Math Library

User Reference Manual

56800E, 56800Ex
Digital Signal Controller

56800Ex_MLIB
Rev. 0
02/2014

freescale.com

Chapter 1 License Agreement

FREESCALE SEMICONDUCTOR SOFTWARE LICENSE AGREEMENT. This is a legal agreement between you (either as an individual or as an authorized representative of your employer) and Freescale Semiconductor, Inc. ("Freescale"). It concerns your rights to use this file and any accompanying written materials (the "Software"). In consideration for Freescale allowing you to access the Software, you are agreeing to be bound by the terms of this Agreement. If you do not agree to all of the terms of this Agreement, do not download the Software. If you change your mind later, stop using the Software and delete all copies of the Software in your possession or control. Any copies of the Software that you have already distributed, where permitted, and do not destroy will continue to be governed by this Agreement. Your prior use will also continue to be governed by this Agreement.

OBJECT PROVIDED, OBJECT REDISTRIBUTION LICENSE GRANT. Freescale grants to you, free of charge, the non-exclusive, non-transferable right (1) to reproduce the Software, (2) to distribute the Software, and (3) to sublicense to others the right to use the distributed Software. The Software is provided to you only in object (machine-readable) form. You may exercise the rights above only with respect to such object form. You may not translate, reverse engineer, decompile, or disassemble the Software except to the extent applicable law specifically prohibits such restriction. In addition, you must prohibit your sublicensees from doing the same. If you violate any of the terms or restrictions of this Agreement, Freescale may immediately terminate this Agreement, and require that you stop using and delete all copies of the Software in your possession or control.

COPYRIGHT. The Software is licensed to you, not sold. Freescale owns the Software, and United States copyright laws and international treaty provisions protect the Software. Therefore, you must treat the Software like any other copyrighted material (e.g. a book or musical recording). You may not use or copy the Software for any other purpose than what is described in this Agreement. Except as expressly provided herein, Freescale does not grant to you any express or implied rights under any Freescale or third-party patents, copyrights, trademarks, or trade secrets. Additionally, you must reproduce and apply any copyright or other proprietary rights notices included on or embedded in the Software to any copies or derivative works made thereof, in whole or in part, if any.

SUPPORT. Freescale is NOT obligated to provide any support, upgrades or new releases of the Software. If you wish, you may contact Freescale and report problems and provide suggestions regarding the Software. Freescale has no obligation whatsoever to respond in any way to such a problem report or suggestion. Freescale may make changes to the Software at any time, without any obligation to notify or provide updated versions of the Software to you.

NO WARRANTY. TO THE MAXIMUM EXTENT PERMITTED BY LAW, FREESCALE EXPRESSLY DISCLAIMS ANY WARRANTY FOR THE

SOFTWARE. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. YOU ASSUME THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE SOFTWARE, OR ANY SYSTEMS YOU DESIGN USING THE SOFTWARE (IF ANY). NOTHING IN THIS AGREEMENT MAY BE CONSTRUED AS A WARRANTY OR REPRESENTATION BY FREESCALE THAT THE SOFTWARE OR ANY DERIVATIVE WORK DEVELOPED WITH OR INCORPORATING THE SOFTWARE WILL BE FREE FROM INFRINGEMENT OF THE INTELLECTUAL PROPERTY RIGHTS OF THIRD PARTIES.

INDEMNITY. You agree to fully defend and indemnify Freescale from any and all claims, liabilities, and costs (including reasonable attorney's fees) related to (1) your use (including your sublicensee's use, if permitted) of the Software or (2) your violation of the terms and conditions of this Agreement.

LIMITATION OF LIABILITY. IN NO EVENT WILL FREESCALE BE LIABLE, WHETHER IN CONTRACT, TORT, OR OTHERWISE, FOR ANY INCIDENTAL, SPECIAL, INDIRECT, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR ANY LOSS OF USE, LOSS OF TIME, INCONVENIENCE, COMMERCIAL LOSS, OR LOST PROFITS, SAVINGS, OR REVENUES TO THE FULL EXTENT SUCH MAY BE DISCLAIMED BY LAW.

