VMS RTL Mathematics (MTH$) Manual

Order Number: AA–LA72A–TE

April 1988

This manual documents the mathematics routines contained in the MTH$ facility of the VMS Run-Time Library.

Revision/Update Information: This document supersedes Section 4 and the MTH$ portion of Part II of the *VAX/VMS Run-Time Library Routines Reference Manual,* Version 4.4.

Software Version: VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

ZK4610

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital§ local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

Preface

This manual provides users of the VMS operating system with detailed usage and reference information on mathematics routines supplied in the MTH$ facility of the Run-Time Library.

Run-Time Library routines can only be used in programs written in languages that produce native code for the VAX hardware. At present, these languages include VAX MACRO and the following compiled high-level languages:

VAX Ada
VAX BASIC
VAX BLISS-32
VAX C
VAX COBOL
VAX COBOL-74
VAX CORAL
VAX DIBOL
VAX FORTRAN
VAX Pascal
VAX PL/I
VAX RPG
VAX SCAN

Interpreted languages which can also access Run-Time Library routines include VAX DSM and DATATRIEVE.

Intended Audience

This manual is intended for system and application programmers who want to call Run-Time Library routines.

Document Structure

This manual is organized into two parts as follows:

- The introductory chapters provide guidelines on using the MTH$ mathematics routines.
- The MTH$ Reference Section provides detailed reference information on each mathematics routine contained in the MTH$ facility of the Run-Time Library. This information is presented using the documentation format described in the *Introduction to the VMS Run-Time Library*. Routine descriptions appear in alphabetical order by routine name.

Associated Documents

The Run-Time Library routines are documented in a series of reference manuals. A general overview of the Run-Time Library and a description of how the Run-Time Library routines are accessed is presented in the *Introduction to the VMS Run-Time Library.* Descriptions of the other RTL facilities and their corresponding routines and usages are discussed in the following books:

- The *VMS RTL DECtalk (DTK$) Manual*
- The *VMS RTL Library (LIB$) Manual*
- The *VMS RTL General Purpose (OTS$) Manual*
- The *VMS RTL Parallel Processing (PPL$) Manual*
- The *VMS RTL Screen Management (SMG$) Manual*
- The *VMS RTL String Manipulation (STR$) Manual*

The VAX Procedure Calling and Condition Handling Standard, which is documented in the *Introduction to System Routines,* contains useful information for anyone who wants to call Run-Time Library routines.

Applications programmers of any language may refer to the *Guide to Creating VMS Modular Procedures* for the Modular Programming Standard and other guidelines.

High-level language programmers will find additional information on calling Run-Time Library routines in their language reference manual. Additional information may also be found in the language user's guide provided with your VAX language.

The *Guide to Using VMS Command Procedures* may also be useful.

For a complete list and description of the manuals in the VMS documentation set, see the *Overview of VMS Documentation.*

Conventions

Convention	Meaning
$\boxed{\text{RET}}$	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

Other conventions used in the documentation of Run-Time Library routines are described in the *Introduction to the VMS Run-Time Library*.

1 Introduction to MTH$

The Run-Time Library mathematics routines may be called to perform a wide variety of computations including the following:

- Complex exponentiation
- Complex function evaluation
- Exponentiation
- Floating-point trigonometric function evaluation
- Miscellaneous function evaluation

The OTS$ facility provides additional language-independent arithmetic support routines.

This introduction to Run-Time Library mathematics routines includes examples of how to call mathematics routines from BASIC, COBOL, FORTRAN, MACRO, PASCAL, and PL/I.

1.1 Entry Point Names

The names of the mathematics routines are formed by adding the MTH$ prefix to the function names.

When function arguments and returned values are of the same data type, the first letter of the name indicates this data type. When function arguments and returned values are of different data types, the first letter indicates the data type of the returned value, and the second letter indicates the data type of the argument(s).

The letters used as data type prefixes are listed below.

Letter	Data Type
I	Word
J	Longword
D	D_floating
G	G_floating
H	H_floating
C	F_floating complex
CD	D_floating complex
CG	G_floating complex

Generally, F-floating data types have no letter designation. For example, MTH$SIN returns an F-floating value of the sine of an F-floating argument and MTH$DSIN returns a D-floating value of the sine of a D-floating argument. However, in some of the miscellaneous functions, F-floating data types are referenced by the letter designation A.

1.2 Calling Conventions

All calls to mathematics routines, as described in the FORMAT section of each routine, accept arguments passed by reference. JSB entry points accept arguments passed by value.

All mathematics routines return values in R0 or R0/R1 except those routines for which the values cannot fit in 64 bits. D-floating complex, G-floating complex and H-floating values are data structures which are larger than 64 bits. Routines that return values which cannot fit in registers R0/R1 return their function values into the first argument in the argument list.

The notation JSB MTH$NAME_Rn, where *n* is the highest register number referenced, indicates that an equivalent JSB entry point is available. No registers are saved; only registers R0:Rn are changed.

Routines with JSB entry points accept a single argument in R0:Rm, where *m*, which is defined below, is dependent on the data type.

Data Type	m
F_floating	0
D_floating	1
G_floating	1
H_floating	3

A routine which returns one value returns it to registers R0:Rm.

When a routine returns two values, for example MTH$SINCOS, the first value is returned in R0:Rm and the second value is returned in (R <m+1> :R <2*m+1>).

Note that for routines that return a single value, n> =m. For routines that return two values, n> =2*m + 1.

All CALL entry points for mathematics routines do the following:

- Disable floating-point underflow
- Enable integer overflow
- Cause no floating-point overflow or other arithmetic traps or faults
- Preserve all other enabled operations across the CALL

JSB entry points execute in the context of the caller with the enable operations as set by the caller. Since the routines do not cause arithmetic traps or faults, their operation is not affected by the setting of the arithmetic trap enables, except as noted.

For more detailed information on CALL and JSB entry points, refer to the *Introduction to the VMS Run-Time Library.*

1.3 Algorithms

For those mathematics routines that have corresponding algorithms, the complete algorithm can be found in the Description section of the routine description appearing in the MTH$ Reference Section of this manual.

1.4 Condition Handling

Error conditions are indicated by using the VAX signaling mechanism. The VAX signaling mechanism signals all conditions in mathematics routines as SEVERE by calling LIB$SIGNAL. When a SEVERE error is signaled, the image is caused to exit after printing an error message. A user-established condition handler can be written to cause execution to continue at the point of the error by returning SS$_CONTINUE. A mathematics routine returns to its caller after the contents of R0/R1 have been restored from the mechanism argument vector CHF$L_MCH_SAVR0/R1. Thus, the user-established handler should correct CHF$L_MCH_SAVR0/R1 to the desired function value to be returned to the caller of the mathematics routine.

D-floating complex, G-floating complex, and H-floating values cannot be corrected with a user-established condition handler, because R2/R3 are not available in the mechanism argument vector.

Note that it is more reliable to correct R0 and R1 to resemble R0 and R1 of a double-precision floating-point value. A double-precision floating-point value correction works for both single- and double-precision values. If the correction is not performed, the floating-point reserved operand –0.0 is returned. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Accessing the floating-point reserved operand will cause a reserved operand fault. See the *VMS RTL Library (LIB$) Manual* for a complete description of how to write user condition handlers for SEVERE errors.

A few mathematics routines signal floating underflow if the calling program (JSB or CALL) has enabled floating underflow faults or traps.

All mathematics routines access input arguments and the real and imaginary parts of complex numbers using floating-point instructions. Therefore, a reserved operand fault can occur in any mathematics routine.

1.5 Complex Numbers

A complex number y is defined as an ordered pair of real numbers r and i, where r is the real part and i is the imaginary part of the complex number.

$$y=(r,i)$$

VMS supports three floating-point complex types: F-floating complex, D-floating complex, and G-floating complex. There is no H-floating complex data type.

Run-Time Library mathematics routines that use complex arguments require two x-floating values to be passed by reference for each argument. The first x-floating value contains r, the real part of the complex number. The second x-floating value contains i, the imaginary part of the complex number. Similarly, Run-Time Library mathematics routines that return complex function values return two x-floating values. Some Language Independent Support (OTS$) routines also calculate complex functions.

Note that complex functions have no JSB entry points.

1.6 Routines Not Documented in the MTH$ Reference Section

The mathematics routines in Table 1–1 are not found in the reference section of this manual. Instead, their entry points and argument information are listed in Appendix A of this manual.

A reserved operand fault can occur for any floating-point input argument in any mathematics routine. Other condition values signaled by each mathematics routine are indicated in the footnotes.

Table 1–1 Additional Mathematics Routines

Entry Point	Function
Absolute Value Routines	
MTH$ABS	F-floating absolute value
MTH$DABS	D-floating absolute value
MTH$GABS	G-floating absolute value
MTH$HABS	H-floating absolute value[1]
MTH$IIABS	Word absolute value[2]
MTH$JIABS	Longword absolute value[2]
Bitwise AND Operator Routines	
MTH$IIAND	Bitwise AND of two word arguments
MTH$JIAND	Bitwise AND of two longword arguments
F-floating Conversion Routines	
MTH$DBLE	Convert F-floating to D-floating (exact)
MTH$GDBLE	Convert F-floating to G-floating (exact)
MTH$IIFIX	Convert F-floating to word (truncated)[2]
MTH$JIFIX	Convert F-floating to longword (truncated)[2]

[1]Returns value to the first argument; value exceeds 64 bits.

[2]Integer overflow exceptions can occur.

Table 1–1 (Cont.) Additional Mathematics Routines

Entry Point	Function
Floating-Point Positive Difference Routines	
MTH$DIM	Positive difference of two F-floating arguments[3,4]
MTH$DDIM	Positive difference of two D-floating arguments[3,4]
MTH$GDIM	Positive difference of two G-floating arguments[3,4]
MTH$HDIM	Positive difference of two H-floating arguments[1,3,4]
MTH$IIDIM	Positive difference of two word arguments[2]
MTH$JIDIM	Positive difference of two longword arguments[2]
Bitwise Exclusive OR Operator Routines	
MTH$IIEOR	Bitwise exclusive OR of two word arguments
MTH$JIEOR	Bitwise exclusive OR of two longword arguments
Integer to Floating-point Conversion Routines	
MTH$FLOATI	Convert word to F-floating (exact)
MTH$DFLOTI	Convert word to D-floating (exact)
MTH$GFLOTI	Convert word to G-floating (exact)
MTH$FLOATJ	Convert longword to F-floating (exact)
MTH$DFLOTJ	Convert word to D-floating (exact)
MTH$GFLOTJ	Convert longword to G-floating (exact)
Conversion to Greatest Floating-point Integer Routines	
MTH$FLOOR	Convert F-floating to greatest F-floating integer
MTH$DFLOOR	Convert D-floating to greatest D-floating integer
MTH$GFLOOR	Convert G-floating to greatest G-floating integer
MTH$HFLOOR	Convert H-floating to greatest H-floating integer[1]

[1]Returns value to the first argument; value exceeds 64 bits.

[2]Integer overflow exceptions can occur.

[3]Floating-point overflow exceptions can occur.

[4]Floating-point underflow exceptions can occur.

Table 1–1 (Cont.) Additional Mathematics Routines

Entry Point	Function
Floating-point Truncation Routines	
MTH$AINT	Convert F-floating to truncated F-floating[3]
MTH$DINT	Convert D-floating to truncated D-floating
MTH$IIDINT	Convert D-floating to truncated word[2]
MTH$JIDINT	Convert D-floating to truncated longword[2]
MTH$GINT	Convert G-floating to truncated G-floating
MTH$IIGINT	Convert G-floating to truncated word[2]
MTH$JIGINT	Convert G-floating to truncated longword[2]
MTH$HINT	Convert H-floating to truncated H-floating[1,3]
MTH$IIHINT	Convert H-floating to truncated word[2]
MTH$JIHINT	Convert H-floating to truncated longword[2]
MTH$IINT	Convert F-floating to truncated word[2]
MTH$JINT	Convert F-floating to truncated longword[2]
Bitwise Inclusive OR Operator Routines	
MTH$IIOR	Bitwise inclusive OR of two word arguments
MTH$JIOR	Bitwise inclusive OR of two longword arguments
Maximum Value Routines	
MTH$AIMAX0	F-floating maximum of n word arguments
MTH$AJMAX0	F-floating maximum of n longword arguments
MTH$IMAX0	Word maximum of n word arguments
MTH$JMAX0	Longword maximum of n longword arguments
MTH$AMAX1	F-floating maximum of n F-floating arguments[2]
MTH$DMAX1	D-floating maximum of n D-floating arguments
MTH$GMAX1	G-floating maximum of n G-floating arguments
MTH$HMAX1	H-floating maximum of n H-floating arguments[1]
MTH$IMAX1	Word maximum of n F-floating arguments[2]
MTH$JMAX1	Longword maximum of n F-floating arguments[2]

[1]Returns value to the first argument; value exceeds 64 bits.

[2]Integer overflow exceptions can occur.

[3]Floating-point overflow exceptions can occur.

Table 1–1 (Cont.) Additional Mathematics Routines

Entry Point	Function
Minimum Value Routines	
MTH$AIMINO	F-floating minimum of n word arguments
MTH$AJMINO	F-floating minimum of n longword arguments
MTH$IMINO	Word minimum of n word arguments
MTH$JMINO	Longword minimum of n longword arguments
MTH$AMIN1	F-floating minimum of n F-floating arguments[2]
MTH$DMIN1	D-floating minimum of n D-floating arguments
MTH$GMIN1	G-floating minimum of n G-floating arguments
MTH$HMIN1	H-floating minimum of n H-floating arguments[1]
MTH$IMIN1	Word minimum of n F-floating arguments[2]
MTH$JMIN1	Longword minimum of n F-floating arguments[2]
Remainder Routines	
MTH$AMOD	Remainder of two F-floating arguments, arg1/arg2[3]
MTH$DMOD	Remainder of two D-floating arguments, arg1/arg2[3]
MTH$GMOD	Remainder of two G-floating arguments, arg1/arg2[3]
MTH$HMOD	Remainder of two H-floating arguments, arg1/arg2[1,3]
MTH$IMOD	Remainder of two word arguments, arg1/arg2[5]
MTH$JMOD	Remainder of two longword arguments, arg1/arg2[5]
Floating-point Conversion to Nearest Value Routines	
MTH$ANINT	Convert F-floating to nearest F-floating integer
MTH$DNINT	Convert D-floating to nearest D-floating integer[3]
MTH$IIDNNT	Convert D-floating to nearest word integer
MTH$JIDNNT	Convert D-floating to nearest longword integer
MTH$GNINT	Convert G-floating to nearest G-floating integer[3]
MTH$IIGNNT	Convert G-floating to nearest word integer[2]
MTH$JIGNNT	Convert G-floating to nearest longword integer[2]
MTH$HNINT	Convert H-floating to nearest H-floating integer[1]

[1]Returns value to the first argument; value exceeds 64 bits.

[2]Integer overflow exceptions can occur.

[3]Floating-point overflow exceptions can occur.

[5]Divide-by-zero exceptions can occur.

Table 1–1 (Cont.) Additional Mathematics Routines

Entry Point	Function
MTH$IIHNNT	Convert H-floating to nearest word integer[2]
MTH$JIHNNT	Convert H-floating to nearest longword integer[2]
MTH$ININT	Convert F-floating to nearest word integer[2]
MTH$JNINT	Convert F-floating to nearest longword integer[3,6]
Bitwise Complement Operator Routines	
MTH$INOT	Bitwise complement of word argument
MTH$JNOT	Bitwise complement of longword argument
Floating-point Multiplication Routines	
MTH$DPROD	D-floating product of two F-floating arguments[3]
MTH$GPROD	G-floating product of two F-floating arguments[3]
Bitwise Shift Operator Routines	
MTH$IISHFT	Bitwise shift of word
MTH$JISHFT	Bitwise shift of longword
Floating-point Sign Function Routines	
MTH$SGN	F- or D-floating sign function
MTH$SIGN	F-floating transfer of sign of y to sign of x
MTH$DSIGN	D-floating transfer of sign of y to sign of x
MTH$GSIGN	G-floating transfer of sign of y to sign of x
MTH$HSIGN	H-floating transfer of sign of y to sign of x[1]

[1]Returns value to the first argument; value exceeds 64 bits.

[2]Integer overflow exceptions can occur.

[3]Floating-point overflow exceptions can occur.

[6]Returns contents of R0 if a negative argument is input.

Table 1–1 (Cont.) Additional Mathematics Routines

Entry Point	Function
MTH$IISIGN	Word transfer of sign of y to sign of x
MTH$JISIGN	Longword transfer of sign of y to sign of x
Conversion of Double to Single Floating-point Routines	
MTH$SNGL	Convert D-floating to F-floating (rounded)[3]
MTH$SNGLG	Convert G-floating to F-floating (rounded)[3,4]

[3]Floating-point overflow exceptions can occur.

[4]Floating-point underflow exceptions can occur.

1.7 Examples of Calls to Run-Time Library Mathematics Routines

1.7.1 BASIC Example

The following BASIC program uses the H-floating data type. BASIC also supports the D-floating, F-floating and G-floating data types, but does not support the complex data types.

```
10      !+
        ! Sample program to demonstrate a call to MTH$HEXP from BASIC.
        !-

        EXTERNAL SUB MTH$HEXP ( HFLOAT, HFLOAT )

        DECLARE HFLOAT X,Y      ! X and Y are H-floating
        DIGITS$ = '###.####################################'
        X = '1.2345678901234567891234567892'H
        CALL MTH$HEXP (Y,X)
        A$ = 'MTH$HEXP of ' + DIGITS$ + ' is ' + DIGITS$
        PRINT USING A$, X, Y
        END
```

The output from this program is as follows:

```
MTH$HEXP of  1.234567890123456789123456789200000
is 3.436893084346008004973301321342110
```

1.7.2 COBOL Example

The following COBOL program uses the F-floating and D-floating data types. COBOL does not support the G-floating and H-floating data types or the complex data types.

This COBOL program calls MTH$EXP and MTH$DEXP.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    FLOATING_POINT.
*
*  Calls MTH$EXP using a Floating Point data type.
*  Calls MTH$DEXP using a Double Floating Point data type.
*
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FLOAT_PT     COMP-1.
01 ANSWER_F     COMP-1.
01 DOUBLE_PT    COMP-2.
01 ANSWER_D     COMP-2.
PROCEDURE DIVISION.
P0.
        MOVE 12.34 TO FLOAT_PT.
        MOVE 3.456 TO DOUBLE_PT.

        CALL "MTH$EXP" USING BY REFERENCE FLOAT_PT GIVING ANSWER_F.
        DISPLAY " MTH$EXP of ", FLOAT_PT CONVERSION, " is ",
                                               ANSWER_F CONVERSION.

        CALL "MTH$DEXP" USING BY REFERENCE DOUBLE_PT GIVING ANSWER_D.
        DISPLAY " MTH$DEXP of ", DOUBLE_PT CONVERSION, " is ",
                                               ANSWER_D CONVERSION .
        STOP RUN.
```

The output from this example program is as follows:

```
MTH$EXP of  1.234000E+01 is  2.286620E+05
MTH$DEXP of  3.456000000000000E+00 is
3.168996280537917E+01
```

1.7.3 FORTRAN Examples

The first two FORTRAN programs below use the D-floating and H-floating data types. The third FORTRAN program below uses the F-floating complex data type. FORTRAN supports the four floating data types and the three complex data types.

1

```
C+
C This FORTRAN program computes exp(x) in
C double precision by using the RTL routine " MTH$DEXP x ".
C
C Declare X,Y and MTH$DEXP as double precision values.
C MTH$DEXP(X) will return a double precision value to variable Y.
C-
        REAL*8 X,Y,MTH$DEXP
        X = 3.456
        Y = MTH$DEXP(X)
        WRITE(6,1) X,Y
1       FORMAT(' ','MTH$DEXP(',F20.15,') IS ',F20.15)
        END
```

The output generated by this FORTRAN example is as follows:

```
MTH$DEXP(3.456000000000000) IS
31.689962805379165
```

2

```
C+
C This FORTRAN program computes exp(x) using
C the RTL routine MTH$HEXP. MTH$HEXP is CALLed by
C MTH$HEXP(return_value , argument)
C
C Declare X,Y as H-floating point values.
C Given X MTH$HEXP will return the value of exp(X) in Y by the call
C CALL MTH$HEXP(Y,X).
C-

        REAL*16 X,Y
        X = 1.23456789012345678912345678 92
        CALL MTH$HEXP(Y,X)
        WRITE(6,1) X,Y
1       FORMAT(' ','MTH$HEXP of ',E35.30,' is ',E35.30)
        END
```

This FORTRAN program generates the following output:

```
MTH$HEXP of .123456789012345678912345678920E+01
is .343689308434600800497330132134E+01
```

3

```
C+
C This FORTRAN program computes the complex log
C of x using the RTL routine MTH$CLOG. This program also demonstrates
C two ways the user can create a complex number.
C
C Declare Z,Z_LOG,MTH$CMPLX, and MTH$CLOG as complex values and R and I
C as real values. MTH$CMPLX takes two real arguments and returns one
C complex number: Z = MTH$CMPLX(R,I) is a complex number with "real"
C part R and "imaginary" part I.
C
C Given a complex number Z, MTH$CLOG(Z) returns the complex natural
C logarithm of Z.
C-

        COMPLEX Z,Z_LOG,MTH$CMPLX,MTH$CLOG
        REAL*4 R,I
        R = 3.142563
        I = 7.4367846
        Z = MTH$CMPLX(R,I)
C+
C Z is a complex number with real part R and imaginary part I.
C-

        TYPE *, ' The complex number z is',z
C+
C  Compute the natural logarithm of Z = (2,1).
C  Directly define the complex number Z.
C-

        Z = (2.0,1.0)
        Z_LOG = MTH$CLOG(Z)
        TYPE *,' The complex log of (2,1) is  ',Z_LOG
        END
```

The output generated by this program is as follows:

