMAL RETURN: The quantity in locations x and x+1 will be loaded into the A and B registers.

STORAGE: 30 words

EXECUTION TIME: .06 ms

PURPOSE:	To store a two word quantity from the A and B registers to two consecutive locations in memory.
CALLING SEQUENCE:	JSB .DST DEF x
ASSEMBLY LANGUAGE:	DST x
NORMAL RETURN:	The quantity in the A and B registers will be stored in locations x and x+1 .
STORAGE:	30 words
EXECUTION TIME:	.06 ms

.FAD

PURPOSE: To add two floating point numbers, x and y.

CALLING SEQUENCE:

```
JSB   .FAD
DEF   y        (x is assumed to be in A and B)
```

ASSEMBLY LANGUAGE:

```
FAD   y        (x is assumed to be in A and B)
```

NORMAL RETURN: The floating point sum is left in the A and B registers.

STORAGE: 94 words

EXECUTION TIME:

Minimum = .3 ms

Average = .5 ms

Maximum = .9 ms

PURPOSE:	To subtract the floating point number y from the floating point number x.
CALLING SEQUENCE:	JSB .FSB DEF y (x is assumed in A and B)
ASSEMBLY LANGUAGE:	FSB y (x is assumed in A and B)
NORMAL RETURN:	The floating point difference (x-y) is placed in the A and B registers.
STORAGE:	94 words
EXECUTION TIME:	Minimum = .3 ms Average = .5 ms Maximum = .9 ms

.FDV

PURPOSE: To divide the floating point quantity x by the floating point quantity y.

CALLING SEQUENCE:

```
JSB    .FDV
DEF    y        (x is assumed in A and B)
```

ASSEMBLY LANGUAGE:

```
FDV    y        (x is assumed in A and B)
```

NORMAL RETURN: The floating point quotient is left in the A and B registers.

STORAGE: 66 words

EXECUTION TIME:

Minimum = 1.2 ms

Average = 1.3 ms

Maximum = 1.5 ms

PURPOSE:	To multiply the two floating point numbers x and y.
CALLING SEQUENCE:	JSB .FMP DEF x (y is assumed in A and B)
ASSEMBLY LANGUAGE:	FMP x (y is assumed in A and B)
NORMAL RETURN:	The floating point product is returned in the A and B registers.
STORAGE:	53 words
EXECUTION TIME:	Minimum = .64 ms Average = .7 ms Maximum = .75 ms

.ITOI

PURPOSE: To calculate i^j, where i and j have integer values.

CALLING SEQUENCE:
```
JSB  .ITOI
DEF  i
DEF  j
```

NORMAL RETURN: Integer result is left in A register. If error condition (below) occurs, overflow bit is set.