COMPLIANCE WITH LAWS; EXPORT RESTRICTIONS. You must use the Software in accordance with all applicable U.S. laws, regulations and statutes. You agree that neither you nor your licensees (if any) intend to or will, directly or indirectly, export or transmit the Software to any country in violation of U.S. export restrictions.

GOVERNMENT USE. Use of the Software and any corresponding documentation, if any, is provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(l) and (2) of the Commercial Computer Software--Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is Freescale Semiconductor, Inc., 6501 William Cannon Drive West, Austin, TX, 78735.

HIGH RISK ACTIVITIES. You acknowledge that the Software is not fault tolerant and is not designed, manufactured or intended by Freescale for incorporation into products intended for use or resale in on-line control equipment in hazardous, dangerous to life or potentially life-threatening environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems, in which the failure of products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). You specifically represent and warrant that you will not use the Software or any derivative work of the Software for High Risk Activities.

CHOICE OF LAW; VENUE; LIMITATIONS. You agree that the statutes and laws of the United States and the State of Texas, USA, without regard to conflicts of laws principles, will apply to all matters relating to this Agreement or the Software, and you agree that any litigation will be subject to the exclusive jurisdiction of the state or federal courts in Texas, USA. You agree that regardless of any statute or law to the contrary, any claim or cause of action arising out of or related to this Agreement or the Software must be filed within one (1) year after such claim or cause of action arose or be forever barred.

PRODUCT LABELING. You are not authorized to use any Freescale trademarks, brand names, or logos.

ENTIRE AGREEMENT. This Agreement constitutes the entire agreement between you and Freescale regarding the subject matter of this Agreement, and supersedes all prior communications, negotiations, understandings, agreements or representations, either written or oral, if any. This Agreement may only be amended in written form, executed by you and Freescale.

SEVERABILITY. If any provision of this Agreement is held for any reason to be invalid or unenforceable, then the remaining provisions of this Agreement will be unimpaired and, unless a modification or replacement of the invalid or unenforceable provision is further held to deprive you or Freescale of a material benefit, in which case the Agreement will immediately terminate, the invalid or unenforceable provision will be replaced with a provision that is valid and enforceable and that comes closest to the intention underlying the invalid or unenforceable provision.

NO WAIVER. The waiver by Freescale of any breach of any provision of this Agreement will not operate or be construed as a waiver of any other or a subsequent breach of the same or a different provision.

Chapter 2 INTRODUCTION

2.1 Overview

This reference manual describes the Math Library (MLIB) for the Freescale 56F800E(X) family of Digital Signal Controllers. This library contains optimized functions.

2.2 Supported Compilers

Math Library (MLIB) is written in assembly language with C-callable interface. The library was built and tested using the CodeWarrior™ Development Studio version 10.3.

The library is delivered in library module 56800Ex_MLIB.lib and is intended for use in small data memory model projects. The interfaces to the algorithms included in this library have been combined into a single public interface include file, gdflib.h. This was done to simplify the number of files required for inclusion by application programs. Refer to the specific algorithm sections of this document for details on the software Application Programming Interface (API), defined and functionality provided for the individual algorithms.

2.3 Installation

If user wants to fully use this library, the CodeWarrior™ Development Studio should be installed prior to the Math Library. In case that Math Library is installed while CodeWarrior™ Development Studio is not present, users can only browse the installed software package, but will not be able to build, download and run code. The installation itself consists of copying the required files to the destination hard drive, checking the presence of CodeWarrior and creating the shortcut under the Start->Programs menu.

The Math Library release is installed in its own folder named 56800Ex_MLIB.

To start the installation process, perform the following steps:

1. Execute 56800Ex_FSLESL_rXX.exe.
2. Follow the FSLESL software installation instructions on your screen.

2.4 Library Integration

The library integration is described in AN4586 which can be downloaded from www.freescale.com.