```
The complex number z is (3.142563,7.436785)
The complex log of (2,1) is
(0.8047190,0.4636476)
```

1.7.4 MACRO Examples

MACRO and BLISS support JSB entry points as well as CALLS and CALLG entry points. Both MACRO and BLISS support the four floating data types and the three complex data types.

The MACRO programs below illustrate the use of the CALLS and CALLG instructions, as well as JSB entry points.

1
```
        .TITLE  EXAMPLE_JSB
;+
;  This example calls MTH$DEXP by using a Macro JSB command.
;  The JSB command expects R0/R1 to contain the quadword input value X.
;  The result of the JSB will be located in R0/R1.
;-
        .EXTRN  MTH$DEXP_R6     ;MTH$DEXP is an external routine.
        .PSECT  DATA, PIC, EXE, NOWRT
X:      .DOUBLE 2.0             ; X is 2.0
        .ENTRY  EXAMPLE_JSB, ^M<>
        MOVQ    X, R0           ; X is in registers R0 and R1
        JSB     G^MTH$DEXP_R6   ; The result is returned in R0/R1.
        RET
        .END    EXAMPLE_JSB
```

This MACRO program generates the following output:

```
R0 <-- 732541EC
R1 <-- ED6EC6A6

That is, MTH$DEXP(2) is 7.3890560989306502
```

2
```
        .TITLE EXAMPLE_CALLG
;+
;  This example calls MTH$HEXP by using a Macro CALLG command.
;  The CALLG command expects that the address of the return value
;  Y, the address of the input value X, and the argument count 2 be
;  stored in memory; this program stores this information in ARGUMENTS.
;  The result of the CALLG will be located in R0/R1.
;-
        .EXTRN  MTH$HEXP        ; MTH$HEXP is an external routine.
        .PSECT  DATA, PIC, EXE, WRT
ARGUMENTS:
        .LONG   2               ; The CALLG will use two arguments.
        .ADDRESS Y, X           ; The first argument must be the address
                                ;  receiving the computed value, while
                                ;  the second argument is used to
                                ;  compute exp(X).
X:      .H_FLOATING 2           ; X = 2.0
Y:      .H_FLOATING 0           ; Y is the result, initially set to 0.
        .ENTRY  EXAMPLE_G, ^M<>
        CALLG   ARGUMENTS, G^MTH$HEXP ; CALLG returns the value to Y.
        RET
        .END    EXAMPLE_G
```

The output generated by this MACRO program is as follows:

```
address of Y <-- D8E64003
             <-- 4DDA4B8D
             <-- 3A3BDCC3
             <-- B68BA206

That is, MTH$HEXP of 2.0 returns
7.389056098930650227230427460575O1
```

3

```
        .TITLE EXAMPLE_CALLS
;+
;  This example calls MTH$HEXP by using the Macro CALLS command.
;  The CALLS command expects the SP to contain the H-floating address of
;  the return value, the address of the input argument X and the argument
;  count 2. The result of the CALLS will be located in registers RO-R3.
;-
        .EXTRN  MTH$HEXP        ; MTH$HEXP is an external routine.
        .PSECT  DATA, PIC, EXE, WRT
Y:      .H_FLOATING 0           ; Y is the result, initially set to 0.
X:      .H_FLOATING 2           ; X = 2
        .ENTRY  EXAMPLE_S, ^M<>
        MOVAL   X, -(SP)        ; The address of X is in the SP.
        MOVAL   Y, -(SP)        ; The address of Y is in the SP
        CALLS   Y, G^MTH$HEXP   ; The value is returned to the address of Y.
        RET
        .END    EXAMPLE_S
```

The output generated by this program is as follows:

```
address of Y <-- D8E64003
             <-- 4DDA4B8D
             <-- 3A3BDCC3
             <-- B68BA206

That is, MTH$HEXP of 2.0 returns
7.389056098930650227230427460575O1
```

4

```
        .TITLE  COMPLEX_EX1
;+
;   This example calls MTH$CLOG by using a MACRO CALLG command.
;   To compute the complex natural logarithm of Z = (2.0,1.0) register
;   RO is loaded with 2.0, the real part of Z, and register R1 is loaded
;   with 1.0, the imaginary part of Z. The CALLG to MTH$CLOG
;   returns the value of the natural logarithm of Z in
;   registers RO and R1. RO gets the real part of Z and R1
;   gets the imaginary part.
;-
        .EXTRN  MTH$CLOG
        .PSECT  DATA, PIC, EXE, NOWRT
ARGS:   .LONG   1               ; The CALLG will use one argument.
        .ADDRESS REAL           ; The one argument that the CALLG
                                ;  uses is the address of the argument
                                ;  of MTH$CLOG.
REAL:   .FLOAT  2               ; real part of Z is 2.0
IMAG:   .FLOAT  1               ; imaginary part Z is 1.0
        .ENTRY  COMPLEX_EX1, ^M<>
        CALLG   ARGS, G^MTH$CLOG; MTH$CLOG return the real part of the
                                ;  complex natural logarithm in RO and
    ;  the imaginary part in R1.
        RET
        .END    COMPLEX_EX1
```

This program generates the following output:

```
R0 <---  0210404E
R1 <---  63383FED

That is, MTH$CLOG(2.0,1.0) is
(0.8047190,0.4636476)
```

5

```
        .TITLE  COMPLEX_EX2
;+
;   This example calls MTH$CLOG by using a MACRO CALLS command.
;   To compute the complex natural logarithm of Z = (2.0,1.0) register
;   R0 is loaded with 2.0, the real part of Z, and register R1 is loaded
;   with 1.0, the imaginary part of Z.  The CALLS to MTH$CLOG
;   returns the value of the natural logarithm of Z in registers R0
;   and R1. R0 gets the real part of Z and R1 gets the imaginary
;   part.
;-
        .EXTRN  MTH$CLOG
        .PSECT  DATA, PIC, EXE, NOWRT
REAL:   .FLOAT  2               ; real part of Z is 2.0
IMAG:   .FLOAT  1               ; imaginary part Z is 1.0
        .ENTRY  COMPLEX_EX2, ^M<>
        MOVAL   REAL, -(SP)     ; SP <-- address of Z. Real part of Z is
                                ;  in @(SP) and imaginary part is in
        CALLS   #1, G^MTH$CLOG  ;  @(SP)+4.
                                ; MTH$CLOG return the real part of the
                                ;  complex natural logarithm in R0 and
                                ;  the imaginary part in R1.
        RET
        .END    COMPLEX_EX2
```

This MACRO example program generates the following output:

```
R0 <---  0210404E
R1 <---  63383FED

That is, MTH$CLOG(2.0,1.0) is
(0.8047190,0.4636476)
```

1.7.5 PASCAL Examples

The following PASCAL programs use the D-floating and H-floating data types. PASCAL also supports the F-floating and G-floating data types. PASCAL does not support the complex data types, however.

1

```
{+}
{ Sample program to demonstrate a call to MTH$DEXP from PASCAL.
{-}

PROGRAM CALL_MTH$DEXP (OUTPUT);

{+}
{ Declare variables used by this program.
{-}

VAR
    X : DOUBLE := 3.456;        { X,Y are D-floating unless overridden }
    Y : DOUBLE;                 { with /DOUBLE qualifier on compilation }
```

```
{+}
{ Declare the RTL routine used by this program.
{-}
[EXTERNAL,ASYNCHRONOUS] FUNCTION MTH$DEXP (VAR value : DOUBLE) : DOUBLE; EXTERN;

BEGIN
    Y := MTH$DEXP (x);
    WRITELN ('MTH$DEXP of ', X:5:3, ' is ', Y:20:16);
END.
```

The output generated by this PASCAL program is as follows:

```
MTH$DEXP of 3.456 is  31.6899656462382318
```

2

```
{+}
{ Sample program to demonstrate a call to MTH$HEXP from PASCAL.
{-}
PROGRAM CALL_MTH$HEXP (OUTPUT);

{+}
{ Declare variables used by this program.
{-}
VAR
    X : QUADRUPLE := 1.2345678901234567891234567892; { X is H-floating }
    Y : QUADRUPLE;                                    { Y is H-floating }

{+}
{ Declare the RTL routine used by this program.
{-}
[EXTERNAL,ASYNCHRONOUS] PROCEDURE MTH$HEXP (VAR h_exp : QUADRUPLE;
value : QUADRUPLE); EXTERN;

BEGIN
    MTH$HEXP (Y,X);
    WRITELN ('MTH$HEXP of ', X:30:28, ' is ', Y:35:33);
END.
```

This PASCAL program generates the following output:

```
MTH$DEXP of 3.456 is  31.6899656462382318
```

1.7.6 PL/I Examples

The following PL/I programs use the D-floating and H-floating data types to test entry points. PL/I also supports the F-floating and G-floating data types. PL/I does not support the complex data types, however.

1
```
/*
*                                                              *
*       This program tests a MTH$D entry point                 *
*                                                              *
*/
TEST:   PROC OPTIONS (MAIN) ;

        DCL (MTH$DEXP)
                ENTRY (FLOAT(53)) RETURNS (FLOAT(53));
        DCL OPERAND FLOAT(53);
        DCL RESULT FLOAT(53);

/*** Begin test ***/
        OPERAND = 3.456;
        RESULT = MTH$DEXP(OPERAND);
        PUT EDIT ('MTH$DEXP of ', OPERAND, ' is ', RESULT)(A(12),F(5,3),A(4),F(20,15));

END TEST;
```

The output generated by this PL/I program is as follows:

```
MTH$DEXP of 3.456 is   31.689962805379165
```

2
```
/*
*                                                              *
*       This program tests a MTH$H entry point.                *
*       Note that in the PL/I statement below, the /G-float switch    *
*       is needed to compile both G- and H-floating point MTH$ routines.    */
TEST:   PROC OPTIONS (MAIN) ;

        DCL (MTH$HEXP)
                ENTRY (FLOAT (113), FLOAT (113)) ;
        DCL OPERAND FLOAT (113);
        DCL RESULT FLOAT (113);

/*** Begin test ***/
        OPERAND = 1.2345789012345678912345678 92;
        CALL MTH$HEXP(RESULT,OPERAND);
        PUT EDIT ('MTH$HEXP of ', OPERAND, ' is ', RESULT) (A(12),F(29,27),A(4),F(29,27));

end test;
```

To run this program, you must use the following DCL commands:

```
$ PLI/G_FLOAT EXAMPLE
$ LINK EXAMPLE
$ RUN EXAMPLE
```

This program generates the following output:

```
MTH$HEXP of 1.234578901234567891234567892 is
3.436930928565989790506225633
```

MTH$ Reference Section

This section provides detailed descriptions of the routines provided by the VMS RTL Mathematics (MTH$) Facility.

MTH$xACOS Arc Cosine of Angle Expressed in Radians

Given the cosine of an angle, the Arc Cosine of Angle Expressed in Radians routine returns that angle (in radians).

FORMAT

MTH$ACOS *cosine*
MTH$DACOS *cosine*
MTH$GACOS *cosine*

Each of the above three formats accepts as input one of the floating-point types.

jsb entries

MTH$ACOS_R4
MTH$DACOS_R7
MTH$GACOS_R7

Each of the above three JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in radians. The angle returned will have a value in the range

$$0 \leq angle \leq \pi$$

MTH$ACOS returns an F-floating number. MTH$DACOS returns a D-floating number. MTH$GACOS returns a G-floating number.

ARGUMENTS

cosine
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The cosine of the angle whose value (in radians) is to be returned. The **cosine** argument is the address of a floating-point number that is this cosine. The absolute value of **cosine** must be less than or equal to 1. For MTH$ACOS, **cosine** specifies an F-floating number. For MTH$DACOS, **cosine** specifies a D-floating number. For MTH$GACOS, **cosine** specifies a G-floating number.

DESCRIPTION

The angle in radians whose cosine is X is computed as:

Value of Cosine	Value Returned
0	$\pi/2$
1	0
-1	π
$0 < X < 1$	$zATAN(zSQRT(1 - X^2)/X)$, where zATAN and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$-1 < X < 0$	$zATAN(zSQRT(1 - X^2)/X) + \pi$
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HACOS.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xACOS routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **cosine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

EXAMPLES

1

```
100     !+
        ! This BASIC program demonstrates the use of
        ! MTH$ACOS.
        !-

        EXTERNAL REAL FUNCTION MTH$ACOS
        DECLARE REAL COS_VALUE, ANGLE
300     INPUT "Cosine value between -1 and +1 "; COS_VALUE
400     IF (COS_VALUE < -1) OR (COS_VALUE > 1)
                THEN PRINT "Invalid cosine value"
                     GOTO 300
500     ANGLE = MTH$ACOS( COS_VALUE )
        PRINT "The angle with that cosine is "; ANGLE; "radians"
32767   END
```

This BASIC program prompts for a cosine value and determines the angle that has that cosine. The output generated by this program is as follows:

```
$ RUN ACOS
Cosine value betwen -1 and +1 ?  .5
The angle with that cosine is 1.0472 radians
```

2

```
PROGRAM GETANGLE(INPUT,OUTPUT);

{+}
{ This PASCAL program uses MTH$ACOS to determine
{ the angle which has the cosine given as input.
{-}

VAR
        COS : REAL;

FUNCTION MTH$ACOS(COS : REAL) : REAL;
        EXTERN;

BEGIN
        WRITE('Cosine value between -1 and +1:  ');
        READ (COS);
        WRITELN('The angle with that cosine is ', MTH$ACOS(COS),
        ' radians');
END.
```

This PASCAL program prompts for a cosine value and determines the angle that has that cosine. The output generated by this program is as follows:

```
$ RUN ACOS
Cosine value between -1 and +1:  .5
The angle with that cosine is 1.04720E+00 radians
```

MTH$xACOSD Arc Cosine of Angle Expressed in Degrees

Given the cosine of an angle, the Arc Cosine of Angle Expressed in Degrees routine returns that angle (in degrees).

FORMAT

MTH$ACOSD *cosine*
MTH$DACOSD *cosine*
MTH$GACOSD *cosine*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ACOSD_R4
MTH$DACOSD_R7
MTH$GACOSD_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in degrees. The angle returned will have a value in the range

$$0 \leq angle \leq 180$$

MTH$ACOSD returns an F-floating number. MTH$DACOSD returns a D-floating number. MTH$GACOSD returns a G-floating number.

ARGUMENTS

cosine
VMS usage: **floating_point**
type: **F_floating, G_floating, D_floating**
access: **read only**
mechanism: **by reference**

Cosine of the angle whose value (in degrees) is to be returned. The **cosine** argument is the address of a floating-point number that is this cosine. The absolute value of **cosine** must be less than or equal to 1. For MTH$ACOSD, **cosine** specifies an F-floating number. For MTH$DACOSD, **cosine** specifies a D-floating number. For MTH$GACOSD, **cosine** specifies a G-floating number.

DESCRIPTION

The angle in degrees whose cosine is X is computed as:

Value of Cosine	Angle Returned
0	90
1	0
−1	180
$0 < X < 1$	$zATAND(zSQRT(1 - X^2)/X)$, where zATAND and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$-1 < X < 0$	$zATAND(zSQRT(1 - X^2)/X) + 180$
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HACOSD.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xACOSD routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **cosine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

EXAMPLE

```
PROGRAM ACOSD(INPUT,OUTPUT);

{+}
{ This PASCAL program demonstrates the use of
{ MTH$ACOSD.
{-}

FUNCTION MTH$ACOSD(COS : REAL): REAL; EXTERN;

VAR
  COSINE : REAL;
  RET_STATUS : REAL;

BEGIN
  COSINE := 0.5;
  RET_STATUS := MTH$ACOSD(COSINE);
  WRITELN('The angle, in degrees, is: ', RET_STATUS);
END.
```

The output generated by this PASCAL example program is as follows:

```
The angle, expressed in degrees, is:  6.00000E+01
```

MTH$xASIN Arc Sine in Radians

Given the sine of an angle, the Arc Sine in Radians routine returns that angle (in radians).

FORMAT

MTH$ASIN *sine*
MTH$DASIN *sine*
MTH$GASIN *sine*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ASIN_R4
MTH$DASIN_R7
MTH$GASIN_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in radians. The angle returned will have a value in the range

$$-\pi/2 \leq angle \leq \pi/2$$

MTH$ASIN returns an F-floating number. MTH$DASIN returns a D-floating number. MTH$GASIN returns a G-floating number.

ARGUMENTS

sine
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The sine of the angle whose value (in radians) is to be returned. The **sine** argument is the address of a floating-point number that is this sine. The absolute value of **sine** must be less than or equal to 1. For MTH$ASIN, **sine** specifies an F-floating number. For MTH$DASIN, **sine** specifies a D-floating number. For MTH$GASIN, **sine** specifies a G-floating number.

MTH$xASIN

DESCRIPTION

The angle in radians whose sine is X is computed as:

Value of Sine	Angle Returned
0	0
1	$\pi/2$
-1	$-\pi/2$
$0 < \|X\| < 1$	$zATAN(X/zSQRT(1 - X^2))$, where zATAN and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HASIN.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xASIN routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **sine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xASIND Arc Sine in Degrees

Given the sine of an angle, the Arc Sine in Degrees routine returns that angle (in degrees).

FORMAT

MTH$ASIND *sine*
MTH$DASIND *sine*
MTH$GASIND *sine*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ASIND_R4
MTH$DASIND_R7
MTH$GASIND_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in degrees. The angle returned will have a value in the range

$$-90 \leq angle \leq 90$$

MTH$ASIND returns an F-floating number. MTH$DASIND returns a D-floating number. MTH$GASIND returns a G-floating number.

ARGUMENTS

sine
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Sine of the angle whose value (in degrees) is to be returned. The **sine** argument is the address of a floating-point number that is this sine. The absolute value of **sine** must be less than or equal to 1. For MTH$ASIND, **sine** specifies an F-floating number. For MTH$DASIND, **sine** specifies a D-floating number. For MTH$GASIND, **sine** specifies a G-floating number.

MTH$xASIND

DESCRIPTION

The angle in degrees whose sine is X is computed as:

Value of Sine	Value Returned
0	0
1	90
-1	-90
$0 < \|X\| < 1$	$zATAND(X/zSQRT(1 - X^2))$, where zATAND and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HASIND.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xASIND routine encountered a floating point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **sine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xATAN Arc Tangent in Radians

Given the tangent of an angle, the Arc Tangent in Radians routine returns that angle (in radians).

FORMAT

MTH$ATAN *tangent*
MTH$DATAN *tangent*
MTH$GATAN *tangent*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ATAN_R4
MTH$DATAN_R7
MTH$GATAN_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in radians. The angle returned will have a value in the range

$$-\pi/2 \leq angle \leq \pi/2$$

MTH$ATAN returns an F-floating number. MTH$DATAN returns a D-floating number. MTH$GATAN returns a G-floating number.

ARGUMENTS

tangent
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The tangent of the angle whose value (in radians) is to be returned. The **tangent** argument is the address of a floating-point number that is this tangent. For MTH$ATAN, **tangent** specifies an F-floating number. For MTH$DATAN, **tangent** specifies a D-floating number. For MTH$GATAN, **tangent** specifies a G-floating number.

MTH$xATAN

DESCRIPTION

In radians, the computation of the arc tangent function is based on the following identities:

$\arctan(X) = X - X^3/3 + X^5/5 - X^7/7 + ...$

$\arctan(X) = X + X * Q(X^2)$,
where $Q(Y) = -Y/3 + Y^2/5 - Y^3/7 + ...$

$\arctan(X) = X * P(X^2)$,
where $P(Y) = 1 - Y/3 + Y^2/5 - Y^3/7 + ...$

$\arctan(X) = \pi/2 - \arctan(1/X)$

$\arctan(X) = \arctan(A) + \arctan((X - A)/(1 + A * X))$
for any real A

The angle in radians whose tangent is X is computed as:

Value of *X*	**Angle Returned**
$0 \leq X \leq 3/32$	$X + X * Q(X^2)$
$3/32 < X \leq 11$	$ATAN(A) + V * (P(V^2))$, where A and ATAN(A) are chosen by table lookup and $V = (X - A)/(1 + A * X)$
$11 < X$	$\pi/2 - W * (P(W^2))$ where $W = 1/X$
$X < 0$	$-zATAN(\lvert X \rvert)$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HATAN.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATAN routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xATAND Arc Tangent in Degrees

Given the tangent of an angle, the Arc Tangent in Degrees routine returns that angle (in degrees).

FORMAT

MTH$ATAND *tangent*
MTH$DATAND *tangent*
MTH$GATAND *tangent*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ATAND_R4
MTH$DATAND_R7
MTH$GATAND_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage:	**floating_point**
type:	**F_floating, D_floating, G_floating**
access:	**write only**
mechanism:	**by value**

Angle in degrees. The angle returned will have a value in the range

$$-90 \leq angle \leq 90$$

MTH$ATAND returns an F-floating number. MTH$DATAND returns a D-floating number. MTH$GATAND returns a G-floating number.

ARGUMENTS

tangent

VMS usage:	**floating_point**
type:	**F_floating, D_floating, G_floating**
access:	**read only**
mechanism:	**by reference**

The tangent of the angle whose value (in degrees) is to be returned. The **tangent** argument is the address of a floating-point number that is this tangent. For MTH$ATAND, **tangent** specifies an F-floating number. For MTH$DATAND, **tangent** specifies a D-floating number. For MTH$GATAND, **tangent** specifies a G-floating number.

MTH$xATAND

DESCRIPTION

The computation of the arc tangent function is based on the following identities:

$$\arctan(X) = (180/\pi) * (X - X^3/3 + X^5/5 - X^7/7 + ...)$$

$$\arctan(X) = 64 * X + X * Q(X^2),$$
where $Q(Y) = 180/\pi * [(1 - 64 * \pi/180)] - Y/3 + Y^2/5 - Y^3/7 + Y^4/9$

$$\arctan(X) = X * P(X^2),$$
where $P(Y) = 180/\pi * [1 - Y/3 + Y^2/5 - Y^3/7 + Y^4/9...]$

$$\arctan(X) = 90 - \arctan(1/X)$$

$$\arctan(X) = \arctan(A) + \arctan((X - A)/(1 + A * X))$$

The angle in degrees whose tangent is X is computed as:

Tangent	Angle Returned
$X \leq 3/32$	$64 * X + X * Q(X^2)$
$3/32 < X \leq 11$	$ATAND(A) + V * P(V^2)$, where A and ATAND(A) are chosen by table lookup and $V = (X - A)/(1 + A * X)$
$11 < X$	$90 - W * (P(W^2))$, where $W = 1/X$
$X < 0$	-zATAND(|X|)

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HATAND.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATAND routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xATAN2 Arc Tangent in Radians with Two Arguments

Given **sine** and **cosine**, the Arc Tangent in Radians with Two Arguments routine returns the angle (in radians) whose tangent is given by the quotient of **sine** and **cosine**, (**sine/cosine**).

FORMAT

MTH$ATAN2 *sine ,cosine*
MTH$DATAN2 *sine ,cosine*
MTH$GATAN2 *sine ,cosine*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle in radians. MTH$ATAN2 returns an F-floating number. MTH$DATAN2 returns a D-floating number. MTH$GATAN2 returns a G-floating number.

ARGUMENTS

sine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Dividend. The **sine** argument is the address of a floating-point number that is this dividend. For MTH$ATAN2, **sine** specifies an F-floating number. For MTH$DATAN2, **sine** specifies a D-floating number. For MTH$GATAN2, **sine** specifies a G-floating number.

cosine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Divisor. The **cosine** argument is the address of a floating-point number that is this divisor. For MTH$ATAN2, **cosine** specifies an F-floating number. For MTH$DATAN2, **cosine** specifies a D-floating number. For MTH$GATAN2, **cosine** specifies a G-floating number.

DESCRIPTION

The angle in radians whose tangent is Y/X is computed as follows, where f is defined in the description of MTH$zCOSH.

Value of Input Arguments	Angle Returned
$X = 0$ *or* $Y/X > 2^{(f+1)}$	$\pi/2 * (signY)$
$X > 0$ *and* $Y/X \leq 2^{(f+1)}$	$zATAN(Y/X)$
$X < 0$ *and* $Y/X \leq 2^{(f+1)}$	$\pi * (signY) + zATAN(Y/X)$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HATAN2.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATAN2 routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. Both **cosine** and **sine** are zero. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xATAND2 Arc Tangent in Degrees with Two Arguments

Given **sine** and **cosine**, the Arc Tangent in Degrees with Two Arguments routine returns the angle (in degrees) whose tangent is given by the quotient of **sine** and **cosine**, (**sine/cosine**).

FORMAT

MTH$ATAND2 *sine ,cosine*
MTH$DATAND2 *sine ,cosine*
MTH$GATAND2 *sine ,cosine*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Angle (in degrees). MTH$ATAND2 returns an F-floating number. MTH$DATAND2 returns a D-floating number. MTH$GATAND2 returns a G-floating number.

ARGUMENTS

sine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Dividend. The **sine** argument is the address of a floating-point number that is this dividend. For MTH$ATAND2, **sine** specifies an F-floating number. For MTH$DATAND2, **sine** specifies a D-floating number. For MTH$GATAND2, **sine** specifies a G-floating number.

cosine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Divisor. The **cosine** argument is the address of a floating-point number that is this divisor. For MTH$ATAND2, **cosine** specifies an F-floating number. For MTH$DATAND2, **cosine** specifies a D-floating number. For MTH$GATAND2, **cosine** specifies a G-floating number.

MTH$xATAND2

DESCRIPTION

The angle in degrees whose tangent is Y/X is computed below and where f is defined in the description of MTH$zCOSH.

Value of Input Arguments	Angle Returned
$X = 0$ *or* $Y/X > 2^{(f+1)}$	$90 * (signY)$
$X > 0$ *and* $Y/X \leq 2^{(f+1)}$	$zATAND(Y/X)$
$X < 0$ *and* $Y/X \leq 2^{(f+1)}$	$180 * (signY) + zATAND(Y/X)$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HATAND2.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATAND2 routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. Both **cosine** and **sine** are zero. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xATANH Hyperbolic Arc Tangent

Given the hyperbolic tangent of an angle, the Hyperbolic Arc Tangent routine returns the hyperbolic arc tangent of that angle.

FORMAT

MTH$ATANH *hyperbolic-tangent*
MTH$DATANH *hyperbolic-tangent*
MTH$GATANH *hyperbolic-tangent*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The hyperbolic arc tangent of **hyperbolic-tangent**. MTH$ATANH returns an F-floating number. MTH$DATANH returns a D-floating number. MTH$GATANH returns a G-floating number.

ARGUMENTS

hyperbolic-tangent
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Hyperbolic tangent of an angle. The **hyperbolic-tangent** argument is the address of a floating-point number that is this hyperbolic tangent. For MTH$ATANH, **hyperbolic-tangent** specifies an F-floating number. For MTH$DATANH, **hyperbolic-tangent** specifies a D-floating number. For MTH$GATANH, **hyperbolic-tangent** specifies a G-floating number.

DESCRIPTION

The hyperbolic arc tangent function is computed as follows:

Value of x	Value Returned
$\lvert X\rvert < 1$	$zATANH(X) = zLOG((X+1)/(X-1))/2$
$\lvert X\rvert \geq 1$	An invalid argument is signaled

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HATANH.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATANH routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument: $\|X\| \geq 1$. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$CxABS Complex Absolute Value

The Complex Absolute Value routine returns the absolute value of a complex number (r,i).

FORMAT

MTH$CABS *complex-number*
MTH$CDABS *complex-number*
MTH$CGABS *complex-number*

Each of the above three formats accepts as input one of the three floating-point complex types.

RETURNS

VMS usage:	**floating_point**
type:	**F_floating, D_floating, G_floating**
access:	**write only**
mechanism:	**by value**

The absolute value of a complex number. MTH$CABS returns an F-floating number. MTH$CDABS returns a D-floating number. MTH$CGABS returns a G-floating number.

ARGUMENT

complex-number

VMS usage:	**complex_number**
type:	**F_floating complex, D_floating complex, G_floating complex**
access:	**read only**
mechanism:	**by reference**

A complex number (r,i), where r and i are both floating-point complex values. The **complex-number** argument is the address of this complex number. For MTH$CABS, **complex-number** specifies an F-floating complex number. For MTH$CDABS, **complex-number** specifies a D-floating complex number. For MTH$CGABS, **complex-number** specifies a G-floating complex number.

DESCRIPTION

The complex absolute value is computed as follows, where *MAX* is the larger of |r| and |i|, and *MIN* is the smaller of |r| and |i|.

$$result = MAX * SQRT((MIN/MAX)^2 + 1)$$

MTH$CxABS

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CxABS routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library when both **r** and **i** are large.

EXAMPLES

1

