Mathematical and Logical Utilities Reference

Carbon > Data Management

2005-11-09

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Contents

Mathematical and Logical Utilities Reference

Framework:	CoreServices/CoreServices.h
Declared in	FixMath.h IOMacOSTypes.h Math64.h ToolUtils.h fenv.h fp.h pyport.h syslog.h

Overview

Important: This is a preliminary document. Although it has been reviewed for technical accuracy, it is not final. Apple Computer is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. For information about updates to this and other developer documentation, you can check the ADC Reference Library Revision List. To receive notification of documentation updates, you can sign up for ADC's free Online Program and receive the weekly Apple Developer Connection News email newsletter. (See http://developer.apple.com/membership for more details about the Online Program.)

You can use the Mathematical and Logical Utilities to perform mathematical and logical operations in Mac OS X programming. This document describes functions you can use to:

- Perform low-level logical manipulation of bits and bytes when using a compiler that does not directly support such manipulations.
- Save disk space by using simple compression and decompression routines.
- Obtain a pseudorandom number.
- Perform mathematical operations with two fixed-point data types supported directly by the Operating System.
- Convert numeric variables of different types.

With the exception of the mathematical operations and conversions, these utilities are intended for programmers who occasionally need to access some of these features and do not require that the algorithms used to implement them be sophisticated. For example, if you are developing an advanced mathematical application, the pseudorandom number generator built into Mac OS might be too simplistic to fit your needs. Similarly, if you wish to access individual bits of memory in a time-critical loop, these routines are probably too slow to be practical.

Carbon supports the Mathematical and Logical Utilities, with the exception of those functions that are 68K-specific. However there are several important differences between the implementation of the Mathematical and Logical Utilities in Mac OS 9 and its implementation in Mac OS X.

The implementation in Carbon on Mac OS X of many floating-point functions defined in `fp.h` is not as accurate as the implementation of those functions in MathLib on Mac OS 8 and 9 (as accessed either directly or through CarbonLib). There are a number of reasons for this difference, including the different expectations of Mac OS 9 and UNIX floating-point clients, compiler limitations, and the need in for an implementation that's independent of assumptions about the size and layout of floating-point data types.

Functions which take parameters or return values of type long double are not exported by the Core Services framework on Mac OS X. Instead, these functions have been replaced with macros that map to the corresponding double-typed functions. While these functions are exported by CarbonLib, CFM applications calling these functions on Mac OS X should note that the implementations of the long double functions on Mac OS X actually have only double precision, with the following four exceptions: `num2decl`, `dec2numl`, `x80told`, and `ldtox80`.

Functions by Task

Converting Among 32-Bit Numeric Types

`Fix2Frac` (page 31)
: Converts a `Fixed` number to a `Fract` number.

`Fix2Long` (page 32)
: Converts a `Fixed` number to a `LongInt` number.

`Frac2Fix` (page 39)
: Converts a `Fract` number to a `Fixed` number.

`Long2Fix` (page 48)
: Converts a `LongInt` number to a `Fixed` number.

Converting Between Fixed-Point and Floating-Point Values

`FixedToFloat` (page 34)
: Converts a `Fixed` number to a `float` number.

`FractToFloat` (page 42)
: Converts a `Fract` number to a `float` number.

`FloatToFixed` (page 36)
: Converts a `float` number to a `Fixed` number.

`FloatToFract` (page 37)
: Converts a `float` number to a `Fract` number.

`Fix2X` (page 32)
: Converts a `Fixed` number to an `Extended` number.

`Frac2X` (page 39)
: Converts a `Fract` number to an `Extended` number.

`X2Fix` (page 78)
Converts an `Extended` number to a `Fixed` number.

`X2Frac` (page 78)
Converts an `Extended` number to a `Fract` number.

Converting Between Fixed-Point and Integral Values

`FixRatio` (page 35)
Obtains the `Fixed` equivalent of a fraction.

`FixRound` (page 36)
Rounds a fixed-point number to the nearest integer.

Getting and Setting Memory Values

`HiWord` (page 44)
Obtains the high-order word of a long word.

`LoWord` (page 49)
Obtains the low-order word of a long word.

Multiplying and Dividing Fixed-Point Numbers

`FixDiv` (page 33)
Divides two variables of the same type (`Fixed`, `Fract`, or `LongInt`) or to divide a `LongInt` or `Fract` number by a `Fixed` number.

`FixMul` (page 34)
Multiplies a variable of type `Fixed` with another variable of type `Fixed` or with a variable of type `Fract` or `LongInt`.

`FracDiv` (page 40)
Divides two variables of the same type (`Fract`, `Fixed`, or `LongInt`) or to divide a `LongInt` or `Fixed` number by a `Fract` number.

`FracMul` (page 41)
Multiplies a variable of type `Fract` with another variable of type `Fract` or with a variable of type `Fixed` or `LongInt`.

Performing Calculations on Fixed-Point Numbers

`FixATan2` (page 33)
Obtains a fast approximation of the arctangent of a fraction.