2.5 API Definition

The description of each function described in this Math Library user reference manual consists of a number of subsections:

Synopsis

This subsection gives the header files that should be included within a source file that references the function or macro. It also shows an appropriate declaration for the function or for a function that can be substituted by a macro. This declaration is not included in your program; only the header file(s) should be included.

Prototype

This subsection shows the original function prototype declaration with all its arguments.

Arguments

This optional subsection describes input arguments to a function or macro.

Description

This subsection is a description of the function or macro. It explains algorithms being used by functions or macros.

Return

This optional subsection describes the return value (if any) of the function or macro.

Range Issues

This optional subsection specifies the ranges of input variables.

Special Issues

This optional subsection specifies special assumptions that are mandatory for correct function calculation; for example saturation, rounding, and so on.

Implementation

This optional subsection specifies, whether a call of the function generates a library function call or a macro expansion.
This subsection also consists of one or more examples of the use of the function. The examples are often fragments of code (not completed programs) for illustration purposes.

See Also

This optional subsection provides a list of related functions or macros.

Performance

This section specifies the actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

2.6 Data Types

The 16-bit DSC core supports four types of two's-complement data formats:

- Signed integer
- Unsigned integer
- Signed fractional
- Unsigned fractional

Signed and unsigned integer data types are useful for general-purpose computation; they are familiar with the microprocessor and microcontroller programmers. Fractional data types allow powerful numeric and digital-signal-processing algorithms to be implemented.

2.6.1 Signed Integer (SI)

This format is used for processing data as integers. In this format, the N-bit operand is represented using the N.0 format (N integer bits). The signed integer numbers lie in the following range:

$$-2^{[N-1]} \leq SI \leq [2^{[N-1]} - 1] \qquad \textbf{\textit{Eqn. 2-1}}$$

This data format is available for bytes, words, and longs. The most negative, signed word that can be represented is –32,768 ($8000), and the most negative, signed long word is –2,147,483,648 ($80000000).

The most positive, signed word is 32,767 ($7FFF), and the most positive signed long word is 2,147,483,647 ($7FFFFFFF).

2.6.2 Unsigned Integer (UI)

The unsigned integer numbers are positive only, and they have nearly twice the magnitude of a signed number of the same size. The unsigned integer numbers lie in the following range:

$$0 \leq UI \leq [2^{[N-1]} - 1] \qquad \textbf{\textit{Eqn. 2-2}}$$

The binary word is interpreted as having a binary point immediately to the right of the integer's least significant bit. This data format is available for bytes, words, and long words. The most positive, 16-bit, unsigned integer is 65,535 ($FFFF), and the most positive, 32-bit, unsigned integer is 4,294,967,295 ($FFFFFFFF). The smallest unsigned integer number is zero ($0000), regardless of size.

2.6.3 Signed Fractional (SF)

In this format, the N-bit operand is represented using the 1.[N–1] format (one sign bit, N–1 fractional bits). The signed fractional numbers lie in the following range:

$$-1.0 \le SF \le 1.0 - 2^{-[N-1]} \qquad \textbf{Eqn. 2-3}$$

This data format is available for words and long words. For both word and long-word signed fractions, the most negative number that can be represented is –1.0; its internal representation is $8000 (word) or $80000000 (long word). The most positive word is $7FFF ($1.0 - 2^{-15}$); its most positive long word is $7FFFFFFF ($1.0 - 2^{-31}$).

2.6.4 Unsigned Fractional (UF)

The unsigned fractional numbers can be positive only, and they have nearly twice the magnitude of a signed number with the same number of bits. The unsigned fractional numbers lie in the following range:

$$0.0 \le UF \le 2.0 - 2^{-[N-1]} \qquad \textbf{Eqn. 2-4}$$

The binary word is interpreted as having a binary point after the MSB. This data format is available for words and longs. The most positive, 16-bit, unsigned number is $FFFF, or $\{1.0 + (1.0 - 2^{-[N-1]})\} = 1.99997$. The smallest unsigned fractional number is zero ($0000).

