

GenRef
v1.01

MDOS Reference guide.

Math Library

(C) Copyright 2004
Beery W. Miller
ALL RIGHTS RESERVED

Math - CONTENTS

MATH OVERVIEW	4
MATH DETAILS – RADIX 100	5
CALLING MATH FUNCTIONS.....	6
FLOATING POINT COMPARE	7
FLOATING POINT SUBTRACT	8
FLOATING POINT ADD	9
FLOATING POINT MULTIPLY	10
FLOATING POINT DIVIDE.....	11
FLOATING POINT POWER	12
FLOATING POINT E^x	13
FLOATING POINT LN(x)	14
FLOATING POINT SQR(x)	15
FLOATING POINT COS(x)	16
FLOATING POINT SIN(x).....	17
FLOATING POINT TAN(x).....	18
FLOATING POINT ATN(x).....	19
FLOATING POINT GREATEST INGETER.....	20
CONVERT FLOATING POINT TO INTEGER	21
CONVERT INTEGER TO FLOATING POINT	22
CONVERT STRING TO INTEGER.....	23
CONVERT STRING TO FLOATING POINT	24
CONVERT FLOATING POINT TO STRING	25

ACKNOWLEDGEMENTS

The initial release of this document was prepared by Beery Miller in 2006. After the initial release documenting the basic notes for using the Math XOP, Lee Stewart requested and received permission to integrate the source code into TurboForth and fbForth. Lee fully documented the use of Radix 100 and its implementation with his notes added to this updated document.

MATH - OVERVIEW

All math management routines in MDOS are provided to aid a programmer in writing applications requiring math operations beyond the immediate instruction set of the TMS 9995 microprocessor. The following math operations are supported within the operating system:

• Floating Point Compare	FCOMP
• Floating Point Subtract	FSUB
• Floating Point Add	FADD
• Floating Point Multiply	FMULT
• Floating Point Divide	FDIV
• Floating Point Power	PWR
• Floating Point e^x	EXP
• Floating Point $\ln(x)$	LOG
• Floating Point \sqrt{x}	SQR
• Floating Point $\cos(x)$	COS
• Floating Point $\sin(x)$	SIN
• Floating Point $\tan(x)$	TAN
• Floating Point $\arctan(x)$	ATN
• Floating Point Greatest Integer	GRI
• Convert Floating Point to Integer	CFI
• Convert Integer to Floating Point	CIF
• Convert String to Integer	CSINT
• Convert String to Floating Point	CSN
• Convert Floating Point to String	CNS

MATH Details – Radix 100

Floating-point math routines use radix-100 format for floating-point numbers. The term “radix” is used in mathematics to mean “number base”. We will use “radix 100” to describe the base-100 or centimal number system and “radix 10” to describe the base-10 or decimal number system. Radix-100 format is the same format used by the XML and GPL routines in the TI-99/4A console. Each floating-point number is stored in 8 bytes (4 cells) with a sign bit, a 7-bit, excess-64 (64-biased) integer exponent of the radix (100) and a normalized, 7-digit (1 radix-100 digit/byte) significand for a total of 8 bytes per floating point number. The signed, radix-100 exponent can be -64 to +63. (Keep in mind that the exponent is for radix-100 notation. Those same exponents radix 10 would be -128 to +126.) The exponent is stored in the most significant byte (MSB) biased by 64, i.e., 64 is added to the actual exponent prior to storing, i.e., -64 to +63 is stored as 0 to 127.

The significand (significant digits of the number) must be normalized, i.e., if the number being represented is not zero, the MSB of the significand must always contain the first non-zero (significant) radix-100 digit, with the radix exponent of such a value that the radix point immediately follows the first digit. This is essentially scientific notation for radix 100. Each byte contains one radix-100 digit of the number, which, of course, means that each byte can have a value from 0 to 99 (0 to 63h) except for the first byte of a non-zero number, which must be 1 to 99. It is easy to view a radix-100 number as a radix-10 number by representing the radix-100 digits as pairs of radix-10 digits because radix 100 is the square of radix 10. In the following list of largest and smallest possible 8-byte floating point numbers, the radix-100 representation is on the left with spaces between pairs of radix-100 digits. The radix-16 (hexadecimal) internal representation of each byte of the number is also shown:

- Largest positive floating point number [hexadecimal: 7F 63 63 63 63 63 63 63]:
 $99.999999999999 \times 100^{63} = 99.999999999999 \times 10^{126}$
 $= 9.9999999999999 \times 10^{127}$
- Largest negative floating point number [hexadecimal: 80 9D 63 63 63 63 63 63]:
 $-99.999999999999 \times 100^{63} = -99.999999999999 \times 10^{126}$
 $= -9.9999999999999 \times 10^{127}$
- Smallest positive floating point number [hexadecimal: 00 01 00 00 00 00 00 00]:
 $01.000000000000 \times 100^{-64} = 1.000000000000 \times 10^{-128}$
- Smallest negative floating point number [hexadecimal: FF FF 00 00 00 00 00 00]:
 $-01.000000000000 \times 100^{-64} = -1.000000000000 \times 10^{-128}$

The only difference in the internal storage of positive and negative floating point numbers is that only the first word (2 bytes) of negative numbers is negated or complemented (two's complement). A floating point zero is represented by zeroing only the first word. The remainder of the floating point number does not need to be zeroed for the number to be treated as zero for all floating point calculations.

All floating point arguments must be on an even byte boundary and the calling registers must be in >F000 to >F060.

CALLING MATH FUNCTIONS

The MDOS Math Library must be called from within a machine code program running as a task under MDOS. You pass arguments to the Math Library via the calling registers.

The MDOS Video Library is invoked from a machine code program when software trap number zero (XOP 0) is called with a library number of 10. The calling program's R0 must contain the 16-bit subprogram at the time of the XOP. The following code fragment will convert a string to a floating point number.

```

      LI      R0,>0011    Convert String to Floating Point
      LI      R1,RES      Result Pointer
      LI      R2,STR1     String Pointer
      LI      R3,3        3 characters to convert
      XOP     @MATH,0     Access subprogram

MATH  DATA  10
STR1  TEXT   "123"
      BYTE   0           Null terminated
RES   DATA  0,0,0,0     Floating Point Result
      EVEN

```

In the preceding example, two hidden assumptions were made. First it is assumed that STR1 is located on a page which is currently mapped into a memory page which has the same 16-bit address page number as its Virtual address page number (read the section on Memory Management.) The second assumption is that MATH is actually at the virtual address MATH, not in some overlay segment with a different virtual address.

Math Library

Floating Point Compare

Function	Floating Point Compare
Parameters	R0 = >0000 R2 = Pointer to Float ₁ R3 = Pointer to Float ₂
Results	Status Register = AG set if (Float ₂ > Float ₁) EQ set if (Float ₂ = Float ₁)

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.

Floating Point Subtract

Function Floating Point Subtract

Parameters R0 = >0001
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁
 R3 = Pointer to Float₂

Results R0 = Error Code
 R1 Pointer = Float₂ – Float₁

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point Add

Function Floating Point Add

Parameters R0 = >0002
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁
 R3 = Pointer to Float₂

Results R0 = Error Code
 R1 Pointer = Float₂ + Float₁

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point Multiply

Function Floating Point Multiply

Parameters R0 = >0003
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁
 R3 = Pointer to Float₂

Results R0 = Error Code
 R1 Pointer = Float₂ * Float₁

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point Divide

Function Floating Point Divide

Parameters R0 = >0004
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁
 R3 = Pointer to Float₂

Results R0 = Error Code
 R1 Pointer = Float₂ / Float₁

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point Power

Function Floating Point Power

Parameters
 R0 = >0005
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁
 R3 = Pointer to Float₂

Results
 R0 = Error Code
 R1 Pointer = Float₂^{Float₁}

Parameter Description

Float1	8 byte representation in Radix 100 format.
Float2	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point e^x

Function Floating Point e^x

Parameters
 R0 = >0006
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results
 R0 = Error Code
 R1 Pointer = e^{Float₁}

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point ln(x)

Function Floating Point ln(x)

