

Cheryl Ewy
Steven Glass

Integer Math Tools

May 9, 1986

Revision History

March 4, 1986	V00:00	Initial Release
April 22, 1986	V00:10	Int2Dec, Long2Dec, Dec2Int and Dec2Long calls modified
May 9, 1986	V00:20	Errors in the input/output lists for the math routines fixed

STANDARD TOOL SET CALLS

IMBootInit Function number = \$01

This call does nothing.

IMStartUp Function number = \$02

This call does nothing.

IMShutDown Function number = \$03

This call does nothing.

IMVersion Function number = \$04

Input	Word	Space for Result
Output	Word	Result

This call returns the version number for the Integer Math tool set.

IMReset Function number = \$05

IMReset is called when a system reset occurs. It does nothing.

IMAActive Function number = \$06

Input	Word	Space for Result
Output	Word	Result

This call returns a non-zero result indicating that the tool set is active.

MATH ROUTINES

These routines come from the Macintosh and are used throughout the tool box.
Several types of numbers are supported -

Integer	The common single word signed integer
Long Integer	The common double word signed integer
Fixed	A two word signed value with 16 bits of fraction
Frac	A two word signed value with 30 bits of fraction

Multiply

Function number = \$09

Input	LongWord	Space for Result
Input	Word	M1
Input	Word	M2
Output	LongWord	Result

Takes the two 16 bit inputs, multiplies them together and produces a 32 bit result . If the inputs were unsigned, the 32 bit result is unsigned. If the inputs were signed, the low word of the 32 bit result is the signed result.

SDivide

Function number = \$0A

Input	Word	Space for Remainder
Input	Word	Space for Quotient
Input	Word	Numerator
Input	Word	Denominator
Output	Word	Remainder
Output	Word	Quotient

Takes the two 16 bit signed inputs and divides them producing two 16 bit signed results.

UDivide

Function number = \$0B

Input	Word	Space for Remainder
Input	Word	Space for Quotient
Input	Word	Numerator
Input	Word	Denominator
Output	Word	Remainder
Output	Word	Quotient

miss, check if 32 bit division is available

Takes the two 16 bit unsigned inputs and divides them producing two 16 bit unsigned results.

LongMul

Function number = \$0C

Input	LongWord	Space for Result
Input	LongWord	Space for Result
Input	LongWord	M1
Input	LongWord	M2
Output	LongWord	Result (most significant)
Output	LongWord	Result (least significant)

Takes the two 32 bit inputs, multiplies them together and produces a 64 bit result . If the inputs were unsigned, the 64 bit result is unsigned. If the inputs were signed, the low two words of the 64 bit result is the signed result.

LongDivide

Function number = \$0D

Input	LongWord	Space for Remainder
Input	LongWord	Space for Quotient
Input	LongWord	Numerator
Input	LongWord	Denominator
Output	LongWord	Remainder
Output	LongWord	Quotient

Takes the two 32 bit unsigned inputs and divides them producing two 32 bit unsigned results.

FixRatio

Function number = \$0E

ARM CPU

Input	LongWord	Space for Result
Input	Word	Numerator
Input	Word	Denominator
Output	LongWord	Result

Takes the two 16 bit signed inputs and produces a 32 bit fixed point result that is the ratio of the numerator and denominator.

FixMul

Function number = \$0F

Input	LongWord	Space for Result
Input	LongWord	M1
Input	LongWord	M2
Output	LongWord	Result

Takes the two 32 bit fixed point inputs and produces a 32 bit fixed point result that is the product of the inputs.

NOTE - The following math routines have not been implemented yet

FracMul Function number = \$10

Multiplies two Frac inputs and returns a frac result.

FixDiv Function number = \$11

Divides two Fixed inputs and returns a fixed result (no remainder)

FracDiv Function number = \$12

Divides two Frac inputs and returns a Frac result (no remainder)

FixRound Function number = \$13

Takes a Fixed input and returns a rounded integer result.

FracSqrt Function number = \$14

Takes a Frac input and returns a Frac square root.

FracCos Function number = \$15

Takes a Frac input and returns its cosine.

FracSin Function number = \$16

Takes a Frac input and returns its sine.