2.7 User Common Types

Table 2-1. User-Defined Typedefs in *56800E_types.h*

Mnemonics	Size — bits	Description
Word8	8	To represent 8-bit signed variable/value.
UWord8	8	To represent 16-bit unsigned variable/value.
Word16	16	To represent 16-bit signed variable/value.
UWord16	16	To represent 16-bit unsigned variable/value.
Word32	32	To represent 32-bit signed variable/value.
UWord32	32	To represent 16-bit unsigned variable/value.
Int8	8	To represent 8-bit signed variable/value.
UInt8	8	To represent 16-bit unsigned variable/value.
Int16	16	To represent 16-bit signed variable/value.
UInt16	16	To represent 16-bit unsigned variable/value.
Int32	32	To represent 32-bit signed variable/value.

Table 2-1. User-Defined Typedefs in *56800E_types.h* (continued)

UInt32	32	To represent 16-bit unsigned variable/value.
Frac16	16	To represent 16-bit signed variable/value.
Frac32	32	To represent 32-bit signed variable/value.
NULL	constant	Represents NULL pointer.
bool	16	Boolean variable.
false	constant	Represents false value.
true	constant	Represents true value.
FRAC16()	macro	Transforms float value from <–1, 1) range into fractional representation <–32768, 32767>.
FRAC32()	macro	Transforms float value from <–1, 1) range into fractional representation <–2147483648, 2147483648>.

2.8 V2 and V3 Core Support

The library has been written to support both 56800E (V2) and 56800Ex (V3) cores. The V3 core offers new set of math instructions which can simplify and accelarete the algorithm runtime. Therefore certain algorithms can have two prototypes.

If the library is used on the 56800Ex core, the V3 algorithms use is recommended because:

- the code is shorter
- the execution is faster
- the precision of 32-bit calculation is higher

The final algorithm is selected by a define. To select the correct algorithm implementation the user has to set up a define: OPTION_CORE_V3. If this define is not defined, it is automatically set up as 0. If its value is 0, the V2 algorithms are used. If its value is 1, the V3 algorithms are used.

The best way is to define this define is in the project properties (see Figure 2-1):

1. In the left hand tree, expand the C/C++ Build node
2. Click on the Settings node
3. Under the Tool Settings tab, click on the DSC Compiler/Input node
4. In the Defined Macros dialog box click on the first icon (+) and type the following: OPTION_CORE_V3=1
5. Click OK
6. Click OK on the Properties dialog box