`FracCos` (page 40)
Obtains a fast approximation of the cosine of a `Fixed` number.

`FracSin` (page 41)
Obtains a fast approximation of the sine of a `Fixed` number.

`FracSqrt` (page 42)
Obtains the square root of a `Fract` number.

Performing Logical Operations

`BitAnd` (page 21)
Performs the `AND` logical operation on two long words.

`BitNot` (page 22)
Performs the `NOT` logical operation on a long word.

`BitOr` (page 22)
Performs the `OR` logical operation on two long words.

`BitShift` (page 23)
Shifts bits in a long word.

`BitXor` (page 24)
Performs the `XOR` logical operation on two long words.

Testing and Setting Bits

`BitClr` (page 21)
Clears a particular bit (to a value of 0).

`BitSet` (page 23)
Sets a particular bit (to a value of 1).

`BitTst` (page 24)
Determines whether a given bit is set.

Miscellaneous Functions

`acos` (page 18)

`acosh` (page 19)

`annuity` (page 19)

`asin` (page 19)

`asinh` (page 20)

`atan` (page 20)

`atan2` (page 20)

`atanh` (page 21)

Functions

acos

```
double_t acos (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

acosh

```
double_t acosh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

annuity

```
double annuity (
   double rate,
   double periods
);
```

Parameters

rate

periods

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

asin

```
double_t asin (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

asinh

```
double_t asinh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

atan

```
double_t atan (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

atan2

```
double_t atan2 (
   double_t y,
   double_t x
);
```

Parameters

y

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

atanh

```
double_t atanh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

BitAnd

Performs the AND logical operation on two long words.

```
long BitAnd (
   long value1,
   long value2
);
```

Parameters

value1
A long word.

value2
A long word.

Return Value
A long word that is the result of the AND operation on the long words passed as arguments. Each bit in the returned value is set if and only if the corresponding bit is set in both value1 and value2.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
ToolUtils.h

BitClr

Clears a particular bit (to a value of 0).

```
void BitClr (
   void *bytePtr,
   long bitNum
);
```

Parameters

bytePtr
A pointer to a byte in memory.

bitNum

The bit to be cleared, specified as a positive offset from the high-order bit of the byte pointed to by the `bytePtr` parameter. The bit being cleared need not be in the same byte pointed to by `bytePtr`.

Special Considerations

The bit numbering scheme used by the `BitClr` function is the opposite of the MC680x0 numbering. To convert an MC680x0 bit number to the format required by the `BitClr` function, subtract the MC680x0 bit number from the highest bit number.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitNot

Performs the `NOT` logical operation on a long word.

```
long BitNot (
   long value
);
```

Parameters

value

A long word.

Return Value
A long word that is the result of the `NOT` operation on the long word passed in as an argument. Each bit in the returned value is set if and only if the corresponding bit is not set in `value`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitOr

Performs the `OR` logical operation on two long words.

```
long BitOr (
   long value1,
   long value2
);
```

Parameters

value1

A long word.

value2

A long word.

Return Value
A long word that is the result of the `OR` operation on the long words passed as arguments. Each bit in the returned value is set if and only if the corresponding bit is set in `value1` or `value2`, or in both `value1` and `value2`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitSet

Sets a particular bit (to a value of 1).

```
void BitSet (
   void *bytePtr,
   long bitNum
);
```

Parameters

`bytePtr`

A pointer to a byte in memory.

`bitNum`

The bit to be set, specified as a positive offset from the high-order bit of the byte pointed to by the `bytePtr` parameter. The bit being set need not be in the byte pointed to by `bytePtr`.

Special Considerations

The bit numbering scheme used by the `BitSet` function is the opposite of the MC680x0 numbering. To convert an MC680x0 bit number to the format required by the `BitSet` function, subtract the MC680x0 bit number from the highest bit number.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitShift

Shifts bits in a long word.

```
long BitShift (
   long value,
   short count
);
```

Parameters

`value`

A long word.

count

The number of bits to shift. If this number is positive, `BitShift` shifts this many positions to the left; if this number is negative, `BitShift` shifts this many positions to the right. The value in this parameter is converted to the result of `MOD` 32.

Return Value
A long word that is the result of shifting the bits in the long word passed in as an argument. The shift's direction and extent are determined by the `count` parameter. Zeroes are shifted into empty positions regardless of the direction of the shift.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitTst

Determines whether a given bit is set.

```
Boolean BitTst (
   const void *bytePtr,
   long bitNum
);
```

Parameters

bytePtr

A pointer to a byte in memory.

bitNum

The bit to be tested, specified as a positive offset from the high-order bit of the byte pointed to by the `bytePtr` parameter. The bit being tested need not be in the byte pointed to by `bytePtr`.

Return Value
`TRUE` if the specified bit is set (that is, has a value of 1) and `FALSE` if the bit is cleared (that is, has a value of 0).

Special Considerations

The bit numbering scheme used by the `BitTst` function is the opposite of the MC680x0 numbering. To convert an MC680x0 bit number to the format required by the `BitTst` function, subtract the MC680x0 bit number from the highest bit number.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`ToolUtils.h`

BitXor

Performs the `XOR` logical operation on two long words.

```
long BitXor (
   long value1,
   long value2
);
```

Parameters

value1

A long word.

value2

A long word.