FixATan2 Function number = \$17

Takes two inputs and returns a fixed point arc tangent of their ratio. The inputs can be long integer, fixed or Frac.

HiWord Function number = \$18

Returns high word of input

LoWord	Function number = \$19	128-bit floating point, value of 0 - 270M
	Returns low word of input.	
Long2Fix	Function number = \$1A	128-bit floating point, value of 0 - 270M
	Converts long integer to fixed.	
Fix2Long	Function number = \$1B	128-bit floating point, value of 0 - 270M
	Converts fixed to long integer.	
Fix2Frac	Function number = \$1C	128-bit floating point, value of 0 - 270M
	Converts fixed to Frac.	
Frac2Fix	Function number = \$1D	128-bit floating point, value of 0 - 270M
	Converts Frac to Fixed.	
Fix2X	Function number = \$1E	128-bit floating point, value of 0 - 270M
	Converts Fixed to extended.	
Frac2X	Function number = \$1F	128-bit floating point, value of 0 - 270M
	Converts Frac to extended.	
X2Fix	Function number = \$20	128-bit floating point, value of 0 - 270M
	Converts exented to Fixed.	
X2Frac	Function number = \$21	128-bit floating point, value of 0 - 270M
	Converts exended to Frac. No boxit required.	

points out, a member and I think needs to be converted to C++

CONVERSION ROUTINES Part 1 of 2

Part 1 of 2 handles 2-byte integers.

These routines convert between a binary value and an ASCII character string representing that value. The binary value can be either a 2-byte integer or a 4-byte integer. The character string can be in either hexadecimal or decimal format.

Int2Hex Function number = \$22

Input Word 2-byte unsigned integer
Input LongWord Pointer to output string
Input LongWord Length of output string
Takes a 2-byte unsigned integer and produces an ASCII string representing the value in hexadecimal format. The string is right-justified and padded at the left with zeros. If the string is too short to represent the value, an error is returned. The ASCII characters in the output string have the high bit clear.

Long2Hex Function number = \$23

Input LongWord 4-byte unsigned integer
Input LongWord Pointer to output string
Input Word Length of output string
Takes a 4-byte unsigned integer and produces an ASCII string representing the value in hexadecimal format. The string is right-justified and padded at the left with zeros. If the string is too short to represent the value, an error is returned. The ASCII characters in the output string have the high bit clear.

Hex2Int Function number = \$24

Input Word Space for result
Input LongWord Pointer to input string
Input Word Length of input string
Output Word 2-byte unsigned integer
Takes an ASCII string representing a hexadecimal value and produces a 2-byte unsigned integer. The string should be right-justified and may be padded at the left with blanks or zeros. The ASCII characters in the string

may have the high bit either set or clear. Illegal characters in the string will cause an error to be returned. If the hexadecimal value is greater than \$FFFF, an overflow error will be returned.

Hex2Long

Function number = \$25
Input LongWord
Input LongWord
Input Word
Output LongWord

Space for Result
Pointer to input string
Length of input string
4-byte unsigned integer

Takes an ASCII string representing a hexadecimal value and produces a 4-byte unsigned integer. The string should be right-justified and may be padded at the left with blanks or zeros. The ASCII characters in the string may have the high bit either set or clear. Illegal characters in the string will cause an error to be returned. If the hexadecimal value is greater than \$FFFFFF, an overflow error will be returned.

Int2Dec

Function number = \$26
Input Word
Input LongWord
Input Word
Input Word

2-byte integer
Pointer to output string
Length of output string
Signed flag

Takes a 2-byte integer and produces an ASCII string representing the value in decimal format. The string is right-justified and padded at the left with blanks. The ASCII characters in the string have the high bit clear. If the Signed flag = 0, the integer will be considered to be unsigned. If the Signed flag > 0, the integer will be considered to be signed. If a signed integer is negative, the string will contain an ASCII minus sign to the left of the most-significant digit. If the string is too short to represent the value, an error is returned.

Long2Dec

Function number = \$27
Input LongWord
Input LongWord
Input Word
Input Word

4-byte integer
Pointer to output string
Length of output string
Signed flag

Takes a 4-byte integer and produces an ASCII string representing the value in decimal format. The string is right-justified and padded at the left with blanks.