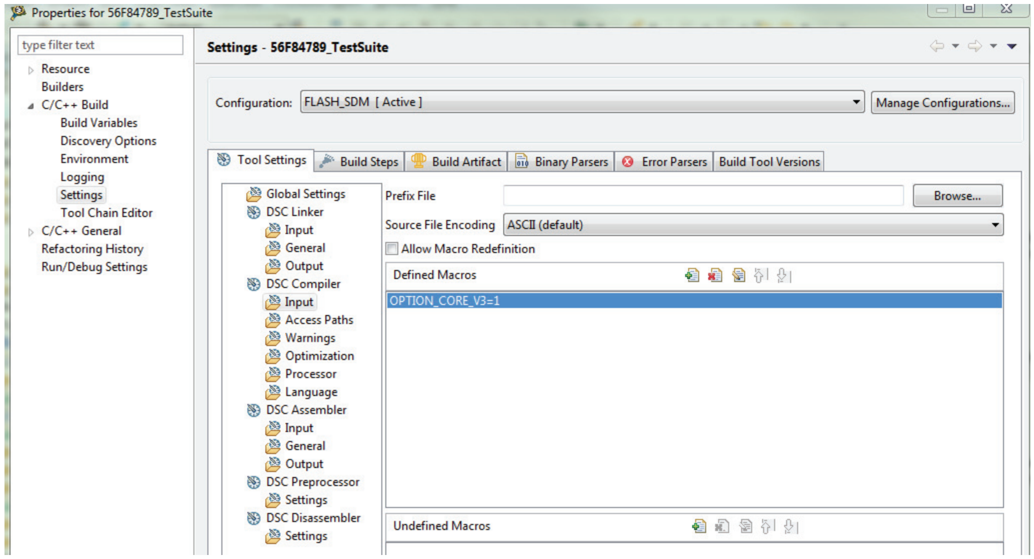

Figure 2-1. V2/V3 core option

2.9 Special Issues

All functions in the Math Library are implemented without storing any of the volatile registers (refer to the compiler manual) used by the respective routine. Only non-volatile registers (C10, D10, R5) are saved by pushing the registers on the stack. Therefore, if the particular registers initialized before the library function call are to be used after the function call, it is necessary to save them manually.

Chapter 3 FUNCTION API

3.0.1 API Summary

Table 3-1. Function API summary

Name	Arguments	Output	Description
MLIB_Abs16	Frac16 f16In	Frac16	Absolute value: $\mathrm{MLIB\_Abs16}(a) = \lvert a\rvert$
MLIB_Abs16Sat	Frac16 f16In	Frac16	Absolute value with saturation: $\mathrm{MLIB\_Abs16Sat}(a) = \lvert a\rvert$
MLIB_Abs32	Frac32 f32In	Frac32	Absolute value: $\mathrm{MLIB\_Abs32}(a) = \lvert a\rvert$
MLIB_Abs32Sat	Frac32 f32In	Frac32	Absolute value with saturation: $\mathrm{MLIB\_Abs32Sat}(a) = \lvert a\rvert$
MLIB_Rnd16	Frac16 f16In	Frac16	Round to the nearest.
MLIB_Rnd16Sat	Frac16 f16In	Frac16	Round to the nearest with saturation.
MLIB_Rnd32	Frac32 f32In	Frac32	Round upper 16 bits to the nearest.
MLIB_Rnd32Sat	Frac32 f32In	Frac32	Round upper 16 bits to the nearest with saturation.
MLIB_Neg16	Frac16 f16In	Frac16	Negative value: $\mathrm{MLIB\_Neg16}(a) = -a$
MLIB_Neg16Sat	Frac16 f16In	Frac16	Negative value with saturation: $\mathrm{MLIB\_Neg16Sat}(a) = -a$
MLIB_Neg32	Frac32 f32In	Frac32	Negative value: $\mathrm{MLIB\_Neg32}(a) = -a$
MLIB_Neg32Sat	Frac32 f32In	Frac32	Negative value with saturation: $\mathrm{MLIB\_Neg32}Sat(a) = -a$
MLIB_Add16	Frac16 f16In Frac16 f16In	Frac16	Addition: $\mathrm{MLIB\_Add16}(a, b) = a + b$
MLIB_Add16Sat	Frac16 f16In Frac16 f16In	Frac16	Addition with saturation: $\mathrm{MLIB\_Add16Sat}(a, b) = a + b$
MLIB_Add32	Frac32 f32In Frac32 f32In	Frac32	Addition: $\mathrm{MLIB\_Add32}(a, b) = a + b$
MLIB_Add32Sat	Frac32 f32In Frac32 f32In	Frac32	Addition with saturation: $\mathrm{MLIB\_Add32Sat}(a, b) = a + b$
MLIB_Sub16	Frac16 f16In Frac16 f16In	Frac16	Subtraction: $\mathrm{MLIB\_Sub16}(a, b) = a - b$
MLIB_Sub16Sat	Frac16 f16In Frac16 f16In	Frac16	Subtraction with saturation:
MLIB_Sub32	Frac32 f32In Frac32 f32In	Frac32	Subtraction: $\mathrm{MLIB\_Sub32}(a, b) = a - b$
MLIB_Sub32Sat	Frac32 f32In Frac32 f32In	Frac32	Subtraction with saturation: $\mathrm{MLIB\_Sub32Sat}(a, b) = a - b$
MLIB_Sh1L16	Frac16 f16In	Frac16	1-bit left shift: $\mathrm{MLIB\_Sh1L16}(a) = a \ll 1$
MLIB_Sh1L16Sat	Frac16 f16In	Frac16	1-bit left shift with saturation: $\mathrm{MLIB\_Sh1L16Sat}(a) = a \ll 1$