```
C+
C    This FORTRAN example forms the absolute value of an
C    F-floating complex number using MTH$CABS and the
C    FORTRAN random number generator RAN.
C
C    Declare Z as a complex value and MTH$CABS as a REAL*4 value.
C    MTH$CABS will return the absolute value of Z:   Z_NEW = MTH$CABS(Z).
C-

        COMPLEX Z
        COMPLEX CMPLX
        REAL*4 Z_NEW,MTH$CABS
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the FORTRAN generic CMPLX.
C-

        Z = CMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r" and
C    imaginary part "i".
C-

       TYPE *, ' The complex number z is',z
       TYPE *, ' It has real part',REAL(Z),'and imaginary part',AIMAG(Z)
       TYPE *, ' '

C+
C    Compute the complex absolute value of Z.
C-

        Z_NEW = MTH$CABS(Z)
        TYPE *, ' The complex absolute value of',z,' is',Z_NEW
        END
```

This example uses an F-floating complex number for **complex-number**. The output of this FORTRAN example is as follows:

```
The complex number z is (0.8535407,0.2043402)
It has real part  0.8535407    and imaginary part  0.2043402

The complex absolute value of (0.8535407,0.2043402) is  0.8776597
```

2
```
C+
C     This FORTRAN example forms the absolute
C     value of a G-floating complex number using
C     MTH$CGABS and the FORTRAN random number
C     generator RAN.
C
C     Declare Z as a complex value and MTH$CGABS as a
C     REAL*8 value. MTH$CGABS will return the absolute
C     value of Z: Z_NEW = MTH$CGABS(Z).
C-

        COMPLEX*16 Z
        REAL*8 Z_NEW,MTH$CGABS

C+
C     Generate a random complex number with the FORTRAN
C     generic CMPLX.
C-

        Z = (12.34567890123,45.536376385345)
        TYPE *, ' The complex number z is',z
        TYPE *, ' '

C+
C     Compute the complex absolute value of Z.
C-

        Z_NEW = MTH$CGABS(Z)
        TYPE *, ' The complex absolute value of',z,' is',Z_NEW
        END
```

This FORTRAN example uses a G-floating complex number for **complex-number**. Because this example uses a G-floating number, it must be compiled as follows:

```
$ FORTRAN/G MTHEX.FOR
```

Notice the difference in the precision of the output generated:

```
The complex number z is (12.3456789012300,45.5363763853450)
The complex absolute value of (12.3456789012300,45.5363763853450) is
 47.1802645376230
```

MTH$CCOS Cosine of a Complex Number (F-floating Value)

The Cosine of a Complex Number (F-floating Value) routine returns the cosine of a complex number as an F-floating value.

FORMAT

MTH$CCOS *complex-number*

RETURNS

VMS usage:	**complex_number**
type:	**F_floating complex**
access:	**write only**
mechanism:	**by value**

The complex cosine of the complex input number. MTH$CCOS returns an F-floating complex number.

ARGUMENTS

complex-number

VMS usage:	**complex_number**
type:	**F_floating complex**
access:	**read only**
mechanism:	**by reference**

A complex number (r,i) where r and i are floating-point numbers. The **complex-number** argument is the address of this complex number. For MTH$CCOS, **complex-number** specifies an F-floating complex number.

DESCRIPTION

The complex cosine is calculated as follows:

$$result = (COS(r) * COSH(i), -SIN(r) * SINH(i))$$

The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$CxCOS.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CCOS routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **i** is greater than about 88.029 for F-floating values.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex
C    cosine of an F-floating complex number using
C    MTH$CCOS and the FORTRAN random number
C    generator RAN.
C
C    Declare Z and MTH$CCOS as complex values.
C    MTH$CCOS will return the cosine value of
C    Z:          Z_NEW = MTH$CCOS(Z)
C-

        COMPLEX Z,Z_NEW,MTH$CCOS
        COMPLEX CMPLX
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic CMPLX.
C-

        Z = CMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r" and
C    imaginary part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' It has real part',REAL(Z),'and imaginary part',AIMAG(Z)
        TYPE *, ' '

C+
C    Compute the complex cosine value of Z.
C-

        Z_NEW = MTH$CCOS(Z)
        TYPE *, ' The complex cosine value of',z,' is',Z_NEW
        END
```

This FORTRAN example demonstrates the use of MTH$CCOS, using the MTH$CCOS entry point. The output of this program is as follows:

```
The complex number z is (0.8535407,0.2043402)
It has real part  0.8535407    and imaginary part  0.2043402
The complex cosine value of (0.8535407,0.2043402) is (0.6710899,-0.1550672)
```

MTH$CxCOS Cosine of a Complex Number

The Cosine of a Complex Number routine returns the cosine of a complex number.

FORMAT

MTH$CDCOS *complex-cosine ,complex-number*
MTH$CGCOS *complex-cosine ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-cosine

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

Complex cosine of the **complex-number**. The complex cosine routines that have D-floating and G-floating complex input values write the address of the complex cosine into the **complex-cosine** argument. For MTH$CDCOS, the **complex-cosine** argument specifies a D-floating complex number. For MTH$CGCOS, the **complex-number** argument specifies a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

A complex number (r,i) where r and i are floating-point numbers. The **complex-number** argument is the address of this complex number. For MTH$CDCOS, **complex-number** specifies a D-floating complex number. For MTH$CGCOS, **complex-number** specifies a G-floating complex number.

DESCRIPTION

The complex cosine is calculated as follows:

$$result = (COS(r) * COSH(i), -SIN(r) * SINH(i))$$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CxCOS routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **i** is greater than about 88.029 for F-floating and D-floating values or greater than 709.089 for G-floating values.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex
C    cosine of a D-floating complex number using
C    MTH$CDCOS and the FORTRAN random number
C    generator RAN.
C
C    Declare Z and MTH$CDCOS as complex values.
C    MTH$CDCOS will return the cosine value of
C    Z:          Z_NEW = MTH$CDCOS(Z)
C-

        COMPLEX*16 Z,Z_NEW,MTH$CDCOS
        COMPLEX*16 DCMPLX
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic DCMPLX.
C-

        Z = DCMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r" and
C    imaginary part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' '

C+
C    Compute the complex cosine value of Z.
C-

        Z_NEW = MTH$CDCOS(Z)
        TYPE *, ' The complex cosine value of',z,' is',Z_NEW
        END
```

This FORTRAN example program demonstrates the use of MTH$CxCOS, using the MTH$CDCOS entry point. Notice the high precision of the output generated:

```
 The complex number z is (0.8535407185554504,0.2043401598930359)
 The complex cosine value of (0.8535407185554504,0.2043401598930359) is
(0.6710899028500762,-0.1550672019621661)
```

MTH$CEXP Complex Exponential (F-floating Value)

The Complex Exponential (F-floating Value) routine returns the complex exponential of a complex number as an F-floating value.

FORMAT

MTH$CEXP *complex-number*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

Complex exponential of the complex input number. MTH$CEXP returns an F-floating complex number.

ARGUMENTS

complex-number
VMS usage: **complex_number**
type: **F_floating complex**
access: **read only**
mechanism: **by reference**

Complex number whose complex exponential is to be returned. This complex number has the form (r,i), where r is the real part and i is the imaginary part. The **complex-number** argument is the address of this complex number. For MTH$CEXP, **complex-number** specifies an F-floating number.

DESCRIPTION

The complex exponential is computed as follows:

$$complex - exponent = (EXP(r) * COS(i), EXP(r) * SIN(i))$$

The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$CxEXP.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CEXP routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **r** is greater than about 88.029 for F-floating values.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex exponential
C    of an F-floating complex number using MTH$CEXP
C    and the FORTRAN random number generator RAN.
C
C    Declare Z and MTH$CEXP as complex values. MTH$CEXP
C    will return the exponential value of Z: Z_NEW = MTH$CEXP(Z)
C-

        COMPLEX Z,Z_NEW,MTH$CEXP
        COMPLEX CMPLX
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic CMPLX.
C-

        Z = CMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r"
C    and imaginary part "i".
C-

       TYPE *, ' The complex number z is',z
       TYPE *, ' It has real part',REAL(Z),'and imaginary part',AIMAG(Z)
       TYPE *, ' '

C+
C    Compute the complex exponential value of Z.
C-

        Z_NEW = MTH$CEXP(Z)
        TYPE *, ' The complex exponential value of',z,' is',Z_NEW
        END
```

This FORTRAN program demonstrates the use of MTH$CEXP as a function call. The output generated by this example is as follows:

```
The complex number z is (0.8535407,0.2043402)
It has real part  0.8535407    and imaginary part  0.2043402
The complex exponential value of (0.8535407,0.2043402) is
 (2.299097,0.4764476)
```

MTH$CxEXP Complex Exponential

The Complex Exponential routine returns the complex exponential of a complex number.

FORMAT

MTH$CDEXP *complex-exponent ,complex-number*
MTH$CGEXP *complex-exponent ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-exponent

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

Complex exponential of **complex-number**. The complex exponential routines that have D-floating complex and G-floating complex input values write the **complex-exponent** into this argument. For MTH$CDEXP, **complex-exponent** argument specifies a D-floating complex number. For MTH$CGEXP, **complex-exponent** specifies a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

Complex number whose complex exponential is to be returned. This complex number has the form (r,i), where *r* is the real part and *i* is the imaginary part. The **complex-number** argument is the address of this complex number. For MTH$CDEXP, **complex-number** specifies a D-floating number. For MTH$CGEXP, **complex-number** specifies a G-floating number.

DESCRIPTION

The complex exponential is computed as follows:

$$complex - exponent = (EXP(r) * COS(i), EXP(r) * SIN(i))$$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CxEXP routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **r** is greater than about 88.029 for D-floating values or greater than about 709.089 for G-floating values.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex exponential
C    of a G-floating complex number using MTH$CGEXP
C    and the FORTRAN random number generator RAN.
C
C    Declare Z and MTH$CGEXP as complex values.
C    MTH$CGEXP will return the exponential value
C    of Z:      CALL MTH$CGEXP(Z_NEW,Z)
C-

        COMPLEX*16 Z,Z_NEW
        COMPLEX*16 MTH$GCMPLX
        REAL*8 R,I
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the FORTRAN
C-   generic CMPLX.
C-

        R = RAN(M)
        I = RAN(M)
        Z = MTH$GCMPLX(R,I)
        TYPE *, ' The complex number z is',z
        TYPE *, ' '

C+
C    Compute the complex exponential value of Z.
C-

        CALL MTH$CGEXP(Z_NEW,Z)
        TYPE *, ' The complex exponential value of',z,' is',Z_NEW
        END
```

This FORTRAN example demonstrates how to access MTH$CGEXP as a procedure call. Because G-floating numbers are used, this program must be compiled using the command "FORTRAN/G filename".

Notice the high precision of the output generated:

```
 The complex number z is (0.853540718555450,0.204340159893036)
 The complex exponential value of (0.853540718555450,0.204340159893036) is
(2.29909677719458,0.476447678044977)
```

MTH$CLOG Complex Natural Logarithm (F-floating Value)

The Complex Natural Logarithm (F-floating Value) routine returns the complex natural logarithm of a complex number as an F-floating value.

FORMAT

MTH$CLOG *complex-number*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

The complex natural logarithm of a complex number. MTH$CLOG returns an F-floating complex number.

ARGUMENTS

complex-number
VMS usage: **complex_number**
type: **F_floating complex**
access: **read only**
mechanism: **by reference**

Complex number whose complex natural logarithm is to be returned. This complex number has the form (r,i), where r is the real part and i is the imaginary part. The **complex-number** argument is the address of this complex number. For MTH$CLOG, **complex-number** specifies an F-floating number.

DESCRIPTION

The complex natural logarithm is computed as follows:

$$CLOG(x) = (LOG(CABS(x)), ATAN2(i, r))$$

The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$CxLOG.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CLOG routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

Examples of using MTH$CLOG from VAX MACRO (using both the CALLS and the CALLG instructions) appear in the introductory section of this manual.

MTH$CxLOG Complex Natural Logarithm

The Complex Natural Logarithm routine returns the complex natural logarithm of a complex number.

FORMAT

MTH$CDLOG *complex-natural-log ,complex-number*
MTH$CGLOG *complex-natural-log ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-natural-log

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

Natural logarithm of the complex number specified by **complex-number**. The complex natural logarithm routines that have D-floating complex and G-floating complex input values write the address of the complex natural logarithm into **complex-natural-log**. For MTH$CDLOG, the **complex-natural-log** argument specifies a D-floating complex number. For MTH$CGLOG, the **complex-natural-log** argument specifies a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

Complex number whose complex natural logarithm is to be returned. This complex number has the form (r,i), where r is the real part and i is the imaginary part. The **complex-number** argument is the address of this complex number. For MTH$CDLOG, **complex-number** specifies a D-floating number. For MTH$CGLOG, **complex-number** specifies a G-floating number.

DESCRIPTION

The complex natural logarithm is computed as follows:

$$CLOG(x) = (LOG(CABS(x)), ATAN2(i, r))$$

CONDITION VALUE SIGNALED

SS$_ROPRAND — Reserved operand. The MTH$CxLOG routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex logarithm
C    of a D-floating complex number by using MTH$CDLOG
C    and the FORTRAN random number generator RAN.
C
C    Declare Z and MTH$CDLOG as complex values. Then MTH$CDLOG
c    will return the logarithm of Z: CALL MTH$CDLOG(Z_NEW,Z).
C
C    Declare Z,Z_LOG, and MTH$DCMPLX as complex values,
C    and R and I as real values. MTH$DCMPLX takes two real
C    arguments and returns one complex number.
C
C    Given a complex number Z, MTH$CDLOG(Z) returns the
C    complex natural logarithm of Z.
C-
        COMPLEX*16 Z,Z_NEW,MTH$DCMPLX
        REAL*8 R,I
        R = 3.1425637846746565
        I = 7.43678469887
        Z = MTH$DCMPLX(R,I)
C+
C    Z is a complex number (r,i) with real part "r" and imaginary
C    part "i".
C-
        TYPE *, ' The complex number z is',z
        TYPE *, ' '
        CALL MTH$CDLOG(Z_NEW,Z)
        TYPE *,' The complex logarithm of',z,' is',Z_NEW
        END
```

This FORTRAN example program uses MTH$CDLOG by calling it as a procedure. The output generated by this program is as follows:

```
 The complex number z is (3.142563784674657,7.436784698870000)
 The complex logarithm of (3.142563784674657,7.436784698870000) is
(2.088587642177504,1.170985519274141)
```

MTH$CMPLX Complex Number Made from F-floating-Point

The Complex Number Made from F-floating-Point routine returns a complex number from two floating-point input values.

FORMAT

MTH$CMPLX *real-part ,imaginary-part*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

A complex number. MTH$CMPLX returns an F-floating complex number.

ARGUMENTS

real-part

VMS usage: **floating_point**
type: **F_floating**
access: **read only**
mechanism: **by reference**

Real part of a complex number. The **real-part** argument is the address of a floating-point number that contains this real part, r, of (r,i). For MTH$CMPLX, **real-part** specifies an F-floating number.

imaginary-part

VMS usage: **floating_point**
type: **F_floating**
access: **read only**
mechanism: **by reference**

Imaginary part of a complex number. The **imag-parg** argument is the address of a floating-point number that contains this imaginary part, i, of (r,i). For MTH$CMPLX, **imaginary-part** specifies an F-floating number.

DESCRIPTION

The MTH$CMPLX routines return a complex number from two F-floating input values. The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$xCMPLX.