Parameters
 R0 = >0007
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results
 R0 = Error Code
 R1 Pointer = ln(Float₁)

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point SQR(x)

Function Floating Point sqr(x)

Parameters R0 = >0008
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results R0 = Error Code
 R1 Pointer = SQR(Float₁)

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point COS(x)

Function Floating Point cos (x)

Parameters R0 = >0009
 R1 = Pointer to Result Destination
 R2 = Pointer to Float1

Results R0 = Error Code
 R1 Pointer = COS(Float₁)

Parameter Description

Float1	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point SIN(x)

Function Floating Point sin(x)

Parameters R0 = >000A
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results R0 = Error Code
 R1 Pointer = SIN(Float₁)

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point TAN(x)

Function Floating Point tan(x)

Parameters R0 = >000B
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results R0 = Error Code
 R1 Pointer = TAN(Float₁)

Parameter Description

Float₁ 8 byte representation in Radix 100 format.

Result 8 byte representation in Radix 100 format.

Error Code R0 = >0000 if no error or error code result.

Floating Point ATN(x)

Function Floating Point atn(x)

Parameters R0 = >000C
 R1 = Pointer to Result Destination
 R2 = Pointer to Float₁

Results R0 = Error Code
 R1 Pointer = ATN(Float₁)

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Floating Point Greatest Integer

Function Floating Point Greatest Integer

Parameters R0 = >000D
R1 = Pointer to Result Destination
R2 = Pointer to Float₁

Results R0 = Error Code
R1 Pointer = || Float₁ ||

Parameter Description

Float ₁	8 byte representation in Radix 100 format.
Result	8 byte representation in Radix 100 format.
Error Code	R0 = >0000 if no error or error code result.

Convert Floating Point to Integer

Function Convert Floating Point to Integer

Parameters R0 = >000E
 R2 = Pointer to Float₁

Results R0 = Error Code
 R1 = Integer

Parameter Description

Float₁ 8 byte representation in Radix 100 format.

Integer 2 byte (16 bit) integer

Error Code R0 = >0000 if no error or error code result.

Convert Integer to Floating Point

Function Convert integer to Floating Point

Parameters R0 = >000F
 R1 = Pointer to Result Destination
 R2 = Integer

Results R0 = Error Code
 R1 Pointer = Float(integer)

Parameter Description

Float1 8 byte representation in Radix 100 format.

Integer 2 byte (16 bit) integer

Error Code R0 = >0000 if no error or error code result.

Convert String to Integer

Function Convert String to Integer

Parameters R0 = >0010
 R2 = Pointer to String

Results R0 = Error Code
 R1 = Integer

Parameter Description

Float1 8 byte representation in Radix 100 format.

String ASCII text string nul terminated.

Error Code R0 = >0000 if no error or error code result.

Convert String to Floating Point

Function Convert String to Floating Point

Parameters R0 = >0011
R1 = Pointer to Result Destination
R2 = Pointer to String
R3 = String Length

Results R0 = Error Code
R1 Pointer = Float(String)

Parameter Description

Float 8 byte representation in Radix 100 format.

String ASCII character text string.

Error Code R0 = >0000 if no error or error code result.

Convert Float to String

Function Convert Float to String

Parameters

R0 = >0012
R1 = Pointer to String
R2 = Pointer to Float₁
R3 = opt1
 Bit 0: 0 = free form (ignore opt₂, opt₃)
 1 = fixed (opt₂, opt₃ are field sizes)
 Bit 1: 1 for explicit sign
 Bit 2: 1 to show sign of positive number as a '+' instead of space
 Bit 3: 1 for extended E-notation
 Bit 4: 1 for extended E-notation (bit 3 must also be 1)
R4 = opt₂ - if fixed format, number of places to left of decimal point,
 including explicit sign.
R5 = opt₃ - if fixed format, number of places to the right of decimal point,
 And including decimal point

If fixed format, with exponent, R4 ,R5 exclude the 3 places for an exponent.

Results

R0 = Error Code
R1 Pointer = String

Parameter Description

Float1	8 byte representation in Radix 100 format.
String	ASCII character text string.
Error Code	R0 = >0000 if no error or error code result.