Table 3-1. Function API summary

MLIB_Sh1R16	Frac16 f16In	Frac16	1-bit right shift: $\mathrm{MLIB\_ShR16}(a, b) = a \gg b$
MLIB_Sh1L32	Frac32 f32In	Frac32	1-bit left shift: $\mathrm{MLIB\_Sh1L32}(a) = a \ll 1$
MLIB_Sh1L32Sat	Frac32 f32In	Frac32	1-bit left shift with saturation: $\mathrm{MLIB\_Sh1L32Sat}(a) = a \ll 1$
MLIB_Sh1R32	Frac32 f32In	Frac32	1-bit right shift: $\mathrm{MLIB\_Sh1R32}(a) = a \gg 1$
MLIB_ShL16	Frac16 f16In Word16 w16N	Frac16	Multi-bit left shift: $\mathrm{MLIB\_ShL16}(a, b) = a \ll b$
MLIB_ShL16Sat	Frac16 f16In Word16 w16N	Frac16	Multi-bit left shift with saturation: $\mathrm{MLIB\_ShL16Sat}(a, b) = a \ll b$
MLIB_ShR16	Frac16 f16In Word16 w16N	Frac16	Multi-bit right shift: $\mathrm{MLIB\_ShR16}(a, b) = a \gg b$
MLIB_ShR16Sat	Frac16 f16In Word16 w16N	Frac16	Multi-bit right shift with saturation: $\mathrm{MLIB\_ShR16Sat}(a, b) = a \gg b$
MLIB_ShL32	Frac32 f32In Word16 w16N	Frac32	Multi-bit left shift: $\mathrm{MLIB\_ShL32}(a, b) = a \ll b$
MLIB_ShL32Sat	Frac32 f32In Word16 w16N	Frac32	Multi-bit left shift with saturation: $\mathrm{MLIB\_ShL32Sat}(a, b) = a \ll b$
MLIB_ShR32	Frac32 f32In Word16 w16N	Frac32	Multi-bit right shift: $\mathrm{MLIB\_ShR32}(a, b) = a \gg b$
MLIB_ShR32Sat	Frac32 f32In Word16 w16N	Frac32	Multi-bit right shift with saturation: $\mathrm{MLIB\_ShR32Sat}(a, b) = a \gg b$
MLIB_Mul16SS	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication: $\mathrm{MLIB\_Mul16SS}(a, b) = (a \cdot b) \gg 16$
MLIB_Mul16SSSat	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with saturation: $\mathrm{MLIB\_Mul16SSSat}(a, b) = (a \cdot b) \gg 16$
MLIB_MulNeg16SS	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with negation: $\mathrm{MLIB\_MulNeg16SS}(a, b) = (-a \cdot b) \gg 16$
MLIB_Mul32SS	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication: $\mathrm{MLIB\_Mul32SS}(a, b) = a \cdot b$
MLIB_Mul32SSSat	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with saturation: $\mathrm{MLIB\_Mul32SSSat}(a, b) = a \cdot b$
MLIB_MulNeg32SS	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with negation: $\mathrm{MLIB\_MulNeg32SS}(a, b) = -a \cdot b$
MLIB_MulRnd16SS	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with rounding to the nearest: $\mathrm{MLIB\_MulRnd16SS}(a, b) = \mathrm{round}\left(\frac{a \cdot b}{65536}\right)$
MLIB_MulRnd16SSSat	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with rounding to the nearest with saturation: $\mathrm{MLIB\_MulRnd16SSSat}(a, b) = \mathrm{round}\left(\frac{a \cdot b}{65536}\right)$
MLIB_MulNegRnd16SS	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with negation with rounding to the nearest: $\mathrm{MLIB\_MulRnd16SS}(a, b) = \mathrm{round}\left(\frac{-a \cdot b}{65536}\right)$