CONDITION VALUE SIGNALED

SS$_ROPRAND — Reserved operand. The MTH$CMPLX routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms two F-floating
C    point complex numbers using MTH$CMPLX
C    and the FORTRAN random number generator RAN.
C
C    Declare Z and MTH$CMPLX as complex values, and R
C    and I as real values.  MTH$CMPLX takes two real
C    F-floating point values and returns one COMPLEX*8 number.
C
C    Note, since CMPLX is a generic name in FORTRAN, it would be
C    sufficient to use CMPLX.
C    CMPLX must be declare to be of type COMPLEX*8.
C
C    Z = CMPLX(R,I)
C-

        COMPLEX Z,MTH$CMPLX,CMPLX
        REAL*4 R,I
        INTEGER M
        M = 1234567
        R = RAN(M)
        I = RAN(M)
        Z = MTH$CMPLX(R,I)

C+
C    Z is a complex number (r,i) with real part "r" and
C    imaginary part "i".
C-

        TYPE *, ' The two input values are:',R,I
        TYPE *, ' The complex number z is',z
        z = CMPLX(RAN(M),RAN(M))
        TYPE *, ' '
        TYPE *, ' Using the FORTRAN generic CMPLX with random R and I:'
        TYPE *, ' The complex number z is',z
        END
```

This FORTRAN example program demonstrates the use of MTH$CMPLX. The output generated by this program is as follows:

```
The two input values are:  0.8535407      0.2043402
The complex number z is (0.8535407,0.2043402)
Using the FORTRAN generic CMPLX with random R and I:
The complex number z is (0.5722565,0.1857677)
```

MTH$xCMPLX Complex Number Made from D- or G-floating-Point

The Complex Number Made from D- or G-floating-Point routine returns a complex number from two D- or G-floating input values.

FORMAT

MTH$DCMPLX *complx ,real-part ,imaginary-part*
MTH$GCMPLX *complx ,real-part ,imaginary-part*

Each of the above formats accepts as input one of floating-point complex types.

RETURNS

None.

ARGUMENTS

complx

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

The floating-point complex value of a complex number. The complex exponential functions that have D-floating complex and G-floating complex input values write the address of this floating-point complex value into **complx**. For MTH$DCMPLX, **complx** specifies a D-floating complex number. For MTH$GCMPLX, **complx** specifies a G-floating complex number. For MTH$CMPLX, **complx** is not used.

real-part

VMS usage: **floating_point**
type: **D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Real part of a complex number. The **real-part** argument is the address of a floating-point number that contains this real part, r, of (r,i). For MTH$DCMPLX, **real-part** specifies a D-floating number. For MTH$GCMPLX, **real-part** specifies a G-floating number.

imaginary-part

VMS usage: **floating_point**
type: **D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Imaginary part of a complex number. The **imag-parg** argument is the address of a floating-point number that contains this imaginary part, i, of (r,i). For MTH$DCMPLX, **imaginary-part** specifies a D-floating number. For MTH$GCMPLX, **imaginary-part** specifies a G-floating number.

CONDITION VALUE SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xCMPLX routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms two D-floating
C    point complex numbers using MTH$CMPLX
C    and the FORTRAN random number generator RAN.
C
C    Declare Z and MTH$DCMPLX as complex values, and R
C    and I as real values.  MTH$DCMPLX takes two real
C    D-floating point values and returns one
C    COMPLEX*16 number.
C
C-

        COMPLEX*16 Z
        REAL*8 R,I
        INTEGER M
        M = 1234567
        R = RAN(M)
        I = RAN(M)
        CALL MTH$DCMPLX(Z,R,I)

C+
C    Z is a complex number (r,i) with real part "r" and imaginary
C    part "i".
C-

        TYPE *, ' The two input values are:',R,I
        TYPE *, ' The complex number z is',Z
        END
```

This FORTRAN example demonstrates how to make a procedure call to MTH$DCMPLX. Notice the difference in the precision of the output generated.

```
 The two input values are:  0.8535407185554504      0.2043401598930359
 The complex number z is (0.8535407185554504,0.2043401598930359)
```

MTH$CONJG Conjugate of a Complex Number (F-floating Value)

The Conjugate of a Complex Number (F-floating Value) routine returns the complex conjugate (r,-i) of a complex number (r,i) as an F-floating value.

FORMAT

MTH$CONJG *complex-number*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

Complex conjugate of a complex number. MTH$CONJG returns an F-floating complex number.

ARGUMENTS

complex-number
VMS usage: **complex_number**
type: **F_floating complex**
access: **read only**
mechanism: **by reference**

A complex number (r,i), where r and i are floating-point numbers. The **complex-number** argument is the address of this floating-point complex number. For MTH$CONJG, **complex-number** specifies an F-floating number.

DESCRIPTION

The MTH$CONJG routine return the complex conjugate (r,-i) of a complex number (r,i) as an F-floating value. The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$xCONJG.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CONJG routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xCONJG Conjugate of a Complex Number

The Conjugate of a Complex Number routine returns the complex conjugate (r,-i) of a complex number (r,i).

FORMAT

MTH$DCONJG *complex-conjugate ,complex-number*
MTH$GCONJG *complex-conjugate ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-conjugate

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

The complex conjugate (r,-i) of the complex number specified by **complex-number**. MTH$DCONJG and MTH$GCONJG write the address of this complex conjugate into **complex-conjugate**. For MTH$DCONJG, the **complex-conjugate** argument specifies the address of a D-floating complex number. For MTH$GCONJG, the **complex-conjugate** argument specifies the address of a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

A complex number (r,i), where r and i are floating-point numbers. The **complex-number** argument is the address of this floating-point complex number. For MTH$DCONJG, **complex-number** specifies a D-floating number. For MTH$GCONJG, **complex-number** specifies a G-floating number.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xCONJG routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex conjugate
C    of a G-floating complex number using MTH$GCONJG
C    and the FORTRAN random number generator RAN.
C
C    Declare Z, Z_NEW, and MTH$GCONJG as a complex values.
C    MTH$GCONJG will return the complex conjugate
C    value of Z:    Z_NEW = MTH$GCONJG(Z).
C-

        COMPLEX*16 Z,Z_NEW,MTH$GCONJG
        COMPLEX*16 MTH$GCMPLX
        REAL*8 R,I,MTH$GREAL,MTH$GIMAG
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic CMPLX.
C-

        R = RAN(M)
        I = RAN(M)
        Z = MTH$GCMPLX(R,I)
        TYPE *, ' The complex number z is',z
        TYPE 1,MTH$GREAL(Z),MTH$GIMAG(Z)
   1    FORMAT(' with real part ',F20.16,' and imaginary part',F20.16)
        TYPE *, ' '

C+
C    Compute the complex absolute value of Z.
C-

        Z_NEW = MTH$GCONJG(Z)
        TYPE *, ' The complex conjugate value of',z,' is',Z_NEW
        TYPE 1,MTH$GREAL(Z_NEW),MTH$GIMAG(Z_NEW)
        END
```

This FORTRAN example demonstrates how to make a function call to MTH$GCONJG. Because G-floating numbers are used, the examples must be compiled with the statement "FORTRAN/G filename".

The output generated by this program is as follows:

```
The complex number z is (0.853540718555450,0.204340159893036)
  with real part    0.8535407185554504
  and imaginary part  0.2043401598930359

The complex conjugate value of
  (0.853540718555450,0.204340159893036) is
  (0.853540718555450,-0.204340159893036)
  with real part    0.8535407185554504
  and imaginary part -0.2043401598930359
```

MTH$xCOS Cosine of Angle Expressed in Radians

The Cosine of Angle Expressed in Radians routine returns the cosine of a given angle (in radians).

FORMAT

MTH$COS *angle-in-radians*
MTH$DCOS *angle-in-radians*
MTH$GCOS *angle-in-radians*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$COS_R4
MTH$DCOS_R7
MTH$GCOS_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Cosine of the angle. MTH$COS returns an F-floating number. MTH$DCOS returns a D-floating number. MTH$GCOS returns a G-floating number.

ARGUMENTS

angle-in-radians
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The angle in radians. The **angle-in-radians** argument is the address of a floating-point number. For MTH$COS, **angle-in-radians** is an F-floating number. For MTH$DCOS, **angle-in-radians** specifies a D-floating number. For MTH$GCOS, **angle-in-radians** specifies a G-floating number.

DESCRIPTION

See the MTH$xSINCOS routine for the algorithm used to compute the cosine.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HCOS.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xCOS procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xCOSD Cosine of Angle Expressed in Degrees

The Cosine of Angle Expressed in Degrees routine returns the cosine of a given angle (in degrees).

FORMAT

MTH$COSD *angle-in-degrees*
MTH$DCOSD *angle-in-degrees*
MTH$GCOSD *angle-in-degrees*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$COSD_R4
MTH$DCOSD_R7
MTH$GCOSD_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Cosine of the angle. MTH$COSD returns an F-floating number. MTH$DCOSD returns a D-floating number. MTH$GCOSD returns a G-floating number.

ARGUMENTS

angle-in-degrees
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number. For MTH$COSD, **angle-in-degrees** specifies an F-floating number. For MTH$DCOSD, **angle-in-degrees** specifies a D-floating number. For MTH$GCOSD, **angle-in-degrees** specifies a G-floating number.

DESCRIPTION

See the MTH$SINCOSD routine for the algorithm used to compute the cosine.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HCOSD.

CONDITION VALUE SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xCOSD procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xCOSH Hyperbolic Cosine

The Hyperbolic Cosine routine returns the hyperbolic cosine of the input value.

FORMAT

MTH$COSH *floating-point-input-value*
MTH$DCOSH *floating-point-input-value*
MTH$GCOSH *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The hyperbolic cosine of the input value **floating-point-input-value**. MTH$COSH returns an F-floating number. MTH$DCOSH returns a D-floating number. MTH$GCOSH returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of this input value. For MTH$COSH, **floating-point-input-value** specifies an F-floating number. For MTH$DCOSH, **floating-point-input-value** specifies a D-floating number. For MTH$GCOSH, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

Computation of the hyperbolic cosine depends on the magnitude of the input argument. The range of the function is partitioned using four data-type-dependent constants: a(z), b(z), and c(z). The subscript *z* indicates the data type. The constants depend on the number of exponent bits (*e*) and the number of fraction bits (*f*) associated with the data type (z).

The values of *e* and *f* are:

z	e	f
F	8	24
D	8	56
G	11	53

The values of the constants in terms of e and f are:

Variable	Value
a(z)	$2^{(-f/2)}$
b(z)	CEILING[$(f+1)/2 * \ln(2)$]
c(z)	$(2^{e-1}) * \ln(2)$

Based on the above definitions, zCOSH(X) is computed as follows:

Value of X	Value Returned
$\|X\| < a(z)$	1
$a(z) \leq \|X\| < .25$	Computed using a power series expansion in $\|X\|^2$
$.25 \leq \|X\| < b(z)$	$(zEXP(\|X\|) + 1/zEXP(\|X\|))/2$
$b(z) \leq \|X\| < c(z)$	$zEXP(\|X\|)/2$
$c(z) \leq \|x\|$	Overflow occurs

This routine description for the H-floating point value is listed alphabetically under MTH$HCOSH.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xCOSH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **floating-point-input-value** is greater than about *yyy*; LIB$SIGNAL copies the reserved operand to the signal mechanism vector. The result is the reserved operand -0.0 unless a condition handler changes the signal mechanism vector. The values of *yyy* are: MTH$COSH—88.722 MTH$DCOSH—88.722 MTH$GCOSH—709.782

MTH$CSIN Sine of a Complex Number (F-floating Value)

The Sine of a Complex Number (F-floating Value) routine returns the sine of a complex number (r,i) as an F-floating value.

FORMAT

MTH$CSIN *complex-number*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

Complex sine of the complex number. MTH$CSIN returns an F-floating complex number.

ARGUMENTS

complex-number
VMS usage: **complex_number**
type: **F_floating complex**
access: **read only**
mechanism: **by reference**

A complex number (r,i), where r and i are floating-point numbers. The **complex-number** argument is the address of this complex number. For MTH$CSIN, **complex-number** specifies an F-floating complex number.

DESCRIPTION

The complex sine is computed as follows:

$$complex - sine = (SIN(r) * COSH(i), COS(r) * SINH(i))$$

The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$CxSIN.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CSIN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **i** is greater than about 88.029 for F-floating values.

MTH$CxSIN Sine of a Complex Number

The Sine of a Complex Number routine returns the sine of a complex number (r,i).

FORMAT

MTH$CDSIN *complex-sine ,complex-number*
MTH$CGSIN *complex-sine ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-sine

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

Complex sine of the complex number. The complex sine routines with D-floating complex and G-floating complex input values write the complex sine into this **complex-sine** argument. For MTH$CDSIN, **complex-sine** specifies a D-floating complex number. For MTH$CGSIN, **complex-sine** specifies a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

A complex number (r,i), where r and i are floating-point numbers. The **complex-number** argument is the address of this complex number. For MTH$CDSIN, **complex-number** specifies a D-floating complex number. For MTH$CGSIN, **complex-number** specifies a G-floating complex number.

DESCRIPTION

The complex sine is computed as follows:

$$complex - sine = (SIN(r) * COSH(i), COS(r) * SINH(i))$$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CxSIN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **i** is greater than about 88.029 for D-floating values or greater than about 709.089 for G-floating values.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex
C    sine of a G-floating complex number using
C    MTH$CGSIN and the FORTRAN random number
C    generator RAN.
C
C    Declare Z and MTH$CGSIN as complex values.
C    MTH$CGSIN will return the sine value
C    of Z:      CALL MTH$CGSIN(Z_NEW,Z)
C-

        COMPLEX*16 Z,Z_NEW
        COMPLEX*16 DCMPLX
        REAL*8 R,I
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic DCMPLX.
C-

        R = RAN(M)
        I = RAN(M)
        Z = DCMPLX(R,I)

C+
C     Z is a complex number (r,i) with real part "r" and
C     imaginary part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' '

C+
C    Compute the complex sine value of Z.
C-

        CALL MTH$CGSIN(Z_NEW,Z)
        TYPE *, ' The complex sine value of',z,' is',Z_NEW
        END
```

This FORTRAN example demonstrates a procedure call to MTH$CGSIN. Because this program uses G-floating numbers, it must be compiled with the statement "FORTRAN/G filename".

The output generated by this program is as follows:

```
 The complex number z is (0.853540718555450,0.204340159893036)
 The complex sine value of (0.853540718555450,0.204340159893036) is
(0.769400835484975,0.135253340912255)
```

MTH$CSQRT Complex Square Root (F-floating Value)

The Complex Square Root (F-floating Value) routine returns the complex square root of a complex number (r,i).

FORMAT

MTH$CSQRT *complex-number*

RETURNS

VMS usage: **complex_number**
type: **F_floating complex**
access: **write only**
mechanism: **by value**

The complex square root of **complex-number**. MTH$CSQRT returns an F-floating number.

ARGUMENTS

complex-number
VMS usage: **complex_number**
type: **F_floating complex**
access: **read only**
mechanism: **by reference**

Complex number (r,i). The **complex-number** argument contains the address of this complex number. For MTH$CSQRT, **complex-number** specifies an F-floating number.

DESCRIPTION

The complex square root is computed as follows.

First, calculate **ROOT** and **Q** using the following equations:

$$ROOT = SQRT((ABS(r) + (CABS(r,i))/2)$$

$$Q = i/(2 * ROOT)$$

Then, the complex result is given as follows:

r	i	CSQRT((r,i))
≥0	Any	(ROOT,Q)
<0	≥0	(Q,ROOT)
<0	<0	(-Q,-ROOT)

The routine descriptions for the D- and G-floating point versions of this routine are listed alphabetically under MTH$CxSQRT.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CSQRT procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$CxSQRT Complex Square Root

The Complex Square Root routine returns the complex square root of a complex number (r,i).

FORMAT

MTH$CDSQRT *complex-square-root ,complex-number*
MTH$CGSQRT *complex-square-root ,complex-number*

Each of the above formats accepts as input one of the floating-point complex types.

RETURNS

None.

ARGUMENTS

complex-square-root

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **write only**
mechanism: **by reference**

Complex square root of the complex number specified by **complex-number**. The complex square root routines that have D-floating complex and G-floating complex input values write the complex square root into **complex-square-root**. For MTH$CDSQRT, **complex-square-root** specifies a D-floating complex number. For MTH$CGSQRT, **complex-square-root** specifies a G-floating complex number.

complex-number

VMS usage: **complex_number**
type: **D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

Complex number (r,i). The **complex-number** argument contains the address of this complex number. For MTH$CDSQRT, **complex-number** specifies a D-floating number. For MTH$CGSQRT, **complex-number** specifies a G-floating number.

DESCRIPTION

The complex square root is computed as follows.

First, calculate **ROOT** and **Q** using the following equations:

$$ROOT = SQRT((ABS(r) + (CABS(r,i))/2)$$

$$Q = i/(2 * ROOT)$$

Then, the complex result is given as follows:

r	i	CSQRT((r,i))
≥0	any	(ROOT,Q)
<0	≥0	(Q,ROOT)
<0	<0	(-Q,-ROOT)

CONDITION VALUE SIGNALED

SS$_ROPRAND — Reserved operand. The MTH$CxSQRT procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms the complex square
C    root of a D-floating complex number using
C    MTH$CDSQRT and the FORTRAN random number
C    generator RAN.
C
C    Declare Z and Z_NEW as complex values. MTH$CDSQRT
C    will return the complex square root of
C    Z:    CALL MTH$CDSQRT(Z_NEW,Z).
C-

        COMPLEX*16 Z,Z_NEW
        COMPLEX*16 DCMPLX
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic CMPLX.
C-

        Z = DCMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r" and imaginary
C    part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' '

C+
C    Compute the complex complex square root of Z.
C-

        CALL MTH$CDSQRT(Z_NEW,Z)
        TYPE *, ' The complex square root of',z,' is',Z_NEW
        END
```

This FORTRAN example program demonstrates a procedure call to MTH$CDSQRT. The output generated by this program is as follows:

```
 The complex number z is (0.8535407185554504,0.2043401598930359)
 The complex square root of (0.8535407185554504,0.2043401598930359) is
(0.9303763973040062,0.1098158554350485)
```

MTH$CVT_x_x Convert One Double-Precision Value

The Convert One Double-Precision Value routines convert one double-precision value to the destination data type and return the result as a function value. MTH$CVT_D_G converts a D-floating value to G-floating and MTH$CVT_G_D converts a G-floating value to a D-floating value.

FORMAT

MTH$CVT_D_G *floating-point-input-val*
MTH$CVT_G_D *floating-point-input-val*

RETURNS

VMS usage: **floating_point**
type: **G_floating, D_floating**
access: **write only**
mechanism: **by value**

The converted value. MTH$CVT_D_G returns a G-floating value. MTH$CVT_G_D returns a D-floating value.

ARGUMENT

floating-point-input-val
VMS usage: **floating_point**
type: **D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value to be converted. The **floating-point-input-val** argument is the address of this input value. For MTH$CVT_D_G, the **floating-point-input-val** argument specifies a D-floating number. For MTH$CVT_G_D, the **floating-point-input-val** argument specifies a G-floating number.

DESCRIPTION

These procedures are designed to function as hardware conversion instructions. They fault on reserved operands. If floating-point overflow is detected, an error is signaled. If floating-point underflow is detected and floating-point underflow is enabled, an error is signaled.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CVT_x_x procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library.
MTH$_FLOUNDMAT	Floating-point underflow in Math Library.

MTH$CVT_xA_xA
Convert an Array of Double-Precision Values

The Convert an Array of Double-Precision Values routines convert a contiguous array of double-precision values to the destination data type and return the results as an array. MTH$CVT_DA_GA converts D-floating values to G-floating and MTH$CVT_GA_DA converts G-floating values to D-floating.

FORMAT

MTH$CVT_DA_GA *floating-point-input-array ,floating-point-dest-array [,array-size]*

MTH$CVT_GA_DA *floating-point-input-array ,floating-point-dest-array [,array-size]*

RETURNS

MTH$CVT_DA_GA and MTH$CVT_GA_DA return the address of the output array to the **floating-point-dest-array** argument.

ARGUMENTS

floating-point-input-array

VMS usage: **floating_point**
type: **D_floating, G_floating**
access: **read only**
mechanism: **by reference, array reference**

Input array of values to be converted. The **floating-point-input-array** argument is the address of an array of floating-point numbers. For MTH$CVT_DA_GA, **floating-point-input-array** specifies an array of D-floating numbers. For MTH$CVT_GA_DA, **floating-point-input-array** specifies an array of a G-floating numbers.

floating-point-dest-array

VMS usage: **floating_point**
type: **G_floating, D_floating**
access: **write only**
mechanism: **by reference, array reference**

Output array of converted values. The **floating-point-dest-array** argument is the address of an array of floating-point numbers. For MTH$CVT_DA_GA, **floating-point-dest-array** specifies an array of G-floating numbers. For MTH$CVT_GA_DA, **floating-point-dest-array** specifies an array of D-floating numbers.

array-size
VMS usage: **longword_signed**
type: **longword (signed)**
access: **read only**
mechanism: **by reference**

Number of array elements to be converted. The default value is 1. The **array-size** argument is the address of a longword containing this number of elements.

DESCRIPTION

These procedures are designed to function as hardware conversion instructions. They fault on reserved operands. If floating-point overflow is detected, an error is signaled. If floating-point underflow is detected and floating-point underflow is enabled, an error is signaled.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$CVT_xA_xA procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library.
MTH$_FLOUNDMAT	Floating-point underflow in Math Library.

MTH$xEXP Exponential

The Exponential routine returns the exponential of the input value.

FORMAT

MTH$EXP *floating-point-input-value*
MTH$DEXP *floating-point-input-value*
MTH$GEXP *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$EXP_R4
MTH$DEXP_R6
MTH$GEXP_R6

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The exponential of **floating-point-input-value**. MTH$EXP returns an F-floating number. MTH$DEXP returns a D-floating number. MTH$GEXP returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number. For MTH$EXP, **floating-point-input-value** specifies an F-floating number. For MTH$DEXP, **floating-point-input-value** specifies a D-floating number. For MTH$GEXP, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

The exponential of x is computed as:

Value of x	Value Returned
$X > c(z)$	Overflow occurs
$X \leq -c(z)$	0
$\lvert X\rvert < 2^{-(f+1)}$	1
Otherwise	$2^Y * 2^U * 2^W$

where:

$Y = INTEGER(x * ln2(E))$

$V = FRAC(x * ln2(E)) * 16$

$U = INTEGER(V)/16$

$W = FRAC(V)/16$

2^W = polynomial approximation of degree 4,8, or 8 for z = F, D, or G.

See also the section on the hyperbolic cosine for definitions of f and c(z).

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HEXP.

CONDITION VALUES SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xEXP routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_FLOOVEMAT

Floating-point overflow in Math Library: **floating-point-input-value** is greater than *yyy*; LIB$SIGNAL copies the reserved operand to the signal mechanism vector. The result is the reserved operand -0.0 unless a condition handler changes the signal mechanism vector.

The values of *yyy* are approximately:

MTH$EXP—88.029
MTH$DEXP—88.029
MTH$GEXP—709.089

MTH$_FLOUNDMAT	Floating-point underflow in Math Library: **floating-point-input-value** is less than or equal to *yyy* and the caller (CALL or JSB) has set hardware floating-point underflow enable. The result is set to 0.0. If the caller has not enabled floating-point underflow (the default), a result of 0.0 is returned but no error is signaled. The values of *yyy* are approximately: MTH$EXP— –88.722 MTH$DEXP— –88.722 MTH$GEXP— –709.774

EXAMPLE

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    FLOATING_POINT.
*
*  Calls MTH$EXP using a Floating Point data type.
*  Calls MTH$DEXP using a Double Floating Point data type.
*
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FLOAT_PT     COMP-1.
01 ANSWER_F     COMP-1.
01 DOUBLE_PT    COMP-2.
01 ANSWER_D     COMP-2.
PROCEDURE DIVISION.
P0.
        MOVE 12.34 TO FLOAT_PT.
        MOVE 3.456 TO DOUBLE_PT.

        CALL "MTH$EXP" USING BY REFERENCE FLOAT_PT GIVING ANSWER_F.
        DISPLAY " MTH$EXP of ", FLOAT_PT CONVERSION, " is ",
                                               ANSWER_F CONVERSION.

        CALL "MTH$DEXP" USING BY REFERENCE DOUBLE_PT GIVING ANSWER_D.
        DISPLAY " MTH$DEXP of ", DOUBLE_PT CONVERSION, " is ",
                                               ANSWER_D CONVERSION .
        STOP RUN.
```

This sample program demonstrates calls to MTH$EXP and MTH$DEXP from COBOL.