STORAGE: 77 words

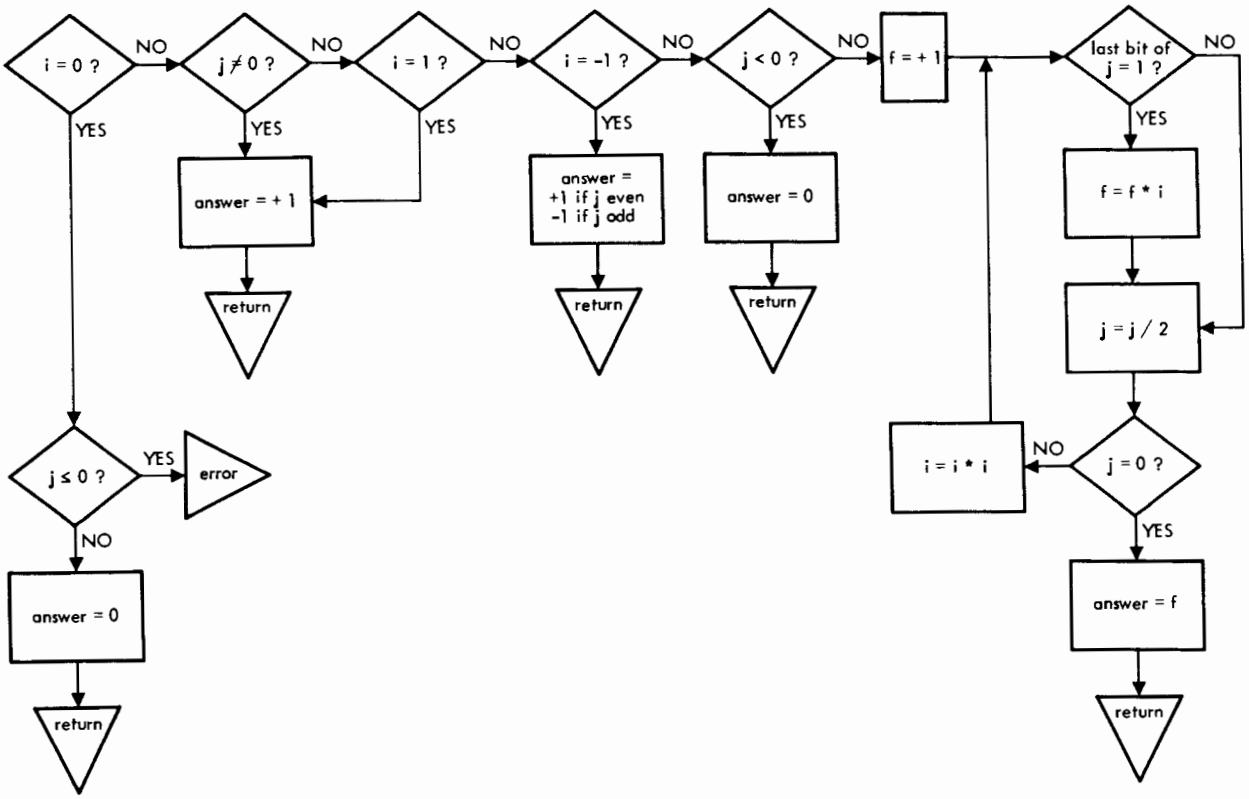
EXECUTION TIME:
Minimum = $(1.5)(\log_2 j)(.09)$ ms
Average = $(1.5)(\log_2 j)(.12)$ ms
Maximum = $(1.5)(\log_2 j)(.18)$ ms

ACCURACY: 23 bits

ERROR CONDITIONS:

Condition	Error Code
$i = 0,\ j \leq 0$	Ø8 UN
$i^j \geq 2^{23}$	Ø8 OF

ALGORITHM:



PURPOSE:	To multiply the two integer quantities i and j.	
CALLING SEQUENCE:	JSB .MPY	
	DEF i	(j is assumed to be in the A register)
ASSEMBLY LANGUAGE:	MPY i	(j is assumed to be in the A register)
NORMAL RETURN:	The result is left in the A and B registers: B containing most significant bits, A containing least significant bits.	
STORAGE:	72 words	
EXECUTION TIME:	Minimum = .09 ms	
	Average = .12 ms	
	Maximum = .15 ms	

.RTOI

PURPOSE:

To calculate x^i, where x has a floating point and i an integer value.

CALLING SEQUENCE:

```
JSB   .RTOI
DEF   x
DEF   i
```

NORMAL RETURN:

Floating point result is left in A and B registers. If error condition (below) occurs, overflow bit is set.

STORAGE:

78 words

EXECUTION TIME:

(Approximate)

Minimum = $(1.5)(\log_2 i)(.64)$ ms

Average = $(1.5)(\log_2 i)(.70)$ ms

Maximum = $(1.5)(\log_2 i)(.75)$ ms

ACCURACY:

The only possibility of inaccuracy is that introduced by roundoff in the FMP or the FDV routine if $i < 0$.