Return Value

A long word that is the result of the XOR operation on the long words passed in as arguments. Each bit in the returned value is set if and only if the corresponding bit is set in either value1 or value2, but not in both value1 and value2.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

ToolUtils.h

ceil

```
double_t ceil (
   double_t x
);
```

Parameters

x

Return Value

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

Aperture Edit Plugin - Borders & Titles

Declared In

fp.h

compound

```
double compound (
   double rate,
   double periods
);
```

Parameters

rate

periods

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

copysign

```
double_t copysign (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

cos

```
double_t cos (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

cosh

```
double_t cosh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

dec2f

```
float dec2f (
   const decimal *d
);
```

Parameters

d

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

dec2l

```
long dec2l (
   const decimal *d
);
```

Parameters

d

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

dec2num

```
double_t dec2num (
   const decimal *d
);
```

Parameters

d

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

dec2s

```
short dec2s (
   const decimal *d
);
```

Parameters

d

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

dec2str

```
void dec2str (
   const decform *f,
   const decimal *d,
   char *s
);
```

Parameters

f

d

s

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

dtox80

```
void dtox80 (
   const double *x,
   extended80 *x80
);
```

Parameters

x

x80

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

erf

```
double_t erf (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

erfc

```
double_t erfc (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

exp

```
double_t exp (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

exp2

```
double_t exp2 (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

expm1

```
double_t expm1 (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

fabs

```
double_t fabs (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
HID Calibrator
HID Config Save
HID Explorer
SIMD Primer

Declared In
`fp.h`

fdim

```
double_t fdim (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

Fix2Frac

Converts a `Fixed` number to a `Fract` number.

```
Fract Fix2Frac (
   Fixed x
);
```

Parameters

x

The `Fixed` number to be converted to a `Fract` number.

Return Value
The `Fract` number equivalent to the `Fixed` number `x`. If `x` is greater than the maximum representable `Fract` number, the `Fix2Frac` function returns $7FFFFFFF. If `x` is less than the negative number with the highest absolute value, `Fix2Frac` returns $80000000.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

Fix2Long

Converts a `Fixed` number to a `LongInt` number.

```
SInt32 Fix2Long (
   Fixed x
);
```

Parameters

x

The `Fixed` number to be converted to a long integer.

Return Value
The long integer nearest to the `Fixed` number `x`. If `x` is halfway between two integers (0.5), it is rounded to the integer with the higher absolute value.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

Fix2X

Converts a `Fixed` number to an `Extended` number.

```
double Fix2X (
   Fixed x
);
```

Parameters

x

The `Fixed` number to be converted to an `Extended` number.

Return Value
The `Extended` equivalent of the `Fixed` number `x`.

Special Considerations

`Fix2X` does not move memory; you can call it at interrupt time.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

FixATan2

Obtains a fast approximation of the arctangent of a fraction.

```
Fixed FixATan2 (
   SInt32 x,
   SInt32 y
);
```

Parameters

x

The numerator of the fraction whose arctangent is to be obtained. This variable can be a `LongInt`, `Fixed`, or `Fract` number.

y

The denominator of the fraction whose arctangent is to be obtained. The number supplied in this variable must be of the same type as that of the number supplied in the `x` parameter.

Return Value
The arctangent of `y/x`, in radians.

Discussion
The approximation of p/4 used to compute the arctangent is the hexadecimal value 0.C910, making the approximation of p equal to 3.1416015625, while p itself equals 3.14159265.... Thus `FixATan2(1, 1)` equals the equivalent of the hexadecimal value 0.C910. Despite the approximation of p, the arctangent value obtained will usually be correct to several decimal places.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

FixDiv

Divides two variables of the same type (`Fixed`, `Fract`, or `LongInt`) or to divide a `LongInt` or `Fract` number by a `Fixed` number.

```
Fixed FixDiv (
   Fixed x,
   Fixed y
);
```

Parameters

x

The first operand, which can be a variable of type `Fixed` or a variable of type `Fract` or `LongInt`.

y

The second operand, which can be a variable of type `Fixed` or it can be a variable of the same type as the variable in parameter `x`.

Return Value
The quotient of the numbers in `x` and `y`. If the `y` parameter is in the format of a `Fixed` number, then the `x` parameter can be in the format of a `Fixed`, `Fract`, or `LongInt` number. If the `y` parameter is in the format of a `Fract` or `LongInt` number, then the `x` parameter must be in the same format.

The returned value is in the format of a `Fixed` number if both `x` and `y` are both `Fixed` numbers, both `Fract` numbers, or both `LongInt` numbers. Otherwise, the returned value is the same type as the number in the `x` parameter.

Division by zero results in $8000000 if `x` is negative, and $7FFFFFFF otherwise; thus the special case 0/0 yields $7FFFFFFF.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
SoftVDigX

Declared In
`FixMath.h`

FixedToFloat

Converts a `Fixed` number to a `float` number.

```
float FixedToFloat (
    Fixed x
);
```

Parameters

x
The `Fixed` number to be converted.