The output generated by this program is as follows:

```
MTH$EXP of  1.234000E+01 is  2.286620E+05
MTH$DEXP of  3.456000000000000E+00 is
3.168996280537917E+01
```

MTH$HACOS Arc Cosine of Angle Expressed in Radians (H-floating Value)

Given the cosine of an angle, the Arc Cosine of Angle Expressed in Radians (H-floating Value) routine returns that angle (in radians) in H-floating-point precision.

FORMAT

MTH$HACOS *h-radians ,cosine*

jsb entries

MTH$HACOS_R8

RETURNS

None.

ARGUMENTS

h-radians

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in radians) whose cosine is specified by **cosine**. The **h-radians** argument is the address of an H-floating number that is this angle. MTH$HACOS writes the address of the angle into **h-radians**.

cosine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The cosine of the angle whose value (in radians) is to be returned. The **cosine** argument is the address of a floating-point number that is this cosine. The absolute value of **cosine** must be less than or equal to 1. For MTH$HACOS, **cosine** specifies an H-floating number.

MTH$HACOS

DESCRIPTION

The angle in radians whose cosine is X is computed as:

Value of Cosine	Value Returned
0	$\pi/2$
1	0
-1	π
$0 < X < 1$	$zATAN(zSQRT(1 - X^2)/X)$, where zATAN and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$-1 < X < 0$	$zATAN(zSQRT(1 - X^2)/X) + \pi$
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xACOS routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **cosine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HACOSD Arc Cosine of Angle Expressed in Degrees (H-Floating Value)

Given the cosine of an angle, the Arc Cosine of Angle Expressed in Degrees (H-Floating Value) routine returns that angle (in degrees) as an H-floating value.

FORMAT

MTH$HACOSD *h-degrees ,cosine*

jsb entries

MTH$HACOSD_R8

RETURNS

None.

ARGUMENTS

h-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in degrees) whose cosine is specified by **cosine**. The **h-degrees** argument is the address of an H-floating number that is this angle. MTH$HACOSD writes the address of the angle into **h-degrees**.

cosine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Cosine of the angle whose value (in degrees) is to be returned. The **cosine** argument is the address of a floating-point number that is this cosine. The absolute value of **cosine** must be less than or equal to 1. For MTH$HACOSD, **cosine** specifies an H-floating number.

DESCRIPTION

The angle in degrees whose cosine is X is computed as:

Value of Cosine	Angle Returned
0	90
1	0
-1	180
$0 < X < 1$	$zATAND(zSQRT(1 - X^2)/X)$, where zATAND and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$-1 < X < 0$	$zATAND(zSQRT(1 - X^2)/X) + 180$
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

CONDITION VALUES SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xACOSD routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_INVARGMAT

Invalid argument. The absolute value of **cosine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HASIN Arc Sine in Radians (H-floating Value)

Given the sine of an angle, the Arc Sine in Radians (H-floating Value) routine returns that angle (in radians) as an H-floating value.

FORMAT

MTH$HASIN *h-radians ,sine*

jsb entries

MTH$HASIN_R8

RETURNS

None.

ARGUMENTS

h-radians

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in radians) whose sine is specified by **sine**. The **h-radians** argument is the address of an H-floating number that is this angle. MTH$HASIN writes the address of the angle into **h-radians**.

sine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The sine of the angle whose value (in radians) is to be returned. The **sine** argument is the address of a floating-point number that is this sine. The absolute value of **sine** must be less than or equal to 1. For MTH$HASIN, **sine** specifies an H-floating number.

MTH$HASIN

DESCRIPTION

The angle in radians whose sine is X is computed as:

Value of Sine	Angle Returned
0	0
1	$\pi/2$
-1	$-\pi/2$
$0 < \|X\| < 1$	$zATAN(X/zSQRT(1 - X^2))$, where zATAN and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xASIN routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **sine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HASIND Arc Sine in Degrees (H-Floating Value)

Given the sine of an angle, the Arc Sine in Degrees (H-Floating Value) routine returns that angle (in degrees) as an H-floating value.

FORMAT

MTH$HASIND *h-degrees ,sine*

jsb entries

MTH$HASIND_R8

RETURNS

None.

ARGUMENTS

h-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in degrees) whose sine is specified by **sine**. The **h-degrees** argument is the address of an H-floating number that is this angle. MTH$HASIND writes the address of the angle into **h-degrees**.

sine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Sine of the angle whose value (in degrees) is to be returned. The **sine** argument is the address of a floating-point number that is this sine. The absolute value of **sine** must be less than or equal to 1. For MTH$HASIND, **sine** specifies an H-floating number.

MTH$HASIND

DESCRIPTION

The angle in degrees whose sine is X is computed as:

Value of Sine	Value Returned
0	0
1	90
−1	−90
$0 < \|X\| < 1$	$zATAND(X/zSQRT(1 - X^2))$, where zATAND and zSQRT are the Math Library arc tangent and square root routines, respectively, of the appropriate data type
$1 < \|X\|$	The error MTH$_INVARGMAT is signaled

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xASIND routine encountered a floating point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of one and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. The absolute value of **sine** is greater than 1. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HATAN Arc Tangent in Radians (H-floating Value)

Given the tangent of an angle, the Arc Tangent in Radians (H-floating Value) routine returns that angle (in radians) as an H-floating value.

FORMAT

MTH$HATAN *h-radians ,tangent*

jsb entries

MTH$HATAN_R8

RETURNS

None.

ARGUMENTS

h-radians

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in radians) whose tangent is specified by **tangent**. The **h-radians** argument is the address of an H-floating number that is this angle. MTH$HATAN writes the address of the angle into **h-radians.**

tangent

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The tangent of the angle whose value (in radians) is to be returned. The **tangent** argument is the address of a floating-point number that is this tangent. For MTH$HATAN, **tangent** specifies an H-floating number.

DESCRIPTION

In radians, the computation of the arc tangent function is based on the following identities:

$$\arctan(X) = X - X^3/3 + X^5/5 - X^7/7 + ...$$

$$\arctan(X) = X + X * Q(X^2),$$
where $Q(Y) = -Y/3 + Y^2/5 - Y^3/7 + ...$

$$\arctan(X) = X * P(X^2),$$
where $P(Y) = 1 - Y/3 + Y^2/5 - Y^3/7 + ...$

$$\arctan(X) = \pi/2 - \arctan(1/X)$$

$$\arctan(X) = \arctan(A) + \arctan((X - A)/(1 + A * X))$$
for any real A

MTH$HATAN

The angle in radians whose tangent is X is computed as:

Value of *X*	Angle Returned
$0 \leq X \leq 3/32$	$X + X * Q(X^2)$
$3/32 < X \leq 11$	$ATAN(A) + V * (P(V^2))$, where A and ATAN(A) are chosen by table lookup and $V = (X - A)/(1 + A * X)$
$11 < X$	$\pi/2 - W * (P(W^2))$ where $W = 1/X$
$X < 0$	$-zATAN(\|X\|)$

CONDITION VALUE SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xATAN routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$HATAND Arc Tangent in Degrees (H-floating Value)

Given the tangent of an angle, the Arc Tangent in Degrees (H-floating Value) routine returns that angle (in degrees) as an H-floating point value.

FORMAT

MTH$HATAND *h-degrees ,tangent*

jsb entries

MTH$HATAND_R8

RETURNS

None.

ARGUMENTS

h-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in degrees) whose tangent is specified by **tangent**. The **h-degrees** argument is the address of an H-floating number that is this angle. MTH$HATAND writes the address of the angle into **h-degrees**.

tangent

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The tangent of the angle whose value (in degrees) is to be returned. The **tangent** argument is the address of a floating-point number that is this tangent. For MTH$HATAND, **tangent** specifies an H-floating number.

DESCRIPTION

The computation of the arc tangent function is based on the following identities:

$$\arctan(X) = 180/\pi * (X - X^3/3 + X^5/5 - X^7/7 + ...)$$

$$\arctan(X) = 64 * X + X * Q(X^2),$$
where $Q(Y) = 180/\pi * [(1 - 64 * \pi/180) - Y/3 + Y^2/5 - Y^3/7 + Y^4/9...]$

$$\arctan(X) = X * P(X^2),$$
where $P(Y) = 180/\pi * [1 - Y/3 + Y^2/5 - Y^3/7 + Y^4/9...]$

$$\arctan(X) = 90 - \arctan(1/X)$$

$$\arctan(X) = \arctan(A) + \arctan((X - A)/(1 + A * X))$$

MTH$HATAND

The angle in degrees whose tangent is X is computed as:

Tangent	Angle Returned
$X \leq 3/32$	$64 * X + X * Q(X^2)$
$3/32 < X \leq 11$	$ATAND(A) + V * P(V^2)$, where A and ATAND(A) are chosen by table lookup and $V = (X - A)/(1 + A * X)$
$11 < X$	$90 - W * (P(W^2))$, where $W = 1/X$
$X < 0$	$-zATAND(\|X\|)$

CONDITION VALUE SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xATAND routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$HATAN2 Arc Tangent in Radians (H-floating Value) with Two Arguments

Given **sine** and **cosine**, the Arc Tangent in Radians (H-floating Value) with Two Arguments routine returns the angle (in radians) as an H-floating value whose tangent is given by the quotient of **sine** and **cosine**, (**sine/cosine**).

FORMAT

MTH$HATAN2 *h-radians ,sine ,cosine*

RETURNS

None.

ARGUMENTS

h-radians

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in radians) whose tangent is specified by (**sine/cosine**). The **h-radians** argument is the address of an H-floating number that is this angle. MTH$HATAN2 writes the address of the angle into **h-radians.**

sine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Dividend. The **sine** argument is the address of a floating-point number that is this dividend. For MTH$HATAN2, **sine** specifies an H-floating number.

cosine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Divisor. The **cosine** argument is the address of a floating-point number that is this divisor. For MTH$HATAN2, **cosine** specifies an H-floating number.

DESCRIPTION

The angle in radians whose tangent is Y/X is computed as follows, where f is defined in the description of MTH$zCOSH.

Value of Input Arguments	Angle Returned
$X = 0$ *or* $Y/X > 2^{(f+1)}$	$\pi/2 * (signY)$
$X > 0$ *and* $Y/X \leq 2^{(f+1)}$	$zATAN(Y/X)$
$X < 0$ *and* $Y/X \leq 2^{(f+1)}$	$\pi * (signY) + zATAN(Y/X)$

CONDITION VALUES SIGNALED

SS$_ROPRAND — Reserved operand. The MTH$HATAN2 routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_INVARGMAT — Invalid argument. Both **cosine** and **sine** are zero. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HATAND2 Arc Tangent in Degrees (H-floating Value) with Two Arguments

Given **sine** and **cosine**, MTH$xHTAND2 returns the angle (in degrees) whose tangent is given by the quotient of **sine** and **cosine**, **(sine/cosine)**.

FORMAT

MTH$HATAND2 *h-degrees ,sine ,cosine*

RETURNS

None.

ARGUMENTS

h-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Angle (in degrees) whose tangent is specified by **(sine/cosine)**. The **h-degrees** argument is the address of an H-floating number that is this angle. MTH$HATAND2 writes the address of the angle into **h-degrees**.

sine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Dividend. The **sine** argument is the address of a floating-point number that is this dividend. For MTH$HATAND2, **sine** specifies an H-floating number.

cosine

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Divisor. The **cosine** argument is the address of a floating-point number that is this divisor. For MTH$HATAND2, **cosine** specifies an H-floating number.

MTH$HATAND2

DESCRIPTION

The angle in degrees whose tangent is Y/X is computed below. The value of f is defined in the description of MTH$zCOSH.

Value of Input Arguments	Angle Returned
$X = 0$ *or* $Y/X > 2^{(f+1)}$	$90 * (signY)$
$X > 0$ *and* $Y/X \leq 2^{(f+1)}$	$zATAND(Y/X)$
$X < 0$ *and* $Y/X \leq 2^{(f+1)}$	$180 * (signY) + zATAND(Y/X)$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HATAND2 routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument. Both **cosine** and **sine** are zero. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HATANH Hyperbolic Arc Tangent (H-floating Value)

Given the hyperbolic tangent of an angle, the Hyperbolic Arc Tangent (H-floating Value) routine returns the hyperbolic arc tangent (as an H-floating value) of that angle.

FORMAT

MTH$HATANH *h-atanh ,hyperbolic-tangent*

RETURNS

None.

ARGUMENTS

h-atanh

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Hyperbolic arc tangent of the hyperbolic tangent specified by **hyperbolic-tangent**. The **h-atanh** argument is the address of an H-floating number that is this hyperbolic arc tangent. MTH$HATANH writes the address of the hyperbolic arc tangent into **h-atanh**.

hyperbolic-tangent

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Hyperbolic tangent of an angle. The **hyperbolic-tangent** argument is the address of a floating-point number that is this hyperbolic tangent. For MTH$HATANH, **hyperbolic-tangent** specifies an H-floating number.

DESCRIPTION

The hyperbolic arc tangent function is computed as follows:

Value of x	Value Returned
$\lvert X\rvert < 1$	$zATANH(X) = zLOG((X+1)/(X-1))/2$
$\lvert X\rvert \geq 1$	An invalid argument is signaled

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xATANH routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_INVARGMAT	Invalid argument: $\lvert X \rvert \geq 1$. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HCOS Cosine of Angle Expressed in Radians (H-floating Value)

The Cosine of Angle Expressed in Radians (H-floating Value) routine returns the cosine of a given angle (in radians) as an H-floating value.

FORMAT

MTH$HCOS *h-cosine ,angle-in-radians*

jsb entries

MTH$HCOS_R5

RETURNS

None.

ARGUMENTS

h-cosine

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Cosine of the angle specified by **angle-in-radians**. The **h-cosine** argument is the address of an H-floating number that is this cosine. MTH$HCOS writes the address of the cosine into **h-cosine**.

angle-in-radians

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The angle in radians. The **angle-in-radians** argument is the address of a floating-point number. For MTH$HCOS, **angle-in-radians** specifies an H-floating number.

DESCRIPTION

See the MTH$xSINCOS routine for the algorithm used to compute the cosine.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HCOS procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$HCOSD Cosine of Angle Expressed in Degrees (H-floating Value)

The Cosine of Angle Expressed in Degrees (H-floating Value) routine returns the cosine of a given angle (in degrees) as an H-floating value.

FORMAT

MTH$HCOSD *h-cosine ,angle-in-degrees*

jsb entries

MTH$HCOSD_R5

RETURNS

None.

ARGUMENTS

h-cosine

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Cosine of the angle specified by **angle-in-degrees**. The **h-cosine** argument is the address of an H-floating number that is this cosine. MTH$HCOSD writes this cosine into **h-cosine**.

angle-in-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number. For MTH$HCOSD, **angle-in-degrees** specifies an H-floating number.

DESCRIPTION

See the MTH$SINCOSD routine for the algorithm used to compute the cosine.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HCOSD procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$HCOSH Hyperbolic Cosine (H-floating Value)

The Hyperbolic Cosine routine returns the hyperbolic cosine of the input value as an H-floating value.

FORMAT

MTH$HCOSH *h-cosh ,floating-point-input-value*

RETURNS

None.

ARGUMENTS

h-cosh
VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Hyperbolic cosine of the input value specified by **floating-point-input-value**. The **h-cosh** argument is the address of an H-floating number that is this hyperbolic cosine. MTH$HCOSH writes the address of the hyperbolic cosine into **h-cosh**.

floating-point-input-value
VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of this input value. For MTH$HCOSH, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

Computation of the hyperbolic cosine depends on the magnitude of the input argument. The range of the function is partitioned using four data-type-dependent constants: a(z), b(z), and c(z). The subscript *z* indicates the data type. The constants depend on the number of exponent bits (*e*) and the number of fraction bits (*f*) associated with the data type (z).

The values of *e* and *f* are as follows:

$$e = 15$$

$$f = 113$$

The values of the constants in terms of e and f are:

Variable	Value
a(z)	$2^{-f/2}$
b(z)	$(f+1)/2 * \ln(2)$
c(z)	$2^{e-1} * \ln(2)$

Based on the above definitions, zCOSH(X) is computed as follows:

Value of X	Value Returned
$\|X\| < a(z)$	1
$a(z) \leq \|X\| < .25$	Computed using a power series expansion in $\|X\|^2$
$.25 \leq \|X\| < b(z)$	$(zEXP(\|X\|) + 1/zEXP(\|X\|))/2$
$b(z) \leq \|X\| < c(z)$	$zEXP(\|X\|)/2$
$c(z) \leq \|X\|$	Overflow occurs

CONDITION VALUES SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$HCOSH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_FLOOVEMAT

Floating-point overflow in Math Library: the absolute value of **floating-point-input-value** is greater than about *yyy*; LIB$SIGNAL copies the reserved operand to the signal mechanism vector. The result is the reserved operand -0.0 unless a condition handler changes the signal mechanism vector. The value of *yyy* is 11356.523.

MTH$HEXP Exponential (H-floating Value)

The Exponential routine returns the exponential of the input value as an H-floating value.

FORMAT

MTH$HEXP *h-exp ,floating-point-input-value*

jsb entries

MTH$HEXP_R6

RETURNS

None.

ARGUMENTS

h-exp

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Exponential of the input value specified by **floating-point-input-value**. The **h-exp** argument is the address of an H-floating number that is this exponential. MTH$HEXP writes the address of the exponential into **h-exp**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number. For MTH$HEXP, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

The exponential of x is computed as:

Value of x	Value Returned
$x > c(z)$	Overflow occurs
$x \leq -c(z)$	0
$\lvert x \rvert < 2^{-(f+1)}$	1
Otherwise	$2^Y * 2^U * 2^W$

where:

$$Y = INTEGER(x * ln2(E))$$

$$V = FRAC(x * ln2(E)) * 16$$

$$U = INTEGER(V)/16$$

MTH$HEXP

$W = FRAC(V)/16$

2^W = polynomial approximation of degree 14 for z = H.

See also the section on the hyperbolic cosine for definitions of f and c(z).

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xEXP routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: **floating-point-input-value** is greater than *yyy;* LIB$SIGNAL copies the reserved operand to the signal mechanism vector. The result is the reserved operand -0.0 unless a condition handler changes the signal mechanism vector. The value of *yyy* is approximately 11355.830 for MTH$HEXP.
MTH$_FLOUNDMAT	Floating-point underflow in Math Library: **floating-point-input-value** is less than or equal to *yyy* and the caller (CALL or JSB) has set hardware floating-point underflow enable. The result is set to 0.0. If the caller has not enabled floating-point underflow (the default), a result of 0.0 is returned but no error is signaled. The value of *yyy* is approximately −11356.523 for MTH$HEXP.

MTH$HLOG Natural Logarithm (H-floating Value)

The Natural Logarithm (H-floating Value) routine returns the natural (base e) logarithm of the input argument as an H-floating value.

FORMAT

MTH$HLOG *h-natlog ,floating-point-input-value*

jsb entries

MTH$HLOG_R8

RETURNS

None.

ARGUMENTS

h-natlog

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Natural logarithm of **floating-point-input-value**. The **h-natlog** argument is the address of an H-floating number that is this natural logarithm. MTH$HLOG writes the address of this natural logarithm into **h-natlog**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this value. For MTH$HLOG, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

Computation of the natural logarithm routine is based on the following:

1. $\ln(X * Y) = \ln(X) + \ln(Y)$
2. $\ln(1 + X) = X - X^2/2 + X^3/3 - X^4/4...$
   for $|X| < 1$
3. $\ln(X) = \ln(A) + 2 * (V + V^3/3 + V^5/5 + V^7/7...)$
   where $V = (X - A)/(X + A)$, $A > 0$,
   and $p(y) = 2 * (1 + y/3 + y^2/5...)$

For $x = 2^n * f$, where n is an integer and f is in the interval of 0.5 to 1, define the following quantities:

$$If\ n \geq 1,\ then\ N = n - 1\ and\ F = 2f$$

$$If\ n \leq 0,\ then\ N = n\ and\ F = f$$

From (1) above it follows that:

4 $\ln(X) = N * \ln(2) + \ln(F)$

Based on the above relationships, zLOG is computed as follows:

1 If $|F - 1| < 2^{-5}$,
$zLOG(X) = N * zLOG(2) + W + W * p(W)$,
where W = F-1.

2 Otherwise,
$zLOG(X) = N * zLOG(2) + zLOG(A) + V * p(V^2)$,
where $V = (F - A)/(F + A)$ and A and zLOG(A) are obtained by table look up.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HLOG procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HLOG2 Base 2 Logarithm (H-floating Value)

The Base 2 Logarithm (H-floating Value) routine returns the base 2 logarithm of the input value specified by **floating-point-input-value** as an H-floating value.

FORMAT

MTH$HLOG2 *h-log2 ,floating-point-input-value*

RETURNS

None.

ARGUMENTS

h-log2

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Base 2 logarithm of **floating-point-input-value**. The **h-log2** argument is the address of an H-floating number that is this base 2 logarithm. MTH$HLOG2 writes the address of this logarithm into **h-log2**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this input value. For MTH$HLOG2, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

The base 2 logarithm function is computed as follows:

$$zLOG2(X) = zLOG2(E) * zLOG(X)$$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HLOG2 procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HLOG10 Common Logarithm (H-floating Value)

The Common Logarithm (H-floating Value) routine returns the common (base 10) logarithm of the input argument as an H-floating value.

FORMAT

MTH$HLOG10 *h-log10 ,floating-point-input-value*

jsb entries

MTH$HLOG10_R8

RETURNS

None.