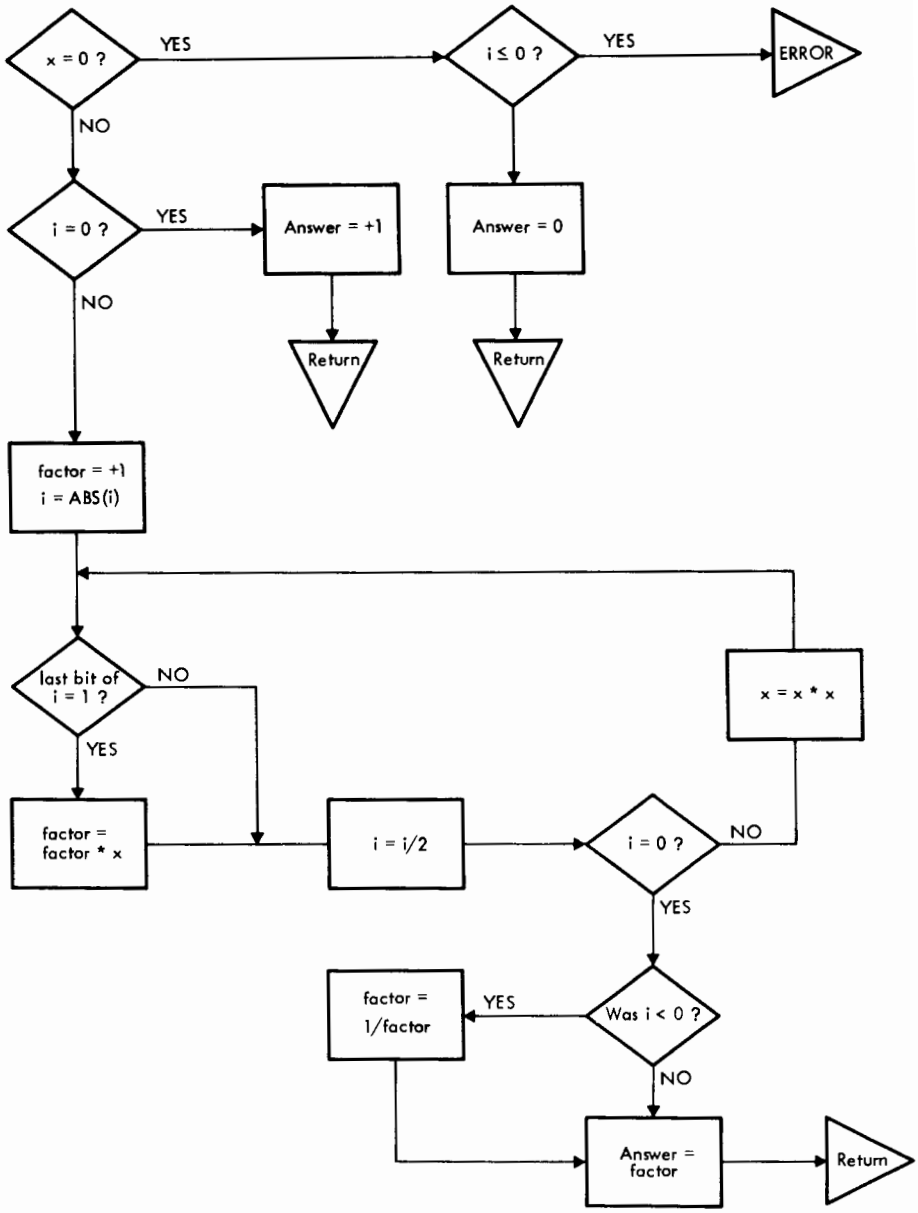
x^i gives the same result as the expression:

$$\underbrace{x*x*x*...*x}_{i \text{ times}} \quad \text{or} \quad 1/\underbrace{x*x*x*....*x}_{i \text{ times}}$$

ERROR CONDITIONS:

Condition	Error Code
$x=0, \quad i \leq 0$	Ø6 UN
$x^{\lvert i \rvert} > 2^{128}$	(floating point overflow)

ALGORITHM:


x = 0 ?
YES
i ≤ 0 ?
YES
ERROR
NO
i = 0 ?
YES
Answer = +1
Answer = 0
NO
Return
Return
factor = +1
i = ABS(i)
last bit of
i = 1 ?
NO
YES
x = x * x
factor =
factor * x
i = i/2
i = 0 ?
NO
YES
factor =
1/factor
YES
Was i < 0 ?
NO
Answer =
factor
Return

.RTOR

PURPOSE: To calculate x^y, where x and y have floating point values.

CALLING SEQUENCE:
```
JSB  .RTOR
DEF  x
DEF  y
```

NORMAL RETURN: Floating point result is left in A and B registers. If error condition (below) occurs, the overflow bit is set.