Return Value
The `float` equivalent of the `Fixed` number.

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`FixMath.h`

FixMul

Multiplies a variable of type `Fixed` with another variable of type `Fixed` or with a variable of type `Fract` or `LongInt`.

```
Fixed FixMul (
   Fixed a,
   Fixed b
);
```

Parameters

a

The first operand, which can be a variable of type `Fixed` or a variable of type `Fract` or `LongInt`.

b

The second operand, which can be a variable of type `Fixed` or a variable of type `Fract` or `LongInt`.

Return Value
The product of the numbers in `a` and `b`. At least one of `a` and `b` should be a variable of type `Fixed`.

The returned value is in the format of a `LongInt` if one of `a` or `b` is a `LongInt`. It is a `Fract` number if one of `a` or `b` is `Fract`. It is a `Fixed` number if both `a` and `b` are `Fixed` numbers.

Overflows are set to the maximum representable value with the correct sign ($80000000 for negative results and $7FFFFFFF for positive results).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FixRatio

Obtains the `Fixed` equivalent of a fraction.

```
Fixed FixRatio (
   short numer,
   short denom
);
```

Parameters

numer

The numerator of the fraction.

denom

The denominator of the fraction.

Return Value
The `Fixed` equivalent of the fraction `numer/denom`.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
SoftVDigX

Declared In
`FixMath.h`

FixRound

Rounds a fixed-point number to the nearest integer.

```
short FixRound (
   Fixed x
);
```

Parameters

x

The `Fixed` number to be rounded.

Return Value

The `Integer` number nearest the `Fixed` number `x`. If the value is halfway between two integers (0.5), it is rounded up. Thus, 4.5 is rounded to 5, and –3.5 is rounded to –3.

Discussion

To round a negative `Fixed` number so that values halfway between two integers are rounded to the number with the higher absolute value, negate the number, round it, and then negate it again.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

SoftVDigX

Declared In

`FixMath.h`

FloatToFixed

Converts a `float` number to a `Fixed` number.

```
Fixed FloatToFixed (
    float x
);
```

Parameters

x

The `float` number to be converted.

Return Value

The `Fixed` equivalent of the `float` number.

Discussion

This function is implemented as an inline macro.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`FixMath.h`

FloatToFract

Converts a `float` number to a `Fract` number.

```
Fract FloatToFract (
    float x
);
```

Parameters

x

The `float` number to be converted.

Return Value

The `Fract` equivalent of the `float` number.

Discussion

This function is implemented as an inline macro.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`FixMath.h`

floor

```
double_t floor (
   double_t x
);
```

Parameters

x

Return Value

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

Aperture Edit Plugin - Borders & Titles

WhackedTV

Declared In

`fp.h`

fmax

```
double_t fmax (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

fmin

```
double_t fmin (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

fmod

```
double_t fmod (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

fpclassify

```
long fpclassify (
    float x
);
```

Parameters

x

A value of type `float` or `double`.

Return Value
Returns one of the FP_ values. See `FP_SNAN` (page 85).

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

Frac2Fix

Converts a `Fract` number to a `Fixed` number.

```
Fixed Frac2Fix (
   Fract x
);
```

Parameters

x

The `Fract` number to be converted to a `Fixed` number.

Return Value
The `Fixed` number that best approximates the `Fract` number `x`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

Frac2X

Converts a `Fract` number to an `Extended` number.

```
double Frac2X (
   Fract x
);
```

Parameters

x

The `Fract` number to be converted to an `Extended` number.

Return Value
The `Extended` equivalent of the `Fract` number `x`.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FracCos

Obtains a fast approximation of the cosine of a `Fixed` number.

```
Fract FracCos (
   Fixed x
);
```

Parameters

x

The `Fixed` number expressed in radians, whose cosine is to be calculated.

Return Value
The cosine, expressed in radians, of the `Fixed` number `x`.

Discussion
The approximation of p/4 used to compute the cosine is the hexadecimal value 0.C910, making the approximation of p equal to 3.1416015625, while p itself equals 3.14159265.... Despite the approximation of p, the cosine value obtained is usually correct to several decimal places.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FracDiv

Divides two variables of the same type (`Fract`, `Fixed`, or `LongInt`) or to divide a `LongInt` or `Fixed` number by a `Fract` number.

```
Fract FracDiv (
   Fract x,
   Fract y
);
```

Parameters

x

The first operand, which can be a variable of type `Fract` or a variable of type `Fixed` or `LongInt`.

y

The second operand, which can be a variable of type `Fract` or a variable of the same type as the variable in parameter `a`.

Return Value
The quotient of the numbers in `a` and `b`. If the `b` parameter is in the format of a `Fract` number, then the `a` parameter can be in the format of a `Fract`, a `Fixed`, or a `LongInt` number. If the `b` parameter is in the format of a `Fixed` or a `LongInt` number, then the `a` parameter must be in the same format.

The returned value is in the format of a `Fract` number if `a` and `b` are both `Fract` numbers, both `Fixed` numbers, or both `LongInt` numbers. Otherwise, the returned value is in the same format as the number in the `a` parameter.

Division by zero results in $8000000 if `a` is negative, and $7FFFFFFF otherwise; thus the special case 0/0 yields $7FFFFFFF.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FracMul

Multiplies a variable of type `Fract` with another variable of type `Fract` or with a variable of type `Fixed` or `LongInt`.