ARGUMENTS

h-log10

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Common logarithm of the input value specified by **floating-point-input-value**. The **h-log10** argument is the address of an H-floating number that is this common logarithm. MTH$HLOG10 writes the address of the common logarithm into **h-log10**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number. For MTH$HLOG10, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

The common logarithm function is computed as follows:

$$zLOG10(X) = zLOG10(E) * zLOG(X)$$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HLOG10 procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HSIN Sine of Angle Expressed in Radians (H-floating Value)

The Sine of Angle Expressed in Radians (H-floating Value) routine returns the sine of a given angle (in radians) as an H-floating value.

FORMAT

MTH$HSIN *h-sine ,angle-in-radians*

jsb entries

MTH$HSIN_R5

RETURNS

None.

ARGUMENTS

h-sine

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

The sine of the angle specified by **angle-in-radians**. The **h-sine** argument is the address of an H-floating number that is this sine. MTH$HSIN writes the address of the sine into **h-sine**.

angle-in-radians

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Angle (in radians). The **angle-in-radians** argument is the address of a floating-point number that is this angle. For MTH$HSIN, **angle-in-radians** specifies an H-floating number.

DESCRIPTION

See the MTH$SINCOS routine for the algorithm used to compute this sine.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HSIN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$HSIND Sine of Angle Expressed in Degrees (H-floating Value)

The Sine of Angle Expressed in Degrees (H-floating Value) routine returns the sine of a given angle (in degrees) as an H-floating value.

FORMAT

MTH$HSIND *h-sine ,angle-in-degrees*

jsb entries

MTH$HSIND_R5

RETURNS

None.

ARGUMENTS

h-sine

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Sine of the angle specified by **angle-in-degrees**. The **h-sine** argument is the address of an H-floating number that is this sine. MTH$HSIND writes the address of the angle into **h-sine**.

angle-in-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number that is this angle. For MTH$HSIND, **angle-in-degrees** specifies an H-floating number.

DESCRIPTION

See MTH$SINCOSD for the algorithm used to compute the sine.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HSIND procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased ecponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOUNDMAT	Floating-point underflow in Math Library. The absolute value of the input angle is less than $180/\pi * 2^{-m}$ (where m = 16,384 for H-floating).

MTH$HSINH Hyperbolic Sine (H-floating Value)

The Hyperbolic Sine (H-floating Value) routine returns the hyperbolic sine of the input value specified by **floating-point-input-value** as an H-floating value.

FORMAT

MTH$HSINH *h-sinh ,floating-point-input-value*

RETURNS

None.

ARGUMENTS

h-sinh

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Hyperbolic sine of the input value specified by **floating-point-input-value**. The **h-sinh** argument is the address of an H-floating number that is this hyperbolic sine. MTH$HSINH writes the address of the hyperbolic sine into **h-sinh**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this value. For MTH$HSINH, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

Computation of the hyperbolic sine function depends on the magnitude of the input argument. The range of the function is partitioned using four data type dependent constants: a(z), b(z), and c(z). The subscript *z* indicates the data type. The constants depend on the number of exponent bits (*e*) and the number of fraction bits (*f*) associated with the data type (z).

The values of *e* and *f* are as follows:

$$e = 15$$

$$f = 113$$

The values of the constants in terms of e and f are:

Variable	Value
a(z)	$2^{(-f/2)}$
b(z)	$(f+1)/2 * \ln(2)$
c(z)	$2^{e-1} * \ln(2)$

Based on the above definitions, zSINH(X) is computed as follows:

Value of X	Value Returned
$\lvert X\rvert < a(z)$	X
$a(z) \leq \lvert X\rvert < 1.0$	zSINH(X) is computed using a power series expansion in $\lvert X\rvert^2$
$1.0 \leq \lvert X\rvert < b(z)$	$(zEXP(X) - zEXP(-X))/2$
$b(z) \leq \lvert X\rvert < c(z)$	$SIGN(X) * zEXP(\lvert X\rvert)/2$
$c(z) \leq \lvert X\rvert$	Overflow occurs

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HSINH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library: the absolute value of **floating-point-input-value** is greater than *yyy*. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1. The value of *yyy* is approximately 11356.523.

MTH$HSQRT Square Root (H-floating Value)

The Square Root (H-floating Value) routine returns the square root of the input value **floating-point-input-value** as an H-floating value.

FORMAT

MTH$HSQRT *h-sqrt ,floating-point-input-value*

jsb entries

MTH$HSQRT_R8

RETURNS

None.

ARGUMENTS

h-sqrt

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Square root of the input value specified by **floating-point-input-value**. The **h-sqrt** argument is the address of an H-floating number that is this square root. MTH$HSQRT writes the address of the square root into **h-sqrt**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

Input value. The **floating-point-input-value** argument is the address of a floating-point number that contains this input value. For MTH$HSQRT, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

The square root of X is computed as follows:

If $X < 0$, an error is signaled.

Let $X = 2^K * F$

where:

K is the exponential part of the floating-point data

F is the fractional part of the floating-point data

If K is even:

$X = 2^{(2*P)} * F$,
$zSQRT(X) = 2^P * zSQRT(F)$,
$1/2 \leq F < 1$, where P = K/2

If K is odd:

$X = 2^{(2*P+1)} * F = 2^{(2*P+2)} * (F/2)$,

$zSQRT(X) = 2^{(P+1)} * zSQRT(F/2)$,

$1/4 \leq F/2 < 1/2$, where p = (K-1)/2

Let $F' = A * F + B$, when K is even:

A = 0.95F6198 (hex)

B = 0.6BA5918 (hex)

Let $F' = A * (F/2) + B$, when K is odd:

A = 0.D413CCC (hex)

B = 0.4C1E248 (hex)

Let K' = P, when K is even

Let K' = P+1, when K is odd

Let $Y[0] = 2^{K'} * F'$ be a straight line approximation within the given interval using coefficients A and B which minimize the absolute error at the midpoint and endpoint.

Starting with Y[0], n Newton-Raphson iterations are performed:

$Y[n+1] = 1/2 * (Y[n] + X/Y[n])$

where n = 5 for H-floating.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HSQRT procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_SQUROONEG	Square root of negative number. Argument **floating-point-input-value** is less than 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$HTAN Tangent of Angle Expressed in Radians (H-floating Value)

The Tangent of Angle Expressed in Radians (H-floating Value) routine returns the tangent of a given angle (in radians) as an H-floating value.

FORMAT

MTH$HTAN *h-tan ,angle-in-radians*

jsb entries

MTH$HTAN_R5

RETURNS

None.

ARGUMENTS

h-tan

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Tangent of the angle specified by **angle-in-radians**. The **h-tan** argument is the address of an H-floating number that is this tangent. MTH$HTAN writes the address of the tangent into **h-tan**.

angle-in-radians

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input angle (in radians). The **angle-in-radians** argument is the address of a floating-point number that is this angle. For MTH$HTAN, **angle-in-radians** specifies an H-floating number.

DESCRIPTION

When the input argument is expressed in radians, the tangent function is computed as follows:

1. If $|X| < 2^{(-f/2)}$, then $zTAN(X) = X$ (see the section on MTH$zCOSH for the definition of *f*)
2. Otherwise, call MTH$zSINCOS to obtain zSIN(X) and zCOS(X); then
   a. If $zCOS(X) = 0$, signal overflow
   b. Otherwise, $zTAN(X) = zSIN(X)/zCOS(X)$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HTAN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in math library.

MTH$HTAND Tangent of Angle Expressed in Degrees (H-floating Value)

The Tangent of Angle Expressed in Degrees (H-floating Value) routine returns the tangent of a given angle (in degrees) as an H-floating value.

FORMAT

MTH$HTAND *h-tan ,angle-in-degrees*

jsb entries

MTH$HTAND_R5

RETURNS

None.

ARGUMENTS

h-tan

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Tangent of the angle specified by **angle-in-degrees**. The **h-tan** argument is the address of an H-floating number that is this tangent. MTH$HTAND writes the address of the tangent into **h-tan**.

angle-in-degrees

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number which is this angle. For MTH$HTAND, **angle-in-degrees** specifies an H-floating number.

DESCRIPTION

When the input argument is expressed in degrees, the tangent function is computed as follows:

1. If $|X| < (180/\pi)*2^{(-2/(e-1))}$ and underflow signaling is enabled, underflow is signaled (see the section on MTH$zCOSH for the definition of e).
2. Otherwise, if $|X| < (180/\pi) * 2^{(-f/2)}$, then $zTAND(X) = (\pi/180) * X$. See the description of MTH$zCOSH for the definition of f.
3. Otherwise, call MTH$zSINCOSD to obtain zSIND(X) and zCOSD(X).
   a. Then, if $zCOSD(X) = 0$, signal overflow
   b. Else, $zTAND(X) = zSIND(X)/zCOSD(X)$

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$HTAND procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in math library.

MTH$HTANH Compute the Hyperbolic Tangent (H-floating Value)

The Compute the Hyperbolic Tangent (H-floating Value) routine returns the hyperbolic tangent of the input value as an H-floating value.

FORMAT

MTH$HTANH *h-tanh ,floating-point-input-value*

RETURNS

None.

ARGUMENTS

h-tanh

VMS usage: **floating_point**
type: **H_floating**
access: **write only**
mechanism: **by reference**

Hyperbolic tangent of the value specified by **floating-point-input-value**. The **h-tanh** argument is the address of a H-floating number that is this hyperbolic tangent. MTH$HTANH writes the address of the hyperbolic tangent into **h-tanh**.

floating-point-input-value

VMS usage: **floating_point**
type: **H_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that contains this input value. For MTH$HTANH, **floating-point-input-value** specifies an H-floating number.

DESCRIPTION

For MTH$HTANH, the hyperbolic tangent of X is computed using a value of 56 for *g* and a value of 40 for *h*. The hyperbolic tangent of X is computed as follows:

Value of x	Hyperbolic Tangent Returned
$\lvert X\rvert \leq 2^{-g}$	X
$2^{-g} < \lvert X\rvert \leq 0.25$	$zSINH(X)/zCOSH(X)$
$0.25 < \lvert X\rvert < h$	$(zEXP(2*X)-1)/(zEXP(2*X)+1)$
$h \leq \lvert X\rvert$	$sign(X)*1$

CONDITION VALUE SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$HTANH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xIMAG Imaginary Part of a Complex Number

The Imaginary Part of a Complex Number routine returns the imaginary part of a complex number.

FORMAT

MTH$AIMAG *complex-number*
MTH$DIMAG *complex-number*
MTH$GIMAG *complex-number*

Each of the above three formats corresponds to one of the three floating-point complex types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Imaginary part of the input **complex-number**. MTH$AIMAG returns an F-floating number. MTH$DIMAG returns a D-floating number. MTH$GIMAG returns a G-floating number.

ARGUMENT

complex-number
VMS usage: **complex_number**
type: **F_floating complex, D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

The input complex number. The **complex-number** argument is the address of this floating-point complex number. For MTH$AIMAG, **complex-number** specifies an F-floating number. For MTH$DIMAG, **complex-number** specifies a D-floating number. For MTH$GIMAG, **complex-number** specifies a G-floating number.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xIMAG routine encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms the imaginary part of
C    a G-floating complex number using MTH$GIMAG
C    and the FORTRAN random number generator
C    RAN.
C
C    Declare Z as a complex value and MTH$GIMAG as
C    a REAL*8 value. MTH$GIMAG will return the imaginary
C    part of Z:   Z_NEW = MTH$GIMAG(Z).
C-

        COMPLEX*16 Z
        COMPLEX*16 DCMPLX
        REAL*8 R,I,MTH$GIMAG
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the
C    FORTRAN generic CMPLX.
C-

        R = RAN(M)
        I = RAN(M)
        Z = DCMPLX(R,I)

C+
C    Z is a complex number (r,i) with real part "r" and
C    imaginary part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' It has imaginary part',MTH$GIMAG(Z)
        END
```

This FORTRAN example demonstrates a procedure call to MTH$GIMAG. Because this example uses G-floating numbers, it must be compiled with the statement "FORTRAN/G filename".

The output generated by this program is as follows:

```
The complex number z is (0.8535407185554504,0.2043401598930359)
It has imaginary part  0.2043401598930359
```

MTH$xLOG Natural Logarithm

The Natural Logarithm routine returns the natural (base e) logarithm of the input argument.

FORMAT

MTH$ALOG *floating-point-input-value*
MTH$DLOG *floating-point-input-value*
MTH$GLOG *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ALOG_R5
MTH$DLOG_R8
MTH$GLOG_R8

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The natural logarithm of **floating-point-input-value**. MTH$ALOG returns an F-floating number. MTH$DLOG returns a D-floating number. MTH$GLOG returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this value. For MTH$ALOG, **floating-point-input-value** specifies an F-floating number. For MTH$DLOG, **floating-point-input-value** specifies a D-floating number. For MTH$GLOG, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

Computation of the natural logarithm routine is based on the following:

1. $\ln(X * Y) = \ln(X) + \ln(Y)$
2. $\ln(1 + X) = X - X^2/2 + X^3/3 - X^4/4 \ldots$
   for $|X| < 1$
3. $\ln(X) = \ln(A) + 2 * (V + V^3/3 + V^5/5 + V^7/7 \ldots)$
   $= ln(A) + V * p(V^2)$, where $V = (X - A)/(X + A)$,
   $A > 0$, and $p(y) = 2 * (1 + y/3 + y^2/5 \ldots)$

For $x = 2^n * f$, where n is an integer and f is in the interval of 0.5 to 1, define the following quantities:

$$If\ n \geq 1,\ then\ N = n - 1\ and\ F = 2f$$

$$If\ n \leq 0,\ then\ N = n\ and\ F = f$$

From (1) above it follows that:

4 $\ln(X) = N * \ln(2) + \ln(F)$

Based on the above relationships, zLOG is computed as follows:

1 If $|F - 1| < 2^{-5}$, $zLOG(X) = N * zLOG(2) + W + W * p(W)$, where W = F-1.

2 Otherwise, $zLOG(X) = N * zLOG(2) + zLOG(A) + V * p(V^2)$, where $V = (F - A)/(F + A)$ and A and zLOG(A) are obtained by table look up.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HLOG.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xLOG procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xLOG2 Base 2 Logarithm

The Base 2 Logarithm routine returns the base 2 logarithm of the input value specified by **floating-point-input-value**.

FORMAT

MTH$ALOG2 *floating-point-input-value*
MTH$DLOG2 *floating-point-input-value*
MTH$GLOG2 *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The base 2 logarithm of **floating-point-input-value**. MTH$ALOG2 returns an F-floating number. MTH$DLOG2 returns a D-floating number. MTH$GLOG2 returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this input value. For MTH$ALOG2, **floating-point-input-value** specifies an F-floating number. For MTH$DLOG2, **floating-point-input-value** specifies a D-floating number. For MTH$GLOG2, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

The base 2 logarithm function is computed as follows:

$$zLOG2(X) = zLOG2(E) * zLOG(X)$$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HLOG2.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xLOG2 procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xLOG10 Common Logarithm

The Common Logarithm routine returns the common (base 10) logarithm of the input argument.

FORMAT

MTH$ALOG10 *floating-point-input-value*
MTH$DLOG10 *floating-point-input-value*
MTH$GLOG10 *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$ALOG10_R5
MTH$DLOG10_R8
MTH$GLOG10_R8

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The common logarithm of **floating-point-input-value**. MTH$ALOG10 returns an F-floating number. MTH$DLOG10 returns a D-floating number. MTH$GLOG10 returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number. For MTH$ALOG10, **floating-point-input-value** specifies an F-floating number. For MTH$DLOG10, **floating-point-input-value** specifies a D-floating number. For MTH$GLOG10, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

The common logarithm function is computed as follows:

$$zLOG10(X) = zLOG10(E) * zLOG(X)$$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HLOG10.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xLOG10 procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_LOGZERNEG	Logarithm of zero or negative value. Argument **floating-point-input-value** is less than or equal to 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$RANDOM Random Number Generator, Uniformly Distributed

The Random Number Generator, Uniformly Distributed routine is a general random number generator.

FORMAT

MTH$RANDOM *seed*

RETURNS

VMS usage: **floating_point**
type: **F_floating**
access: **write only**
mechanism: **by value**

MTH$RANDOM returns an F-floating random number.

ARGUMENT

seed
VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

The integer seed, a 32-bit number whose high-order 24 bits are converted by MTH$RANDOM to an F-floating random number. The **seed** argument is the address of an unsigned longword that contains this integer seed. The seed is modified by each call to MTH$RANDOM.

DESCRIPTION

This routine must be called again to obtain the next pseudorandom number. The seed is updated automatically.

The result is a floating-point number that is uniformly distributed between 0.0 inclusively and 1.0 exclusively.

There are no restrictions on the seed, although it should be initialized to different values on separate runs in order to obtain different random sequences. MTH$RANDOM uses the following method to update the seed passed as the argument:

$$SEED = (69069 * SEED + 1)(modulo 2^{32})$$

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$RANDOM procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
RAND:   PROCEDURE OPTIONS (MAIN);
DECLARE FOR$SECNDS ENTRY (FLOAT BINARY (24))
                RETURNS (FLOAT BINARY (24));
DECLARE MTH$RANDOM ENTRY (FIXED BINARY (31))
                RETURNS (FLOAT BINARY (24));
DECLARE TIME FLOAT BINARY (24);
DECLARE SEED FIXED BINARY (31);
DECLARE I FIXED BINARY (7);
DECLARE RESULT FIXED DECIMAL (2);
        /* Get floating random time value      */
TIME = FOR$SECNDS (0E0);
        /* Convert to fixed                    */
SEED = TIME;
        /* Generate 100 random numbers between 1 and 10 */
DO I = 1 TO 100;
        RESULT = 1 + FIXED ( (10E0 * MTH$RANDOM (SEED) ),31 );
        PUT LIST (RESULT);
        END;
END RAND;
```

This PL/I program demonstrates the use of MTH$RANDOM. The value returned by FOR$SECNDS is used as the seed for the random-number generator to insure a different sequence each time the program is run. The random value returned is scaled so as to represent values between 1 and 10.

Because this program generates random numbers, the output generated will be different each time the program is executed. One example of the outut generated by this program is as follows:

```
7    4    6    5    9   10    5    5    3    8    8    1    3    1    3    2
4    4    2    4    4    8    3    8    9    1    7    1    8    6    9   10
1   10   10    6    7    3    2    2    1    2    6    6    3    9    5    8
6    2    3    6   10    8    5    5    4    2    8    5    9    6    4    2
8    5    4    9    8    7    6    6    8   10    9    5    9    4    5    7
1    2    2    3    6    5    2    3    4    4    8    9    2    8    5    5
3    8    1    5
```

MTH$xREAL Real Part of a Complex Number

The Real Part of a Complex Number routine returns the real part of a complex number.

FORMAT

MTH$REAL *complex-number*
MTH$DREAL *complex-number*
MTH$GREAL *complex-number*

Each of the above three formats accepts as input one of the three floating-point complex types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Real part of the complex number. MTH$REAL returns an F-floating number. MTH$DREAL returns a D-floating number. MTH$GREAL returns a G-floating number.

ARGUMENT

complex-number
VMS usage: **complex_number**
type: **F_floating complex, D_floating complex, G_floating complex**
access: **read only**
mechanism: **by reference**

The complex number whose real part is returned by MTH$REAL. The **complex-number** argument is the address of this floating-point complex number. For MTH$REAL, **complex-number** is an F-floating complex number. For MTH$DREAL, **complex-number** is a D-floating complex number. For MTH$GREAL, **complex-number** is a G-floating complex number.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xREAL procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

EXAMPLE

```
C+
C    This FORTRAN example forms the real
C    part of an F-floating complex number using
C    MTH$REAL and the FORTRAN random number
C    generator RAN.
C
C    Declare Z as a complex value and MTH$REAL as a
C    REAL*4 value.  MTH$REAL will return the real
C    part of Z:    Z_NEW = MTH$REAL(Z).
C-

        COMPLEX Z
        COMPLEX CMPLX
        REAL*4 MTH$REAL
        INTEGER M
        M = 1234567

C+
C    Generate a random complex number with the FORTRAN
C    generic CMPLX.
C-

        Z = CMPLX(RAN(M),RAN(M))

C+
C    Z is a complex number (r,i) with real part "r" and imaginary
C    part "i".
C-

        TYPE *, ' The complex number z is',z
        TYPE *, ' It has real part',MTH$REAL(Z)
        END
```

This FORTRAN example demonstrates the use of MTH$REAL. The output of this program is as follows:

```
The complex number z is (0.8535407,0.2043402)
It has real part  0.8535407
```

MTH$xSIN Sine of Angle Expressed in Radians

The Sine of Angle Expressed in Radians routine returns the sine of a given angle (in radians).

FORMAT

MTH$SIN *angle-in-radians*
MTH$DSIN *angle-in-radians*
MTH$GSIN *angle-in-radians*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$SIN_R4
MTH$DSIN_R7
MTH$GSIN_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Sine of the angle specified by **angle-in-radians**. MTH$SIN returns an F-floating number. MTH$DSIN returns a D-floating number. MTH$GSIN returns a G-floating number.

ARGUMENTS

angle-in-radians
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Angle (in radians). The **angle-in-radians** argument is the address of a floating-point number that is this angle. For MTH$SIN, **angle-in-radians** specifies an F-floating number. For MTH$DSIN, **angle-in-radians** specifies a D-floating number. For MTH$GSIN, **angle-in-radians** specifies a G-floating number.

DESCRIPTION

See the MTH$SINCOS routine for the algorithm used to compute this sine.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HSIN.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xSIN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xSINCOS Sine and Cosine of Angle Expressed in Radians

The Sine and Cosine of Angle Expressed in Radians routine returns the sine and cosine of a given angle (in radians).

FORMAT

MTH$SINCOS *angle-in-radians ,sine ,cosine*
MTH$DSINCOS *angle-in-radians ,sine ,cosine*
MTH$GSINCOS *angle-in-radians ,sine ,cosine*
MTH$HSINCOS *angle-in-radians ,sine ,cosine*

Each of the above four formats accepts as input one of the four floating-point types.

jsb entries

MTH$SINCOS_R5
MTH$DSINCOS_R7
MTH$GSINCOS_R7
MTH$HSINCOS_R7

Each of the above four JSB entries accepts as input one of the four floating-point types.

RETURNS

MTH$SINCOS, MTH$DSINCOS, MTH$GSINCOS, and MTH$HSINCOS return the sine and cosine of the input angle by reference in the **sine** and **cosine** arguments.