ASCII representation with blanks. The ASCII characters in the string have the high bit clear. If the Signed flag = 0, the integer will be considered to be unsigned. If the Signed flag \neq 0, the integer will be considered to be signed. If a signed integer is negative, the string will contain an ASCII minus sign to the left of the most-significant digit. If the string is too short to represent the value, an error is returned.

~~Must not end with a space!~~ ~~String must be right-justified and may be padded at the left with blanks or zeros.~~

~~Signed flag must be 0 or 1.~~ ~~0 = Unsigned, 1 = Signed.~~

Dec2Int ~~Input~~ ~~String~~ ~~Output~~ Function number = \$28

String	Input	String	Space for result
	Input	LongWord	Pointer to input string
	Input	Word	Length of input string
	Input	Word	Signed flag
	Output	Word	2-byte integer

Takes an ASCII string representing a decimal value and produces a 2-byte integer. The string should be right-justified and may be padded at the left with blanks or zeros. The ASCII characters in the string may have the high bit either set or clear. If the Signed flag = 0, the value will be considered to be unsigned. If the Signed flag \neq 0, the value will be considered to be signed. If the value is signed, the string may contain an ASCII plus or minus sign directly in front of the most-significant digit. Illegal characters in the string will cause an error to be returned. If a signed value is greater than 32,767 or less than -32,768 an overflow error will be returned. If an unsigned value is greater than 65,535 an overflow error will be returned.

Dec2Long Function number = \$29

Input	LongWord	Space for Result
Input	LongWord	Pointer to input string
Input	Word	Length of input string
Input	Word	Signed flag
Output	LongWord	4-byte integer

Takes an ASCII string representing a decimal value and produces a 4-byte integer. The string should be right-justified and may be padded at the left with blanks or zeros. The ASCII characters in the string may have the high bit either set or clear. If the Signed flag = 0, the value will be considered to be unsigned. If the Signed flag \neq 0, the value will be considered to be signed. If the value is signed, the string may contain an ASCII plus or minus sign directly in front of the most-significant digit. Illegal characters in the string will cause an error to be returned. If a signed value is greater than 2,147,483,647 or less than -2,147,483,648

an overflow error will be returned. If an unsigned value is greater than 4,294,967,295 an overflow error will be returned.

HexIt

Function number = \$2A
Input LongWord Space for result
Input Word 2-byte unsigned integer
Output LongWord 4-byte hexadecimal string

Takes a 2-byte unsigned integer and returns a 4-byte ASCII string representing the value in hexadecimal format.

Input	Output	Specs
LongWord	LongWord	16.16
Word	Word	16.16
LongWord	LongWord	16.16

Il codice sottostante genera un C++ classe che implementa la funzione HexIt. La classe ha due metodi: uno per il tipo Word e uno per il tipo LongWord. Entrambi i metodi prendono come input un valore di tipo unsigned int e restituiscono una stringa di tipo char* che rappresenta il valore in formato hexadecimale. Il metodo per il tipo Word utilizza la funzione itoa() per convertire il valore in una stringa di 4 caratteri. Il metodo per il tipo LongWord utilizza la funzione itoa() per convertire il valore in una stringa di 8 caratteri. Entrambi i metodi restituiscono una stringa di 8 caratteri, ma solo gli ultimi 4 sono utili per rappresentare il valore in formato hexadecimale.

HexIt - Definition

Input	Output	Specs
unsigned short	char[8]	16.16
unsigned int	char[8]	16.16
long	char[8]	16.16
unsigned long	char[8]	16.16

Il codice sottostante implementa la classe HexIt. La classe ha quattro metodi: uno per il tipo Word, uno per il tipo LongWord, uno per il tipo unsigned int e uno per il tipo long. I primi tre metodi chiamano la funzione itoa() per convertire il valore in una stringa di 4 caratteri. Il quarto metodo chiamano la funzione itoa() per convertire il valore in una stringa di 8 caratteri. Tutti i metodi restituiscono una stringa di 8 caratteri, ma solo gli ultimi 4 sono utili per rappresentare il valore in formato hexadecimale.

ERROR CODES

\$0B01	Bad input parameter
\$0B02	Illegal character in string
\$0B03	Integer or Long Integer overflow
\$0B04	String overflow