```
Fract FracMul (
   Fract x,
   Fract y
);
```

Parameters

x
The first operand, which can be a variable of type `Fract` or a variable of type `Fixed` or `LongInt`.

y
The second operand, which can be a variable of type `Fract` or a variable of type `Fixed` or `LongInt`.

Return Value
The product of the numbers in `a` and `b`. At least one of `a` or `b` should be a variable of type `Fract`.

The returned value is in the format of a `LongInt` number if one of `a` and `b` is a `LongInt` number. It is a `Fixed` number if one of `a` or `b` is a `Fixed` number. It is a `Fract` number if both `a` and `b` are `Fract` numbers.

Overflows are set to the maximum representable value with the correct sign ($80000000 for negative results and $7FFFFFFF for positive results).

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FracSin

Obtains a fast approximation of the sine of a `Fixed` number.

```
Fract FracSin (
   Fixed x
);
```

Parameters

x

The `Fixed` number expressed in radians, whose sine is to be calculated.

Return Value
The sine, expressed in radians, of the `Fixed` number `x`.

Discussion
The approximation of p/4 used to compute the sine is the hexadecimal value 0.C910, making the approximation of p equal to 3.1416015625, while p itself equals 3.14159265.... Despite the approximation of p, the sine value obtained is usually correct to several decimal places.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FracSqrt

Obtains the square root of a `Fract` number.

```
Fract FracSqrt (
   Fract x
);
```

Parameters

x

The `Fract` number to obtain a square root of. This parameter is interpreted as being unsigned in the range 0 through 4 – 2–30, inclusive. That is, the bit of the `Fract` number that ordinarily has weight -2 is instead interpreted as having weight 2.

Return Value
The square root of the specified `Fract` number. The result is unsigned in the range 0 through 2, inclusive.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`FixMath.h`

FractToFloat

Converts a `Fract` number to a `float` number.

```
float FixedToFract (
    Fract x
);
```

Parameters

x

The `Fract` number to be converted.

Return Value

The `float` equivalent of the `Fract` number.

Discussion

This function is implemented as an inline macro.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`FixMath.h`

frexp

```
double_t frexp (
   double_t x,
   int *exponent
);
```

Parameters

x

exponent

Return Value

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`fp.h`

gamma

```
double_t gamma (
   double_t x
);
```

Parameters

x

Return Value

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

Gamma Filter for FxPlug and AE

SoftVDigX

Declared In
`fp.h`

HiWord

Obtains the high-order word of a long word.

```
SInt16 HiWord (
    SInt32 x
);
```

Parameters

x

The long word whose high word is to be returned.

Return Value
The high-order word of the long word specified by the `x` parameter.

Discussion
One use of this function is to obtain the integral part of a fixed-point number.

To copy a range of bytes from one memory location to another, you should ordinarily use the Memory Manager function, `BlockMove`.

Availability

Declared In
`ToolUtils.h`

hypot

```
double_t hypot (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`pyport.h`

isfinite

```
long isfinite (
    float x
);
```

Parameters

x

A value of type `float` or `double`.

Return Value
Returns a non-zero value only if the argument is finite.

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

isnan

```
long isnan (
    float x
);
```

Parameters

x

A value of type `float` or `double`.

Return Value
Returns a non-zero value only if the argument is not a number (NaN).

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

isnormal

```
long isnormal (
    float x
);
```

Parameters

x

A value of type `float` or `double`.

Return Value
Returns a non-zero value only if the argument is normalized.

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

ldexp

```
double_t ldexp (
   double_t x,
   int n
);
```

Parameters

x

n

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

lgamma

```
double_t lgamma (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

log

```
double_t log (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
dist_fft
FBOBunnies
FilterDemo
LSMSmartCategorizer
VelEng Multiprecision

Declared In
`syslog.h`

log10

```
double_t log10 (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
WhackedTV

Declared In
`fp.h`

log1p

```
double_t log1p (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

log2

```
double_t log2 (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

logb

```
double_t logb (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

Long2Fix

Converts a `LongInt` number to a `Fixed` number.

```
Fixed Long2Fix (
   SInt32 x
);
```

Parameters

x

The long integer to be converted to a `Fixed` number.

Return Value
The `Fixed` number equivalent to the long integer `x`. If `x` is greater than the maximum representable fixed-point number, the `Long2Fix` function returns $7FFFFFFF. If `x` is less than the negative number with the highest absolute value, `Long2Fix` returns $80000000.