ARGUMENTS

angle-in-radians

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **read only**
mechanism: **by reference**

Angle (in radians) whose sine and cosine are to be returned. The **angle-in-radians** argument is the address of a floating-point number that is this angle. For MTH$SINCOS, **angle-in-radians** is an F-floating number. For MTH$DSINCOS, **angle-in-radians** is a D-floating number. For MTH$GSINCOS, **angle-in-radians** is a G-floating number. For MTH$HSINCOS, **angle-in-radians** is an H-floating number.

sine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **write only**
mechanism: **by reference**

Sine of the angle specified by **angle-in-radians**. The **sine** argument is the address of a floating-point number. MTH$SINCOS writes an F-floating number into **sine**. MTH$DSINCOS writes a D-floating number into **sine**. MTH$GSINCOS writes a G-floating number into **sine**. MTH$HSINCOS writes an H-floating number into **sine**.

cosine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **write only**
mechanism: **by reference**

Cosine of the angle specified by **angle-in-radians**. The **cosine** argument is the address of a floating-point number. MTH$SINCOS writes an F-floating number into **cosine**. MTH$DSINCOS writes a D-floating number into **cosine**. MTH$GSINCOS writes a G-floating number into **cosine**. MTH$HSINCOS writes an H-floating number into **cosine**.

DESCRIPTION

All routines with JSB entry points accept a single argument in R0:Rm, where *m*, which is defined below, is dependent on the data type.

Data Type	m
F_floating	0
D_floating	1
G_floating	1
H_floating	3

In general, Run-Time Library routines with JSB entry points return one value in R0:Rm. The MTH$SINCOS routine returns two values, however. The sine of **angle-in-radians** is returned in R0:Rm and the cosine of **angle-in-radians** is returned in (R <m+1> :R <2*m+1>).

In radians, the computation of zSIN(X) and zCOS(X) is based on the following polynomial expansions:

$$\begin{aligned}\sin(X) &= X - X^3/(3!) + X^5/(5!) - X^7/(7!)...\\ &= X + X * P(X^2), \text{ where}\\ &P(y) = y/(3!) + y^2/(5!) + y^3/(7!)...\end{aligned}$$

$$\begin{aligned}\cos(X) &= 1 - X^2/(2!) + x^4/(4!) - X^6/(6!)...\\ &= Q(X^2), \text{ where}\\ &Q(y) = (1 - y/(2!) + y^2/(4!) + y^3/(6!)...)\end{aligned}$$

1 If $|X| < 2^{(-f/2)}$,
then $zSIN(X) = X$ and $zCOS(X) = 1$
(see the section on MTH$zCOSH for the definition of f)

2 If $2^{-f/2} \leq |X| < \pi/4$,
then $zSIN(X) = X + P(X^2)$
and zCOS(X) = $Q(X^2)$

3 If $\pi/4 \leq |X|$ *and* $X > 0$,

a. Let $J = INT(X/(\pi/4))$
and $I = J modulo\ 8$

b. If J is even, let $Y = X - J * (\pi/4)$
otherwise,
let $Y = (J + 1) * (\pi/4) - X$

With the above definitions, the following table relates zSIN(X) and zCOS(X) to zSIN(Y) and zCOS(Y):

Value of *I*	zSIN(X)	zCOS(X)
0	zSIN(Y)	zCOS(Y)
1	zCOS(Y)	zSIN(Y)
2	zCOS(Y)	-zSIN(Y)
3	zSIN(Y)	-zCOS(Y)
4	-zSIN(Y)	-zCOS(Y)
5	-zCOS(Y)	-zSIN(Y)
6	-zCOS(Y)	zSIN(Y)
7	-zSIN(Y)	zCOS(Y)

c. zSIN(Y) and zCOS(Y) are computed as follows:
$zSIN(Y) = Y + P(Y^2)$,
and $zCOS(Y) = Q(Y^2)$

4 If $\pi/4 \leq |X|$ *and* $X < 0$,
then $zSIN(X) = -zSIN(|X|)$
and $zCOS(X) = zCOS(|X|)$

CONDITION VALUE RETURNED

SS$_ROPRAND	Reserved operand. The MTH$xSINCOS procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$xSINCOSD Sine and Cosine of Angle Expressed in Degrees

The Sine and Cosine of Angle Expressed in Degrees routine returns the sine and cosine of a given angle (in degrees).

FORMAT

MTH$SINCOSD *angle-in-degrees ,sine ,cosine*
MTH$DSINCOSD *angle-in-degrees ,sine ,cosine*
MTH$GSINCOSD *angle-in-degrees ,sine ,cosine*
MTH$HSINCOSD *angle-in-degrees ,sine ,cosine*

Each of the above four formats accepts as input one of the four floating-point types.

jsb entries

MTH$SINCOSD_R5
MTH$DSINCOSD_R7
MTH$GSINCOSD_R7
MTH$HSINCOSD_R7

Each of the above four JSB entries accepts as input one of the four floating-point types.

RETURNS

MTH$SINCOSD, MTH$DSINCOSD, MTH$GSINCOSD, and MTH$HSINCOSD return the sine and cosine of the input angle by reference in the **sine** and **cosine** arguments.

ARGUMENTS

angle-in-degrees

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **read only**
mechanism: **by reference**

Angle (in degrees) whose sine and cosine are returned by MTH$xSINCOSD. The **angle-in-degrees** argument is the address of a floating-point number that is this angle. For MTH$SINCOSD, **angle-in-degrees** is an F-floating number. For MTH$DSINCOSD, **angle-in-degrees** is a D-floating number. For MTH$GSINCOSD, **angle-in-degrees** is a G-floating number. For MTH$HSINCOSD, **angle-in-degrees** is an H-floating number.

sine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **write only**
mechanism: **by reference**

Sine of the angle specified by **angle-in-degrees**. The **sine** argument is the address of a floating-point number. MTH$SINCOSD writes an F-floating number into **sine**. MTH$DSINCOSD writes a D-floating number into **sine**. MTH$GSINCOSD writes a G-floating number into **sine**. MTH$HSINCOSD writes an H-floating number into **sine**.

cosine

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating, H_floating**
access: **write only**
mechanism: **by reference**

Cosine of the angle specified by **angle-in-degrees**. The **cosine** argument is the address of a floating-point number. MTH$SINCOSD writes an F-floating number into **cosine**. MTH$DSINCOSD writes a D-floating number into **cosine**. MTH$GSINCOSD writes a G-floating number into **cosine**. MTH$HSINCOSD writes an H-floating number into **cosine**.

DESCRIPTION

All routines with JSB entry points accept a single argument in R0:Rm, where *m*, which is defined below, is dependent on the data type.

Data Type	m
F_floating	0
D_floating	1
G_floating	1
H_floating	3

In general, Run-Time Library routines with JSB entry points return one value in R0:Rm. The MTH$SINCOSD routine returns two values, however. The sine of **angle-in-degrees** is returned in R0:Rm and the cosine of **angle-in-degrees** is returned in (R <m+1> :R <2*m+1>).

In degrees, the computation of zSIND(X) and zCOSD(X) is based on the following polynomial expansions:

$$\begin{aligned} SIND(X) &= (C*X) - (C*X)^3/(3!) + \\ &\quad (C*X)^5/(5!) - (C*X)^7/(7!) \ldots \\ &= X/2^6 + X * P(X^2), \text{ where} \\ &\quad P(y) = -y/(3!) + y^2/(5!) - y^3/(7!) \ldots \end{aligned}$$

$$\begin{aligned} COSD(X) &= 1 - (C*X)^2/(2!) + \\ &\quad (C*X)^4/(4!) - (C*X)^6/(6!) \ldots \\ &= Q(X^2), \text{ where} \\ &\quad Q(y) = 1 - y/(2!) + y^2/(4!) - y^3/(6!) \ldots \\ &\quad \text{and } C = \pi/180 \end{aligned}$$

1 If $|X| < (180/\pi) * 2^{-2^{e-1}}$ and underflow signaling is enabled, underflow is signaled for zSIND(X) and zSINCOSD(X).
See MTH$zCOSH for the definition of e.

OTHERWISE:

2 If $|X| < (180/\pi) * 2^{(-f/2)}$,
then $zSIND(X) = (\pi/180) * X$ and $zCOSD(X) = 1$.
(See MTH$zCOSH for the definition of *f*.)

3 If $(180/\pi) * 2^{(-f/2)} \leq |X| < 45$
then $zSIND(X) = X/2^6 + P(X^2)$
and $zCOSD(X) = Q(X^2)$

4 If $45 \leq |X|$ *and* $X > 0$,

a. Let $J = INT(X/(45))$and
$I = J$ *modulo* 8

b. If J is even, let $Y = X - J * 45$;
otherwise, let $Y = (J + 1) * 45 - X$.
With the above definitions, the following table relates zSIND(X) and zCOSD(X) to zSIND(Y) and zCOSD(Y):

Value of *I*	zSIND(X)	zCOSD(X)
0	zSIND(Y)	zCOSD(Y)
1	zCOSD(Y)	zSIND(Y)
2	zCOSD(Y)	-zSIND(Y)
3	zSIND(Y)	-zCOSD(Y)
4	-zSIND(Y)	-zCOSD(Y)
5	-zCOSD(Y)	-zSIND(Y)
6	-zCOSD(Y)	zSIND(Y)
7	-zSIND(Y)	zCOSD(Y)

c. zSIND(Y) and zCOSD(Y) are computed as follows:
$zSIND(Y) = Y/2^6 + P(Y^2)$
$zCOSD(Y) = Q(Y^2)$

d. If $45 \leq |X|$ *and* $X < 0$,
then $zSIND(X) = -zSIND(|X|)$
and $zCOSD(X) = zCOSD(|X|)$

CONDITION VALUES SIGNALED

SS$_ROPRAND — Reserved operand. The MTH$xSINCOSD procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_FLOUNDMAT — Floating-point underflow in math library. The absolute value of the input angle is less than $180/\pi * 2^{-m}$ (where m = 128 for F-floating and D-floating, 1,024 for G-floating, and 16,384 for H-floating).

MTH$xSIND Sine of Angle Expressed in Degrees

The Sine of Angle Expressed in Degrees routine returns the sine of a given angle (in degrees).

FORMAT

MTH$SIND *angle-in-degrees*
MTH$DSIND *angle-in-degrees*
MTH$GSIND *angle-in-degrees*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$SIND_R4
MTH$DSIND_R7
MTH$GSIND_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The sine of the angle. MTH$SIND returns an F-floating number. MTH$DSIND returns a D-floating number. MTH$GSIND returns a G-floating number.

ARGUMENTS

angle-in-degrees
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number that is this angle. For MTH$SIND, **angle-in-degrees** specifies an F-floating number. For MTH$DSIND, **angle-in-degrees** specifies a D-floating number. For MTH$GSIND, **angle-in-degrees** specifies a G-floating number.

DESCRIPTION

See MTH$SINCOSD for the algorithm that is used to compute the sine.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HSIND.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$SIND procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased ecponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOUNDMAT	Floating-point underflow in math library. The absolute value of the input angle is less than $180/\pi * 2^{-m}$ (where m = 128 for F-floating and D-floating, and 1,024 for G-floating).

MTH$xSINH Hyperbolic Sine

The Hyperbolic Sine routine returns the hyperbolic sine of the input value specified by **floating-point-input-value**.

FORMAT

MTH$SINH *floating-point-input-value*
MTH$DSINH *floating-point-input-value*
MTH$GSINH *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The hyperbolic sine of **floating-point-input-value**. MTH$SINH returns an F-floating number. MTH$DSINH returns a D-floating number. MTH$GSINH returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that is this value. For MTH$SINH, **floating-point-input-value** specifies an F-floating number. For MTH$DSINH, **floating-point-input-value** specifies a D-floating number. For MTH$GSINH, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

Computation of the hyperbolic sine function depends on the magnitude of the input argument. The range of the function is partitioned using four data type dependent constants: a(z), b(z), and c(z). The subscript *z* indicates the data type. The constants depend on the number of exponent bits (*e*) and the number of fraction bits (*f*) associated with the data type (z).

The values of *e* and *f* are:

z	e	f
F	8	24
D	8	56
G	11	53

The values of the constants in terms of e and f are:

Variable	Value
a(z)	$2^{(-f/2)}$
b(z)	CEILING[$(f+1)/2 * \ln(2)$]
c(z)	$(2^{(e-1)} * \ln(2))$

Based on the above definitions, zSINH(X) is computed as follows:

Value of X	Value Returned
$\|X\| < a(z)$	X
$a(z) \leq \|X\| < 1.0$	zSINH(X) is computed using a power series expansion in $\|X\|^2$
$1.0 \leq \|X\| < b(z)$	$(zEXP(X) - zEXP(-X))/2$
$b(z) \leq \|X\| < c(z)$	$SIGN(X) * zEXP(\|X\|)/2$
$c(z) \leq \|X\|$	Overflow occurs

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HSINH.

CONDITION VALUES SIGNALED

SS$_ROPRAND

Reserved operand. The MTH$xSINH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$_FLOOVEMAT

Floating-point overflow in Math Library: the absolute value of **floating-point-input-value** is greater than *yyy*. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

The values of *yyy* are approximately:

MTH$SINH—88.722
MTH$DSINH—88.722
MTH$GSINH—709.782

MTH$xSQRT Square Root

The Square Root routine returns the square root of the input value **floating-point-input-value**.

FORMAT

MTH$SQRT *floating-point-input-value*
MTH$DSQRT *floating-point-input-value*
MTH$GSQRT *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$SQRT_R3
MTH$DSQRT_R5
MTH$GSQRT_R5

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The square root of **floating-point-input-value**. MTH$SQRT returns an F-floating number. MTH$DSQRT returns a D-floating number. MTH$GSQRT returns a G-floating number.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

Input value. The **floating-point-input-value** argument is the address of a floating-point number that contains this input value. For MTH$SQRT, **floating-point-input-value** specifies an F-floating number. For MTH$DSQRT, **floating-point-input-value** specifies a D-floating number. For MTH$GSQRT, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

The square root of X is computed as follows:

If $X < 0$, an error is signaled.

Let $X = 2^K * F$

where:

K is the exponential part of the floating-point data

F is the fractional part of the floating-point data

If K is even:

$X = 2^{(2*P)} * F,$
$zSQRT(X) = 2^P * zSQRT(F),$
$1/2 \leq F < 1$, where P = K/2

If K is odd:

$X = 2^{(2*P+1)} * F = 2^{(2*P+2)} * (F/2),$
$zSQRT(X) = 2^{(P+1)} * zSQRT(F/2),$
$1/4 \leq F/2 < 1/2$, where p = (K-1)/2

Let $F' = A * F + B$, when K is even:

A = 0.95F6198 (hex)

B = 0.6BA5918 (hex)

Let $F' = A * (F/2) + B$, when K is odd:

A = 0.D413CCC (hex)

B = 0.4C1E248 (hex)

Let K' = P, when K is even

Let K' = P+1, when K is odd

Let $Y[0] = 2^{K'} * F'$ be a straight line approximation within the given interval using coefficients A and B which minimize the absolute error at the midpoint and endpoint.

Starting with Y[0], n Newton-Raphson iterations are performed:

$$Y[n + 1] = 1/2 * (Y[n] + X/Y[n])$$

where n = 2, 3, or 3 for z = F-floating, D-floating, or G-floating, respectively.

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HSQRT.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xSQRT procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_SQUROONEG	Square root of negative number. Argument **floating-point-input-value** is less than 0.0. LIB$SIGNAL copies the floating-point reserved operand to the mechanism argument vector CHF$L_MCH_SAVR0/R1. The result is the floating-point reserved operand unless you have written a condition handler to change CHF$L_MCH_SAVR0/R1.

MTH$xTAN Tangent of Angle Expressed in Radians

The Tangent of Angle Expressed in Radians routine returns the tangent of a given angle (in radians).

FORMAT

MTH$TAN *angle-in-radians*
MTH$DTAN *angle-in-radians*
MTH$GTAN *angle-in-radians*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$TAN_R4
MTH$DTAN_R7
MTH$GTAN_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The tangent of the angle specified by **angle-in-radians**. MTH$TAN returns an F-floating number. MTH$DTAN returns a D-floating number. MTH$GTAN returns a G-floating number.

ARGUMENTS

angle-in-radians
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input angle (in radians). The **angle-in-radians** argument is the address of a floating-point number that is this angle. For MTH$TAN, **angle-in-radians** specifies an F-floating number. For MTH$DTAN, **angle-in-radians** specifies a D-floating number. For MTH$GTAN, **angle-in-radians** specifies a G-floating number.

DESCRIPTION

When the input argument is expressed in radians, the tangent function is computed as follows:

1. If $|X| < 2^{(-f/2)}$, then $zTAN(X) = X$ (see the section on MTH$zCOSH for the definition of *f*)
2. Otherwise, call MTH$zSINCOS to obtain zSIN(X) and zCOS(X); then
   a. If $zCOS(X) = 0$, signal overflow
   b. Otherwise, $zTAN(X) = zSIN(X)/zCOS(X)$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HTAN.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xTAN procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library.

MTH$xTAND Tangent of Angle Expressed in Degrees

The Tangent of Angle Expressed in Degrees routine returns the tangent of a given angle (in degrees).

FORMAT

MTH$TAND *angle-in-degrees*
MTH$DTAND *angle-in-degrees*
MTH$GTAND *angle-in-degrees*

Each of the above formats accepts as input one of the floating-point types.

jsb entries

MTH$TAND_R4
MTH$DTAND_R7
MTH$GTAND_R7

Each of the above JSB entries accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

Tangent of the angle specified by **angle-in-degrees**. MTH$TAND returns an F-floating number. MTH$DTAND returns a D-floating number. MTH$GTAND returns a G-floating number.

ARGUMENTS

angle-in-degrees
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input angle (in degrees). The **angle-in-degrees** argument is the address of a floating-point number which is this angle. For MTH$TAND, **angle-in-degrees** specifies an F-floating number. For MTH$DTAND, **angle-in-degrees** specifies a D-floating number. For MTH$GTAND, **angle-in-degrees** specifies a G-floating number.

DESCRIPTION

When the input argument is expressed in degrees, the tangent function is computed as follows:

1. If $|X| < (180/\pi)*2^{(-2/(e-1))}$ and underflow signaling is enabled, underflow is signaled (see the section on MTH$zCOSH for the definition of e).
2. Otherwise, if $|X| < (180/\pi) * 2^{(-f/2)}$, then $zTAND(X) = (\pi/180) * X$. See the description of MTH$zCOSH for the definition of f.
3. Otherwise, call MTH$zSINCOSD to obtain zSIND(X) and zCOSD(X).
   a. Then, if $zCOSD(X) = 0$, signal overflow
   b. Else, $zTAND(X) = zSIND(X)/zCOSD(X)$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HTAND.

CONDITION VALUES SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xTAND procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.
MTH$_FLOOVEMAT	Floating-point overflow in Math Library.

MTH$xTANH Compute the Hyperbolic Tangent

The Compute the Hyperbolic Tangent routine returns the hyperbolic tangent of the input value.

FORMAT

MTH$TANH *floating-point-input-value*
MTH$DTANH *floating-point-input-value*
MTH$GTANH *floating-point-input-value*

Each of the above formats accepts as input one of the floating-point types.

RETURNS

VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **write only**
mechanism: **by value**

The hyperbolic tangent of **floating-point-input-value**. MTH$TANH returns an F-floating number. MTH$DTANH returns a D-floating number. MTH$GTANH returns a G-floating number. Unlike the other three routines, MTH$HTANH returns the hyperbolic tangent by reference in the **h-tanh** argument.

ARGUMENTS

floating-point-input-value
VMS usage: **floating_point**
type: **F_floating, D_floating, G_floating**
access: **read only**
mechanism: **by reference**

The input value. The **floating-point-input-value** argument is the address of a floating-point number that contains this input value. For MTH$TANH, **floating-point-input-value** specifies an F-floating number. For MTH$DTANH, **floating-point-input-value** specifies a D-floating number. For MTH$GTANH, **floating-point-input-value** specifies a G-floating number.

DESCRIPTION

In calculating the hyperbolic tangent of x, the values of g and h are:

z	g	h
F	12	10
D	28	21
G	26	20

For MTH$TANH, MTH$DTANH, and MTH$GTANH the hyperbolic tangent of x is then computed as follows:

Value of x	Hyperbolic Tangent Returned
$\|x\| \leq 2^{-g}$	X
$2^{-g} < \|X\| < 0.5$	$xTANH(X) = X + X^3 * R(X^2)$, where $R(X^2)$ is a rational function of X^2.
$0.5 \leq \|X\| < 1.0$	$xTANH(X) = xTANH(xHI) + xTANH(xLO) * C/B$ where $C = 1 - xTANH(xHI) * xTANH(xHI)$, $B = 1 + xTANH(xHI) * xTANH(xLO)$, $xHI = 1/2 + N/16 + 1/32$ for N=0,1,...,7, and $xLO = X - xHI$.
$1.0 < \|X\| < h$	$xTANH(X) = (xEXP(2*X) - 1)/(xEXP(2*X) + 1)$
$h \leq \|X\|$	$xTANH(X) = sign(X) * 1$

The routine description for the H-floating point version of this routine is listed alphabetically under MTH$HTANH.

CONDITION VALUE SIGNALED

SS$_ROPRAND	Reserved operand. The MTH$xTANH procedure encountered a floating-point reserved operand due to incorrect user input. A floating-point reserved operand is a floating-point datum with a sign bit of 1 and a biased exponent of zero. Floating-point reserved operands are reserved for future use by DIGITAL.

MTH$UMAX Compute Unsigned Maximum

The Compute Unsigned Maximum routine computes the unsigned longword maximum of n unsigned longword arguments, where n is greater than or equal to 1.

FORMAT

MTH$UMAX *argument [argument,...]*

RETURNS

VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

Maximum value returned by MTH$UMAX.

ARGUMENTS

argument

VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Argument whose maximum MTH$UMAX computes. Each **argument** argument is an unsigned longword that contains one of these values.

argument

VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Additional arguments whose maximum MTH$UMAX computes. Each **argument** argument is an unsigned longword that contains one of these values.

DESCRIPTION

MTH$UMAX is the unsigned version of MTH$JMAX0.

CONDITION VALUES RETURNED

None.

MTH$UMIN Compute Unsigned Minimum

The Compute Unsigned Minimum routine computes the unsigned longword minimum of n unsigned longword arguments, where n is greater than or equal to 1.