STORAGE: 45 words

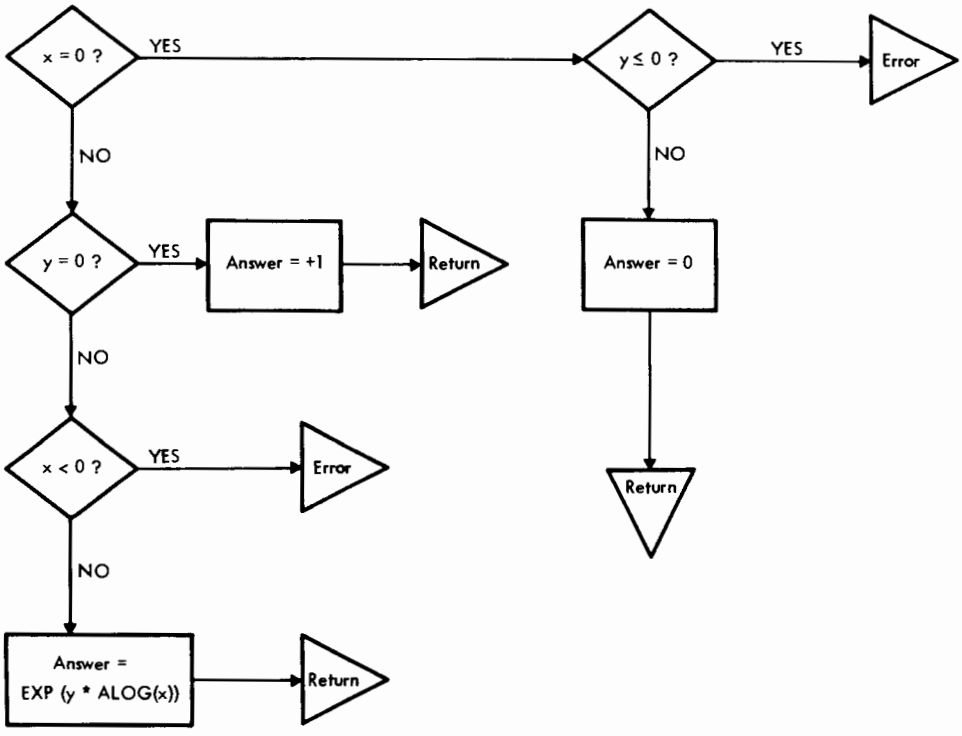
EXECUTION TIME:
Minimum = 12.24 ms
Average = 15.4 ms
Maximum = 21.3 ms

ACCURACY: 22 bits

ERROR CONDITIONS:

Condition	Error Code
$x < 0,\ y \leq 0$ $x = 0,\ y \neq 0$	Ø4 UN
$\lvert Y*ALOG(X)\rvert \geq 124$	Ø7 OF

ALGORITHM:



CHAPTER 3
SERVICE ROUTINES

PURPOSE:

To evaluate the Chebyshev series represented by the table of coefficients at the argument x; where x has a floating point value.

CALLING SEQUENCE:

```
DLD   x
JSB   CHEBY
DEF   c      (c=starting address of table of floating
             point Chebyshev coefficients; the table
             is terminated by a zero word.)
```

NORMAL RETURN:

Floating point result is left in A and B registers.

STORAGE:

57 words

EXECUTION TIME:

Minimum = n(1.24) ms
Average = n(1.7) ms
Maximum = n(2.55) ms

(where n=number of Chebyshev coefficients in table.)

ACCURACY:

22 bits

ERROR CONDITIONS:

Possibility of floating point overflow in the case of TAN(x) when x is very close to a multiple of $\pi/2$.

COM

PURPOSE: To calculate the algebraic negative of x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD   x
JSB   COM
```

NORMAL RETURN: Floating point result is left in A and B registers.

STORAGE: 9 words

EXECUTION TIME:

Minimum = .3 ms

Average = .5 ms

Maximum = .9 ms

ACCURACY: 23 bits

PURPOSE: To write an error message, supplied by the calling routine, on the Standard Teleprinter Output device.

CALLING SEQUENCE:

```
JSB   .ERRR
ASC   1,xx    (xx = 2 digit code identifying calling routine)
ASC   1,yy    (yy = 2 letter code identifying type of error)
```

NORMAL RETURN: Message is printed, A and B registers are destroyed.

STORAGE: 21 words

EXPO

PURPOSE:	To extract exponent part of x, where x has a floating point value.
CALLING SEQUENCE:	DLD x JSB EXPO
NORMAL RETURN:	Integer result is left in A register.
EXECUTION TIME:	.012 ms
STORAGE:	8 words
ACCURACY:	23 bits

PURPOSE:	To "unpack" a floating point number.
CALLING SEQUENCE:	JSB .FLUN
NORMAL RETURN:	Result is left with exponent in A register, and lower part of mantissa in B register.
STORAGE:	13 words
EXECUTION TIME:	.03 ms

.IENT

PURPOSE:

To calculate ENTIER(x); the greatest integer not algebraically exceeding x, where x has a floating point value.

CALLING SEQUENCE:

```
DLD  x
JSB  .IENT
JMP  errtn   (address of exit to be taken if
              EXPO(x) > 14)
```

NORMAL RETURN:

Integer result is left in A register.

EXECUTION TIME:

Minimum = .22 ms
Average = .38 ms
Maximum = .54 ms

STORAGE:

29 words

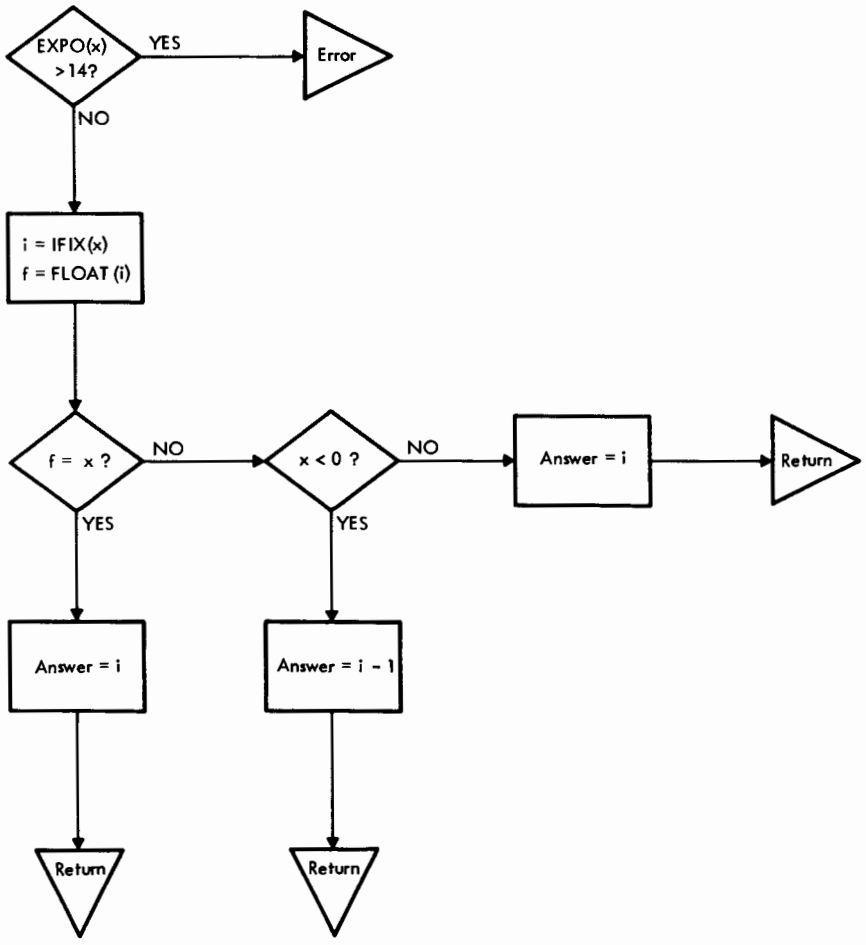
ACCURACY:

Exact

ERROR CONDITIONS:

Condition	Error Code
EXPO(x) > 14	(depends on error routine)

ALGORITHM:



PURPOSE:	To extract mantissa part of x, where x has a floating point value. Thus x = MANT(x)*2**EXPO(x).
CALLING SEQUENCE:	DLD x JSB MANT
NORMAL RETURN:	Floating point result is left in A and B registers.
EXECUTION TIME:	.023 ms
STORAGE:	9 words
ACCURACY:	23 bits

.PACK

PURPOSE: To transform the signed 31-bit mantissa contained in A and B registers to normalized floating point format.

CALLING SEQUENCE:

```
JSB    .PACK
BSS    1        (Contains exponent portion of floating
                 point number)
```

NORMAL RETURN: Floating point result is left in A and B registers.

STORAGE: 42 words

EXECUTION TIME: .1 to .15 ms

PURPOSE: To calculate $x\,(2^n)$, where x has a floating point value and n is an integer.

CALLING SEQUENCE:

```
DLD   x
JSB   PWR2
DEC   n
```


NORMAL RETURN: Floating point result is left in A and B registers.

EXECUTION TIME: .075 ms

STORAGE: 20 words

ALGORITHM: Exponent of x is increased by n.

ACCURACY: 23 bits

ERROR CONDITIONS: None. Calling routine should check for possibility of overflow.

APPENDIXES

SUMMARY OF ERROR CONDITIONS AND CODES A

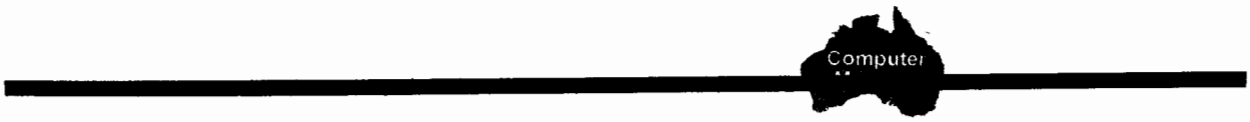
Error codes are produced on the Standard Teleprinter Output device at object program execution time; the general format is:

xx yy where xx is a two-digit code identifying the routine issuing the code, and yy is a two-letter mnemonic identifying the type of error. UN = underflow, OF = overflow, OR = out of range.

Error Code	Subroutine	Condition
02 UN	ALOG	$x \leq 0$
03 UN	SQRT	$x < 0$
04 UN	.RTOR	$x = 0, y \leq 0$ $x < 0, y \neq 0$
05 OR	SIN COS	$ABS(x) > 2^{14}$
06 UN	.RTOI	$x = 0, i \leq 0$
07 OF	.RTOR	$x^{\|y\|}$ out of range
	EXP	$ABS(x)*\log_2 e > 124$
08 OF	.ITOI	i^j out of range
08 UN	.ITOI	$i = 0, j \leq 0$
09 OR	TAN	$x > 2^{14}$

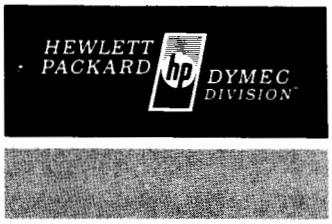
Overflow may be caused in the TAN(x) routine when tan(x) is out of range; in the .RTOI routine when x**ABS(i) is out of range.