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
ASCIIMoviePlayerSample

Declared In
`FixMath.h`

LoWord

Obtains the low-order word of a long word.

```
SInt16 LoWord (
    SInt32 x
);
```

Parameters

x

The long word whose low word is to be returned.

Return Value
The low-order word of the long word specified by the `x` parameter.

Discussion
One use of this function is to obtain the fractional part of a fixed-point number.

To copy a range of bytes from one memory location to another, you should ordinarily use the Memory Manager function, `BlockMove`.

Availability

Declared In
`ToolUtils.h`

modf

```
double_t modf (
   double_t x,
   double_t *iptr
);
```

Parameters

x

iptr

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

modff

```
float modff (
   float x,
   float *iptrf
);
```

Parameters

x

iptrf

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

nan

```
double nan (
   const char *tagp
);
```

Parameters

tagp

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

nanf

```
float nanf (
   const char *tagp
);
```

Parameters

tagp

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

nearbyint

```
double_t nearbyint (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

nextafterd

```
double nextafterd (
   double x,
   double y
);
```

Parameters

x

y

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

nextafterf

```
float nextafterf (
   float x,
   float y
);
```

Parameters

x

y

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

num2dec

```
void num2dec (
   const decform *f,
   double_t x,
   decimal *d
);
```

Parameters

f

x

d

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

pi

```
pi ();
```

Parameters

Return Value

Availability

Declared In
fp.h

pow

```
double_t pow (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
Gamma Filter for FxPlug and AE

WhackedTV

Declared In
`fp.h`

randomx

```
double_t randomx (
   double_t *x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

relation

```
relop relation (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

remainder

```
double_t remainder (
   double_t x,
   double_t y
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
SoftVDigX

Declared In
fp.h

remquo

```
double_t remquo (
   double_t x,
   double_t y,
   int *quo
);
```

Parameters

x

y

quo

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

rint

```
double_t rint (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

rinttol

```
long rinttol (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

round

```
double_t round (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

roundtol

```
long roundtol (
   double_t round
);
```

Parameters

round

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

S32Set

```
SInt32 S32Set (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Absolute

```
SInt64 S64Absolute (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Add

```
SInt64 S64Add (
   SInt64 left,
   SInt64 right
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64And

```
Boolean S64And (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64BitwiseAnd

```
SInt64 S64BitwiseAnd (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64BitwiseEor

```
SInt64 S64BitwiseEor (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64BitwiseNot

```
SInt64 S64BitwiseNot (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64BitwiseOr

```
SInt64 S64BitwiseOr (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Compare

```
SInt32 S64Compare (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Div

```
SInt64 S64Div (
   SInt64 dividend,
   SInt64 divisor
);
```

Parameters

dividend

divisor

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Divide

```
SInt64 S64Divide (
   SInt64 dividend,
   SInt64 divisor,
   SInt64 *remainder
);
```

Parameters

dividend

divisor

remainder

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Eor

```
Boolean S64Eor (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Max

```
SInt64 S64Max (
   void
);
```

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Min

```
SInt64 S64Min (
   void
);
```

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Multiply

```
SInt64 S64Multiply (
   SInt64 left,
   SInt64 right
);
```

Parameters

xparam

yparam

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Negate

```
SInt64 S64Negate (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Not

```
Boolean S64Not (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Or

```
Boolean S64Or (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Set

```
SInt64 S64Set (
   SInt32 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64SetU

```
SInt64 S64SetU (
   UInt32 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64ShiftLeft

```
SInt64 S64ShiftLeft (
   SInt64 value,
   UInt32 shift
);
```

Parameters

value

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64ShiftRight

```
SInt64 S64ShiftRight (
   SInt64 value,
   UInt32 shift
);
```

Parameters

value

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

S64Subtract

```
SInt64 S64Subtract (
   SInt64 left,
   SInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

scalb

```
double_t scalb (
   double_t x,
   _scalb_n_type n
);
```

Parameters

x

n

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

signbit

```
long signbit (
    float x
);
```

Parameters

x

A value of type `float` or `double`, NaN, infinity, or zero.

Return Value
Returns a non-zero value only if the sign of the argument is negative.

Discussion
This function is implemented as an inline macro.

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

sin

```
double_t sin (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

sinh

```
double_t sinh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

SInt64ToUInt64

```
UInt64 SInt64ToUInt64 (
   SInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`Math64.h`

sqrt

```
double_t sqrt (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

str2dec

```
void str2dec (
   const char *s,
   short *ix,
   decimal *d,
   short *vp
);
```

Parameters

s

ix

d

vp

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

tan

```
double_t tan (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

tanh

```
double_t tanh (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

trunc

```
_trunc_return_type trunc (
   double_t x
);
```

Parameters

x

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
fp.h

U32SetU

```
UInt32 U32SetU (
   UInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Add

```
UInt64 U64Add (
   UInt64 left,
   UInt64 right
);
```

Parameters

x

y

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64And

```
Boolean U64And (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64BitwiseAnd

```
UInt64 U64BitwiseAnd (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64BitwiseEor