FORMAT

MTH$UMIN *argument [argument,...]*

RETURNS

VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

Minimum value returned by MTH$UMIN.

ARGUMENTS

argument
VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Argument whose minimum MTH$UMIN computes. Each **argument** argument is an unsigned longword that contains one of these values.

argument
VMS usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Additional arguments whose minimum MTH$UMIN computes. Each **argument** argument is an unsigned longword that contains one of these values.

DESCRIPTION

MTH$UMIN is the unsigned version of MTH$JMIN0.

CONDITION VALUES RETURNED

None.

A Undocumented MTH$ Routines

This appendix lists all of the entry point and argument information for the MTH$ routines not documented in the MTH$ Reference Section of this manual.

Table A–1 Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$ABS		*F-floating Absolute Value Routine*
	Format:	**MTH$ABS** f-floating
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$DABS		*D-floating Absolute Value Routine*
	Format:	**MTH$DABS** d-floating
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GABS		*G-floating Absolute Value Routine*
	Format:	**MTH$GABS** g-floating
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HABS		*H-floating Absolute Value Routine*
	Format:	**MTH$ABS** h-abs-val, h-floating
	Returns:	None
	h-abs-val:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference
MTH$IIABS		*Word Absolute Value Routine*
	Format:	**MTH$IIABS** word
	Returns:	word_signed, word (signed), write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$JIABS		*Longword Absolute Value Routine*
	Format:	**MTH$JIABS** longword
	Returns:	longword_signed, longword (signed), write only, by value
	longword:	longword_signed, longword (signed), read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$IIAND		*Bitwise AND of Two Word Parameters Routine*
	Format:	**MTH$IIAND** word1, word2
	Returns:	word_unsigned, word (unsigned), write only, by value
	word1:	word_unsigned, word (unsigned), read only, by reference
	word2:	word_unsigned, word (unsigned), read only, by reference
MTH$JIAND		*Bitwise AND of Two Longword Parameters Routine*
	Format:	**MTH$JIAND** longword1, longword2
	Returns:	longword_unsigned, longword (unsigned), write only, by value
	longword1:	longword_unsigned, longword (unsigned), read only, by reference
	longword2:	longword_unsigned, longword (unsigned), read only, by reference
MTH$DBLE		*Convert F-floating to D-floating (Exact) Routine*
	Format:	**MTH$DBLE** f-floating
	Returns:	floating_point, D_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$GDBLE		*Convert F-floating to G-floating (Exact) Routine*
	Format:	**MTH$GDBLE** f-floating
	Returns:	floating_point, G_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$DIM		*Positive Difference of Two F-floating Parameters Routine*
	Format:	**MTH$DIM** f-floating1, f-floating2
	Returns:	floating_point, F_floating, write only, by value
	f-floating1:	floating_point, F_floating, read only, by reference
	f-floating2:	floating_point, F_floating, read only, by reference
MTH$DDIM		*Positive Difference of Two D-floating Parameters Routine*
	Format:	**MTH$DDIM** d-floating1, d-floating2
	Returns:	floating_point, D_floating, write only, by value
	d-floating1:	floating_point, D_floating, read only, by reference
	d-floating2:	floating_point, D_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$GDIM		*Positive Difference of Two G-floating Parameters Routine*
	Format:	**MTH$GDIM** g-floating1, g-floating2
	Returns:	floating_point, G_floating, write only, by value
	g-floating1:	floating_point, G_floating, read only, by reference
	g-floating2:	floating_point, G_floating, read only, by reference
MTH$HDIM		*Positive Difference of Two H-floating Parameters Routine*
	Format:	**MTH$HDIM** h-floating, h-floating1, h-floating2
	Returns:	None
	h-floating:	floating_point, H_floating, write only, by reference
	h-floating1:	floating_point, H_floating, read only, by reference
	h-floating2:	floating_point, H_floating, read only, by reference
MTH$IIDIM		*Positive Difference of Two Word Parameters Routine*
	Format:	**MTH$IIDIM** word1, word2
	Returns:	word_signed, word (signed), write only, by value
	word1:	word_signed, word (signed), read only, by reference
	word2:	word_signed, word (signed), read only, by reference
MTH$JIDIM		*Positive Difference of Two Longword Parameters Routine*
	Format:	**MTH$JIDIM** longword1, longword2
	Returns:	longword_signed, longword (signed), write only, by value
	longword1:	longword_signed, longword (signed), read only, by reference
	longword2:	longword_signed, longword (signed), read only, by reference
MTH$IIEOR		*Bitwise Exclusive OR of Two Word Parameters Routine*
	Format:	**MTH$IIEOR** word1, word2
	Returns:	word_unsigned, word (unsigned), write only, by value
	word1:	word_unsigned, word (unsigned), read only, by reference
	word2:	word_unsigned, word (unsigned), read only, by reference
MTH$JIEOR		*Bitwise Exclusive OR of Two Longword Parameters Routine*
	Format:	**MTH$JIEOR** longword1, longword2
	Returns:	longword_unsigned, longword (unsigned), write only, by value
	longword1:	longword_unsigned, longword (unsigned), read only, by reference
	longword2:	longword_unsigned, longword (unsigned), read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$IIFIX		*Convert F-floating to Word (Truncated) Routine*
	Format:	**MTH$IIFIX** f-floating
	Returns:	word_signed, word (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$JIFIX		*Convert F-floating to Longword (Truncated) Routine*
	Format:	**MTH$JIFIX** f-floating
	Returns:	longword_signed, longword (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$FLOATI		*Convert Word to F-floating (Exact) Routine*
	Format:	**MTH$FLOATI** word
	Returns:	floating_point, F_floating, write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$DFLOTI		*Convert Word to D-floating (Exact) Routine*
	Format:	**MTH$DFLOTI** word
	Returns:	floating_point, D_floating, write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$GFLOTI		*Convert Word to G-floating (Exact) Routine*
	Format:	**MTH$GFLOTI** word
	Returns:	floating_point, G_floating, write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$FLOATJ		*Convert Longword to F-floating (Exact) Routine*
	Format:	**MTH$FLOATJ** longword
	Returns:	floating_point, F_floating, write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$DFLOTJ		*Convert Longword to D-floating (Exact) Routine*
	Format:	**MTH$DFLOTJ** longword
	Returns:	floating_point, D_floating, write only, by value
	longword:	longword_signed, longword (signed), read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$GFLOTJ		*Convert Longword to G-floating (Exact) Routine*
	Format:	**MTH$GFLOTJ** longword
	Returns:	floating_point, G_floating, write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$FLOOR		*Convert F-floating to Greatest F-floating Integer Routine*
	Format:	**MTH$FLOOR** f-floating
	JSB:	**MTH$FLOOR_R1** f-floating
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$DFLOOR		*Convert D-floating to Greatest D-floating Integer Routine*
	Format:	**MTH$DFLOOR** d-floating
	JSB:	**MTH$DFLOOR_R3** d-floating
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GFLOOR		*Convert G-floating to Greatest G-floating Integer Routine*
	Format:	**MTH$GFLOOR** g-floating
	JSB:	**MTH$GFLOOR_R3** g-floating
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HFLOOR		*Convert H-floating to Greatest H-floating Integer Routine*
	Format:	**MTH$HFLOOR** max-h-float, h-floating
	JSB:	**MTH$HFLOOR_R7** h-floating
	Returns:	None
	max-h-float:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference
MTH$AINT		*Convert F-floating to Truncated F-floating Routine*
	Format:	**MTH$AINT** f-floating
	JSB:	**MTH$AINT_R2** f-floating
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$DINT		*Convert D-floating to Truncated D-floating Routine*
	Format:	**MTH$DINT** d-floating
	JSB:	**MTH$DINT_R4** d-floating
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$IIDINT		*Convert D-floating to Word (Truncated) Routine*
	Format:	**MTH$IIDINT** d-floating
	Returns:	word_signed, word (signed), write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$JIDINT		*Convert D-floating to Longword (Truncated) Routine*
	Format:	**MTH$JIDINT** d-floating
	Returns:	longword_signed, longword (signed), write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GINT		*Convert G-floating to G-floating (Truncated) Routine*
	Format:	**MTH$GINT** g-floating
	JSB:	**MTH$GINT_R4** g-floating
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$IIGINT		*Convert G-floating to Word (Truncated) Routine*
	Format:	**MTH$IIGINT** g-floating
	Returns:	word_signed, word (signed), write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$JIGINT		*Convert G-floating to Longword (Truncated) Routine*
	Format:	**MTH$JIGINT** g-floating
	Returns:	longword_signed, longword (signed), write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HINT		*Convert H-floating to H-floating (Truncated) Routine*
	Format:	**MTH$HINT** trunc-h-flt, h-floating
	JSB:	**MTH$HINT_R8** h-floating
	Returns:	None
	trunc-h-flt:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$IIHINT		*Convert H-floating to Truncated Word Routine*
	Format:	**MTH$IIHINT** h-floating
	Returns:	word_signed, word (signed), write only, by value
	h-floating:	floating_point, H_floating, read only, by reference
MTH$JIHINT		*Convert H-floating to Truncated Longword Routine*
	Format:	**MTH$JIHINT** h-floating
	Returns:	longword_signed, longword (signed), write only, by value
	h-floating:	floating_point, H_floating, read only, by reference
MTH$IINT		*Convert F-floating to Word (Truncated) Routine*
	Format:	**MTH$IINT** f-floating
	Returns:	word_signed, word (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$JINT		*Convert F-floating to Longword (Truncated) Routine*
	Format:	**MTH$JINT** f-floating
	Returns:	longword_signed, longword (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$IIOR		*Bitwise Inclusive OR of Two Word Parameters Routine*
	Format:	**MTH$IIOR** word1, word2
	Returns:	word_unsigned, word (unsigned), write only, by value
	word1:	word_unsigned, word (unsigned), read only, by reference
	word2:	word_unsigned, word (unsigned), read only, by reference
MTH$JIOR		*Bitwise Inclusive OR of Two Longword Parameters Routine*
	Format:	**MTH$JIOR** longword1, longword2
	Returns:	longword_unsigned, longword (unsigned), write only, by value
	longword1:	longword_unsigned, longword (unsigned), read only, by reference
	longword2:	longword_unsigned, longword (unsigned), read only, by reference
MTH$AIMAX0		*F-floating Maximum of N Word Parameters Routine*
	Format:	**MTH$AIMAX0** word, . . .
	Returns:	floating_point, F_floating, write only, by value
	word:	word_signed, word (signed), read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$AJMAX0		*F-floating Maximum of N Longword Parameters Routine*
	Format:	**MTH$AJMAX0** longword, . . .
	Returns:	floating_point, F_floating, write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$IMAX0		*Word Maximum of N Word Parameters Routine*
	Format:	**MTH$IMAX0** word, . . .
	Returns:	word_signed, word (signed), write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$JMAX0		*Longword Maximum of N Longword Parameters Routine*
	Format:	**MTH$JMAX0** longword, . . .
	Returns:	longword_signed, longword (signed), write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$AMAX1		*F-floating Maximum of N F-floating Parameters Routine*
	Format:	**MTH$AMAX1** f-floating, . . .
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$DMAX1		*D-floating Maximum of N D-floating Parameters Routine*
	Format:	**MTH$DMAX1** d-floating, . . .
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GMAX1		*G-floating Maximum of N G-floating Parameters Routine*
	Format:	**MTH$GMAX1** g-floating, . . .
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HMAX1		*H-floating Maximum of N H-floating Parameters Routine*
	Format:	**MTH$HMAX1** h-float-max, h-floating, . . .
	Returns:	None
	h-float-max:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$IMAX1		*Word Maximum of N F-floating Parameters Routine*
	Format:	**MTH$IMAX1** f-floating, . . .
	Returns:	word_signed, word (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$JMAX1		*Longword Maximum of N F-floating Parameters Routine*
	Format:	**MTH$JMAX1** f-floating, . . .
	Returns:	longword_signed, longword (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$AIMIN0		*F-floating Minimum of N Word Parameters Routine*
	Format:	**MTH$AIMIN0** word, . . .
	Returns:	floating_point, F_floating, write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$AJMIN0		*F-floating Minimum of N Longword Parameters Routine*
	Format:	**MTH$AJMIN0** longword, . . .
	Returns:	floating_point, F_floating, write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$IMIN0		*Word Minimum of N Word Parameters Routine*
	Format:	**MTH$IMIN0** word, . . .
	Returns:	word_signed, word (signed), write only, by value
	word:	word_signed, word (signed), read only, by reference
MTH$JMIN0		*Longword Minimum of N Longword Parameters Routine*
	Format:	**MTH$JMIN0** longword, . . .
	Returns:	longword_signed, longword (signed), write only, by value
	longword:	longword_signed, longword (signed), read only, by reference
MTH$AMIN1		*F-floating Minimum of N F-floating Parameters Routine*
	Format:	**MTH$AMIN1** f-floating, . . .
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$DMIN1		*D-floating Minimum of N D-floating Parameters Routine*
	Format:	**MTH$DMIN1** d-floating, . . .
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GMIN1		*G-floating Minimum of N G-floating Parameters Routine*
	Format:	**MTH$GMIN1** g-floating, . . .
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HMIN1		*H-floating Minimum of N H-floating Parameters Routine*
	Format:	**MTH$HMIN1** h-float-max, h-floating, . . .
	Returns:	None
	h-float-max:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference
MTH$IMIN1		*Word Minimum of N F-floating Parameters Routine*
	Format:	**MTH$IMIN1** f-floating, . . .
	Returns:	word_signed, word (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$JMIN1		*Longword Minimum of N F-floating Parameters Routine*
	Format:	**MTH$JMIN1** f-floating, . . .
	Returns:	longword_signed, longword (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$AMOD		*Remainder of Two F-floating Parameters Routine*
	Format:	**MTH$AMOD** dividend, divisor
	Returns:	floating_point, F_floating, write only, by value
	dividend:	floating_point, F_floating, read only, by reference
	divisor:	floating_point, F_floating, read only, by reference
MTH$DMOD		*Remainder of Two D-floating Parameters Routine*
	Format:	**MTH$DMOD** dividend, divisor
	Returns:	floating_point, D_floating, write only, by value
	dividend:	floating_point, D_floating, read only, by reference
	divisor:	floating_point, D_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$GMOD		*Remainder of Two G-floating Parameters Routine*
	Format:	**MTH$GMOD** dividend, divisor
	Returns:	floating_point, G_floating, write only, by value
	dividend:	floating_point, G_floating, read only, by reference
	divisor:	floating_point, G_floating, read only, by reference
MTH$HMOD		*Remainder of Two H-floating Parameters Routine*
	Format:	**MTH$HMOD** h-mod, dividend, divisor
	Returns:	None
	h-mod:	floating_point, H_floating, write only, by reference
	dividend:	floating_point, H_floating, read only, by reference
	divisor:	floating_point, H_floating, read only, by reference
MTH$IMOD		*Remainder of Two Word Parameters Routine*
	Format:	**MTH$IMOD** dividend, divisor
	Returns:	word_signed, word (signed), write only, by value
	dividend:	word_signed, word (signed), read only, by reference
	divisor:	word_signed, word (signed), read only, by reference
MTH$JMOD		*Remainder of Two Longword Parameters Routine*
	Format:	**MTH$JMOD** dividend, divisor
	Returns:	longword_signed, longword (signed), write only, by value
	dividend:	longword_signed, longword (signed), read only, by reference
	divisor:	longword_signed, longword (signed), read only, by reference
MTH$ANINT		*Convert F-floating to Nearest F-floating Integer Routine*
	Format:	**MTH$ANINT** f-floating
	Returns:	floating_point, F_floating, write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$DNINT		*Convert D-floating to Nearest D-floating Integer Routine*
	Format:	**MTH$DNINT** d-floating
	Returns:	floating_point, D_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$IIDNNT		*Convert D-floating to Word Integer Routine*
	Format:	**MTH$IIDNNT** d-floating
	Returns:	word_signed, word (signed), write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$JIDNNT		*Convert D-floating to Nearest Longword Integer Routine*
	Format:	**MTH$JIDNNT** d-floating
	Returns:	longword_signed, longword (signed), write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$GNINT		*Convert G-floating to Nearest G-floating Integer Routine*
	Format:	**MTH$GNINT** g-floating
	Returns:	floating_point, G_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$IIGNNT		*Convert G-floating to Nearest Word Integer Routine*
	Format:	**MTH$IIGNNT** g-floating
	Returns:	word_signed, word (signed), write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$JIGNNT		*Convert G-floating to Nearest Longword Integer Routine*
	Format:	**MTH$JIGNNT** g-floating
	Returns:	longword_signed, longword (signed), write only, by value
	g-floating:	floating_point, G_floating, read only, by reference
MTH$HNINT		*Convert H-floating to Nearest H-floating Integer Routine*
	Format:	**MTH$HNINT** nearst-h-flt, h-floating
	Returns:	None
	nearst-h-flt:	floating_point, H_floating, write only, by reference
	h-floating:	floating_point, H_floating, read only, by reference
MTH$IIHNNT		*Convert H-floating to Nearest Word Integer Routine*
	Format:	**MTH$IIHNNT** h-floating
	Returns:	word_signed, word (signed), write only, by value
	h-floating:	floating_point, H_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$JIHNNT		*Convert H-floating to Nearest Longword Integer Routine*
	Format:	**MTH$JIHNNT** h-floating
	Returns:	longword_signed, longword (signed), write only, by value
	h-floating:	floating_point, H_floating, read only, by reference
MTH$ININT		*Convert F-floating to Nearest Word Integer Routine*
	Format:	**MTH$ININT** f-floating
	Returns:	word_signed, word (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$JNINT		*Convert F-floating to Nearest Longword Integer Routine*
	Format:	**MTH$JNINT** f-floating
	Returns:	longword_signed, longword (signed), write only, by value
	f-floating:	floating_point, F_floating, read only, by reference
MTH$INOT		*Bitwise Complement of Word Parameter Routine*
	Format:	**MTH$INOT** word
	Returns:	word_unsigned, word (unsigned), write only, by value
	word:	word_unsigned, word (unsigned), read only, by reference
MTH$JNOT		*Bitwise Complement of Longword Parameter Routine*
	Format:	**MTH$JNOT** longword
	Returns:	longword_unsigned, longword (unsigned), write only, by value
	longword:	longword_unsigned, longword (unsigned), read only, by reference
MTH$DPROD		*D-floating Product of Two F-floating Parameters Routine*
	Format:	**MTH$DPROD** f-floating1, f-floating2
	Returns:	floating_point, D_floating, write only, by value
	f-floating1:	floating_point, F_floating, read only, by reference
	f-floating2:	floating_point, F_floating, read only, by reference
MTH$GPROD		*G-floating Product of Two F-floating Parameters Routine*
	Format:	**MTH$GPROD** f-floating1, f-floating2
	Returns:	floating_point, G_floating, write only, by value
	f-floating1:	floating_point, F_floating, read only, by reference
	f-floating2:	floating_point, F_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$SGN		*F-floating Sign Function*
	Format:	**MTH$SGN** f-floating
	Returns:	longword_signed, longword (signed), write only, by reference
	f-floating:	floating_point, F_floating, read only, by reference
MTH$SGN		*D-floating Sign Function*
	Format:	**MTH$SGN** d-floating
	Returns:	longword_signed, longword (signed), write only, by reference
	d-floating:	floating_point, D_floating, read only, by reference
MTH$IISHFT		*Bitwise Shift of Word Routine*
	Format:	**MTH$IISHFT** word, shift-cnt
	Returns:	word_unsigned, word (unsigned), write only, by value
	word:	word_unsigned, word (unsigned), read only, by reference
	shift-cnt:	word_signed, word (signed), read only, by reference
MTH$JISHFT		*Bitwise Shift of Longword Routine*
	Format:	**MTH$JISHFT** longword, shift-cnt
	Returns:	longword_unsigned, longword (unsigned), write only, by value
	longword:	longword_unsigned, longword (unsigned), read only, by reference
	shift-cnt:	longword_signed, longword (signed), read only, by reference
MTH$SIGN		*F-floating Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$SIGN** f-float-x, f-float-y
	Returns:	floating_point, F_floating, write only, by value
	f-float-x:	floating_point, F_floating, read only, by reference
	f-float-y:	floating_point, F_floating, read only, by reference
MTH$DSIGN		*D-floating Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$DSIGN** d-float-x, d-float-y
	Returns:	floating_point, D_floating, write only, by value
	d-float-x:	floating_point, D_floating, read only, by reference
	d-float-y:	floating_point, D_floating, read only, by reference

Table A–1 (Cont.) Undocumented MTH$ Routines

Routine Name		Entry Point Information
MTH$GSIGN		*G-floating Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$GSIGN** g-float-x, g-float-y
	Returns:	floating_point, G_floating, write only, by value
	g-float-x:	floating_point, G_floating, read only, by reference
	g-float-y:	floating_point, G_floating, read only, by reference
MTH$HSIGN		*H-floating Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$HSIGN** h-result, h-float-x, h-float-y
	Returns:	None
	h-result:	floating_point, H_floating, write only, by reference
	h-float-x:	floating_point, H_floating, read only, by reference
	h-float-y:	floating_point, H_floating, read only, by reference
MTH$IISIGN		*Word Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$IISIGN** word-x, word-y
	Returns:	word_signed, word (signed), write only, by value
	word-x:	word_signed, word (signed), read only, by reference
	word-y:	word_signed, word (signed), read only, by reference
MTH$JISIGN		*Longword Transfer of Sign of Y to Sign of X Routine*
	Format:	**MTH$JISIGN** longwrd-x, longwrd-y
	Returns:	longword_signed, longword (signed), write only, by reference
	longwrd-x:	longword_signed, longword (signed), read only, by reference
	longwrd-y:	longword_signed, longword (signed), read only, by reference
MTH$SNGL		*Convert D-floating to F-floating (Rounded) Routine*
	Format:	**MTH$SNGL** d-floating
	Returns:	floating_point, F_floating, write only, by value
	d-floating:	floating_point, D_floating, read only, by reference
MTH$SNGLG		*Convert G-floating to F-floating (Rounded) Routine*
	Format:	**MTH$SNGLG** g-floating
	Returns:	floating_point, F_floating, write only, by value
	g-floating:	floating_point, G_floating, read only, by reference

Index

R

S

T

Reader's Comments

VMS RTL Mathematics (MTH$) Manual
AA–LA72A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page Description

______ ______________________________

______ ______________________________

______ ______________________________

______ ______________________________

______ ______________________________

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

-- Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here

Cut Along Dotted Line