MISCELLANEOUS ROUTINES B



The following routines are also contained in the FORTRAN library.

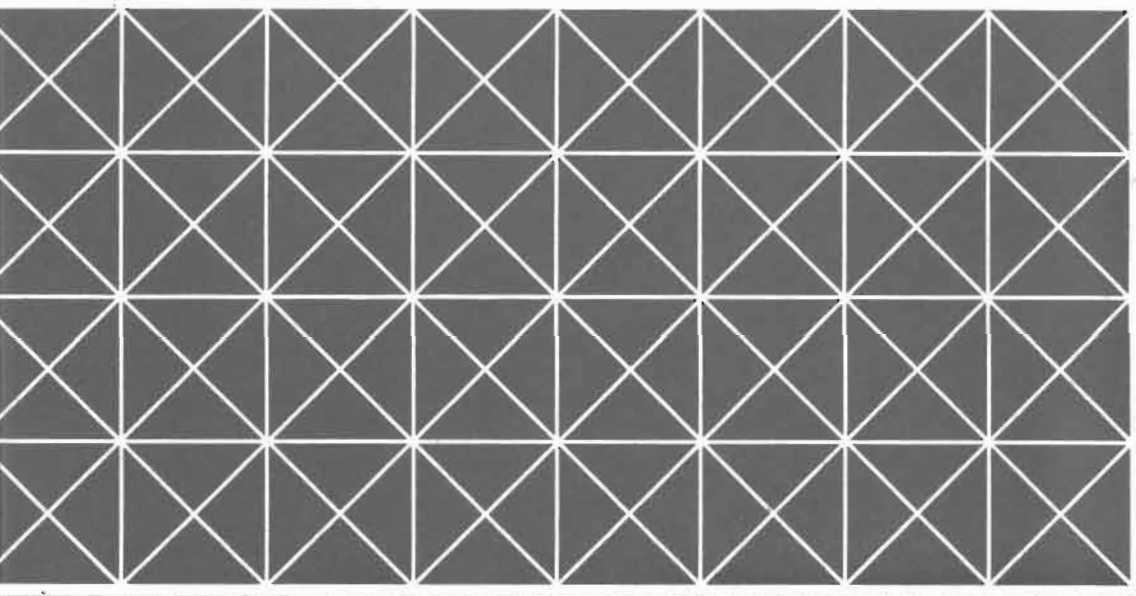
..DLC(x)	Loads and complements the floating point quantity x.
.ENTR	Transfers the parameters of a FORTRAN function to the FORTRAN library routine being entered.
..FCM	Complements the floating point number in the A and B registers.
FRMTR	This routine handles all I/O generated by a FORTRAN program.
.GOTO	Performs translation of a FORTRAN GO TO statement.
IAND(i,j)	Takes the logical product of i and j and leaves result in the A register.
IEOF(u)	Returns true value (sign bit = 1) in A if end-of-file has been reached on unit u; false (sign bit = 0) if not.
IOR(i,j)	Takes logical inclusive or of i and j and returns result in the A register.
.MAP.	This routine obtains the address of an element of a two-dimensional array.
OVF	Returns true value (sign bit = 1) in A if overflow bit is set; false (sign bit = 0) if not.
.PAUS(i)	This routine prints PAUSE on the Standard Teleprinter Output device and stops computer processing; i is displayed in the A register. Press RUN to continue.
.STOP(i)	Prints STOP on the teleprinter and stops computer processing; i is displayed in the A register. Processing is not resumed.

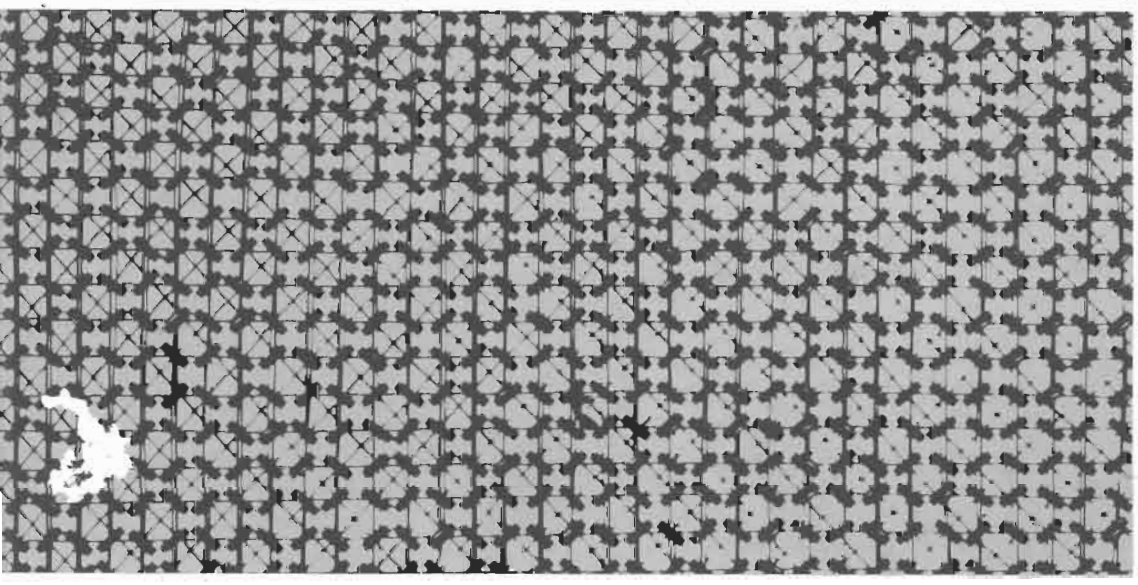


395 Page Mill Road, Palo Alto, California 94306 Area Code 415 326-1755

MARCH 1967

PRINTED IN U.S.A.





HEWLETT PACKARD hp DYMEC DIVISION

02116-9032

PRINTED IN U.S.A.