```
UInt64 U64BitwiseEor (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64BitwiseNot

```
UInt64 U64BitwiseNot (
   UInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64BitwiseOr

```
UInt64 U64BitwiseOr (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Compare

```
SInt32 U64Compare (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Div

Under evaluation

```
UInt64 U64Div (
   UInt64 dividend,
   UInt64 divisor
);
```

Parameters

dividend

divisor

Return Value

Availability

Declared In
Math64.h

U64Divide

```
UInt64 U64Divide (
   UInt64 dividend,
   UInt64 divisor,
   UInt64 *remainder
);
```

Parameters

dividend

divisor

remainder

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Eor

```
Boolean U64Eor (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Max

```
UInt64 U64Max (
   void
);
```

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Multiply

```
UInt64 U64Multiply (
   UInt64 left,
   UInt64 right
);
```

Parameters

xparam

yparam

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Not

```
Boolean U64Not (
   UInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Or

```
Boolean U64Or (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Set

```
UInt64 U64Set (
   SInt32 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64SetU

```
UInt64 U64SetU (
   UInt32 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64ShiftLeft

```
UInt64 U64ShiftLeft (
   UInt64 value,
   UInt32 shift
);
```

Parameters

value

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64ShiftRight

```
UInt64 U64ShiftRight (
   UInt64 value,
   UInt32 shift
);
```

Parameters

value

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

U64Subtract

```
UInt64 U64Subtract (
   UInt64 left,
   UInt64 right
);
```

Parameters

left

right

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

UInt64ToSInt64

```
SInt64 UInt64ToSInt64 (
   UInt64 value
);
```

Parameters

value

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
Math64.h

WideAdd

```
wide * WideAdd (
   wide *target,
   const wide *source
);
```

Parameters

target

source

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Related Sample Code
SoftVDigX

Declared In
FixMath.h

WideBitShift

```
wide * WideBitShift (
   wide *target,
   SInt32 shift
);
```

Parameters

src

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideCompare

```
short WideCompare (
   const wide *target,
   const wide *source
);
```

Parameters

target

source

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideDivide

```
SInt32 WideDivide (
   const wide *dividend,
   SInt32 divisor,
   SInt32 *remainder
);
```

Parameters

dividend

divisor

remainder

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideMultiply

```
wide * WideMultiply (
   SInt32 multiplicand,
   SInt32 multiplier,
   wide *target
);
```

Parameters

multiplicand

multiplier

target

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideNegate

```
wide * WideNegate (
   wide *target
);
```

Parameters

target

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideShift

```
wide * WideShift (
   wide *target,
   SInt32 shift
);
```

Parameters

target

shift

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideSquareRoot

```
UInt32 WideSquareRoot (
   const wide *source
);
```

Parameters

source

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideSubtract

```
wide * WideSubtract (
   wide *target,
   const wide *source
);
```

Parameters

target

source

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

WideWideDivide

```
wide * WideWideDivide (
   wide *dividend,
   SInt32 divisor,
   SInt32 *remainder
);
```

Parameters

dividend

divisor

remainder

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
FixMath.h

X2Fix

Converts an `Extended` number to a `Fixed` number.

```
Fixed X2Fix (
   double x
);
```

Parameters

x

The `Extended` number to be converted to a `Fixed` number.

Return Value

The best `Fixed` approximation of the `Extended` number `x`. If `x` is greater than the maximum representable `Fixed` number, the `X2Fix` function returns $7FFFFFFF. If `x` is less than the negative number with the highest absolute value, `X2Fix` returns $80000000.

Availability

Available in Mac OS X version 10.0 and later.

Related Sample Code

LiveVideoMixer2

Declared In

`FixMath.h`

X2Frac

Converts an `Extended` number to a `Fract` number.

```
Fract X2Frac (
   double x
);
```

Parameters

x

The `Extended` number to be converted to a `Fract` number.

Return Value

The best `Fract` approximation of the `Extended` number `x`. If `x` is greater than the maximum representable `Fract` number, the `X2Frac` function returns $7FFFFFFF. If `x` is less than the negative number with the highest absolute value, `X2Frac` returns $80000000.

Availability

Available in Mac OS X version 10.0 and later.

Declared In

`FixMath.h`

x80tod

```
double x80tod (
   const extended80 *x80
);
```

Parameters

x80

Return Value

Availability
Available in Mac OS X version 10.0 and later.

Declared In
`fp.h`

Data Types

decform

```
struct decform {
    char style;
    char unused;
    short digits;
};
typedef struct decform decform;
```

Fields
`style`
`unused`
`digits`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`fp.h`

decimal

```
struct decimal {
    char sgn
    char unused
    short exp
    struct {
        unsigned char length;
        unsigned char text[36];
        unsigned char pad;
    } sig;
};
typedef struct decimal decimal;
```

Fields

`sgn`

`unused`

`exp`

`length`

`text`

`pad`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`fp.h`

double_t

```
typedef double double_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`fp.h`

fenv_t

```
typedef SInt32 fenv_t;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
`fenv.h`

fexcept_t

```
typedef SInt32 fexcept_t;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
`fenv.h`

Fixed

Defines a data type for fixed-point decimal numbers.

```
typedef SInt32 Fixed;
```

Discussion
This data type uses a 16-bit signed integer and a 16-bit fraction to represent fixed-point decimal numbers in the interval:

$$[-32768, 32767 + ((2^{16} - 1) / 2^{16})]$$

For example, the number 1.5 would be represented as `0x00018000`, and the number -1.3 would be represented as `0xFFFEB334`. To convert numbers between `Fixed` and `float`, you can use the functions `FixedToFloat` (page 34) and `FloatToFixed` (page 36).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`IOMacOSTypes.h`

Fract

Defines a high-precision data type for fixed-point decimal numbers.

```
typedef SInt32 Fract;
```

Discussion
This data type uses a 2-bit signed integer and a 30-bit fraction to represent fixed-point decimal numbers in the interval

$$[-2, 1 + ((2^{30} - 1) / 2^{30})]$$

with higher precision than the `Fixed` (page 81) data type. For example, the number 1.5 would be represented as `0x60000000`, and the number -1.3 would be represented as `0xACCCCCCD`. To convert numbers between `Fract` and `float`, you can use the functions `FractToFloat` (page 42) and `FloatToFract` (page 37).

Availability
Available in Mac OS X v10.0 and later.

Declared In
IOMacOSTypes.h

float_t

```
typedef float float_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
fp.h

relop

```
typedef short relop;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
fp.h

_scalb_n_type

```
typedef int _scalb_n_type;
```

_trunc_return_type

```
typedef double_t _trunc_return_type;
```

Constants

DECSTROUTLEN

```
enum {
    DECSTROUTLEN = 80
};
```

Constants
DECSTROUTLEN

FE_INEXACT

Definitions of floating-point exception macros.

```
enum {
    FE_INEXACT                  = 0x02000000,
    FE_DIVBYZERO                = 0x04000000,
    FE_UNDERFLOW                = 0x08000000,
    FE_OVERFLOW                 = 0x10000000,
    FE_INVALID                  = 0x20000000,
    FE_ALL_EXCEPT               = 0x3E000000
};
```

Constants

FE_INEXACT
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

FE_DIVBYZERO
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

FE_UNDERFLOW
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

FE_OVERFLOW
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

`FE_INVALID`
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

`FE_ALL_EXCEPT`
Available in Mac OS X v10.1 through Mac OS X v10.1.
Declared in `fenv.h`.

FE_LDBLPREC

```
enum {
    FE_LDBLPREC = 0,
    FE_DBLPREC = 1,
    FE_FLTPREC = 2
};
```

Constants
`FE_LDBLPREC`
`FE_DBLPREC`
`FE_FLTPREC`

FE_TONEAREST

Definitions of rounding direction macros.

```
enum {
    FE_TONEAREST                = 0x00000000,
    FE_TOWARDZERO               = 0x00000001,
    FE_UPWARD                   = 0x00000002,
    FE_DOWNWARD                 = 0x00000003
};
```

Constants
`FE_TONEAREST`
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

`FE_TOWARDZERO`
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

`FE_UPWARD`
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

`FE_DOWNWARD`
Available in Mac OS X v10.0 through Mac OS X v10.1.
Declared in `fenv.h`.

fixed1

```
enum {
    fixed1 = 0x00010000,
    fract1 = 0x40000000,
    positiveInfinity = 0x7FFFFFFF,
    negativeInfinity = 0x80000000
};
```

Constants

`fixed1`

`fract1`

`positiveInfinity`

`negativeInfinity`

FP_SNAN

```
enum {
    FP_SNAN = 0,
    FP_QNAN = 1,
    FP_INFINITE = 2,
    FP_ZERO = 3,
    FP_NORMAL = 4,
    FP_SUBNORMAL = 5
};
```

Constants

`FP_SNAN`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

`FP_QNAN`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

`FP_INFINITE`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

`FP_ZERO`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

`FP_NORMAL`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

`FP_SUBNORMAL`

Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared in `fp.h`.

Relational Operator

```
typedef short relop;
enum {
    GREATERTHAN = 0,
    LESSTHAN = 1,
    EQUALTO = 2,
    UNORDERED = 3
};
```

Constants

`GREATERTHAN`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

`LESSTHAN`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

`EQUALTO`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

`UNORDERED`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

SIGDIGLEN

```
enum {
    SIGDIGLEN = 36
};
```

Constants

`SIGDIGLEN`

Special Values

```
#define HUGE_VAL
#define INFINITY
```

Constants

`HUGE_VAL`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

`INFINITY`
Available in Mac OS X v10.0 and later.
Declared in `fp.h`.

Document Revision History

This table describes the changes to *Mathematical and Logical Utilities Reference*.

Date	Notes
2005-11-09	Updated availability information.
2005-07-07	Added descriptions of Fixed and Fract data types.
2003-02-20	Updated for Mac OS X version 10.2.

Document Revision History

Index

G

H

I

L

M

N

P

R

S

T

U

W

X