Table 3-1. Function API summary

MLIB_MulNegRnd16SSSat	Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication with negation with rounding to the nearest with saturation: $\mathrm{MLIB\_MulRnd16SSSat}(a, b) = \mathrm{round}\left(\frac{-a \cdot b}{65536}\right)$
MLIB_MulRnd32SS	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with the upper 16 bits rounding to the nearest: $\mathrm{MLIB\_MulRnd32SS}(a, b) = \left[\mathrm{round}\left(\frac{a \cdot b}{65536}\right)\right] \ll 16$
MLIB_MulRnd32SSSat	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with the upper 16 bits rounding to the nearest with saturation: $\mathrm{MLIB\_MulRnd32SSSat}(a, b) = \left[\mathrm{round}\left(\frac{a \cdot b}{65536}\right)\right] \ll 16$
MLIB_MulNegRnd32SS	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with negation with the upper 16 bits rounding to the nearest: $\mathrm{MLIB\_MulNegRnd32SS}(a, b) = \left[\mathrm{round}\left(\frac{-a \cdot b}{65536}\right)\right] \ll 16$
MLIB_MulNegRnd32SSSat	Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication with negation with the upper 16 bits rounding to the nearest with saturation: $\mathrm{MLIB\_MulNegRnd32SSSat}(a, b) = \left[\mathrm{round}\left(\frac{-a \cdot b}{65536}\right)\right] \ll 16$
MLIB_Mul32LS	Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication: $\mathrm{MLIB\_Mul32LS}(a, b) = (a \cdot b) \gg 16$
MLIB_Mul32LSSat	Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication with saturation: $\mathrm{MLIB\_Mul32LSSat}(a, b) = (a \cdot b) \gg 16$
MLIB_MulNeg32LS	Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication with negation: $\mathrm{MLIB\_MulNeg32LS}(a, b) = (-a \cdot b) \gg 16$
MLIB_Mul32LL	Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication: $\mathrm{MLIB\_Mul32LL}(a, b) = (a \cdot b) \gg 32$
MLIB_Mul32LLSat	Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication with saturation: $\mathrm{MLIB\_Mul32LLSat}(a, b) = (a \cdot b) \gg 32$
MLIB_MulNeg32LL	Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication with negation: $\mathrm{MLIB\_Mul32NegLL}(a, b) = (-a \cdot b) \gg 32$
MLIB_Mac16SSS	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-accumulation: $\mathrm{MLIB\_Mac16SSS}(a, b, c) = a + [(b \cdot c) \gg 16]$
MLIB_Mac16SSSSat	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-accumulation with saturation: $\mathrm{MLIB\_Mac16SSSSat}(a, b, c) = a + [(b \cdot c) \gg 16]$
MLIB_Msu16SSS	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-subtraction: $\mathrm{MLIB\_Msu16SSS}(a, b, c) = a - [(b \cdot c) \gg 16]$
MLIB_Msu16SSSSat	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-subtraction with saturation: $\mathrm{MLIB\_Msu16SSSSat}(a, b, c) = a - [(b \cdot c) \gg 16]$

Table 3-1. Function API summary

MLIB_Mac32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation: $\mathrm{MLIB\_Mac32LSS}(a, b, c) = a + b \cdot c$
MLIB_Mac32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation with saturation: $\mathrm{MLIB\_Mac32LSSSat}(a, b, c) = a + b \cdot c$
MLIB_Msu32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction: $\mathrm{MLIB\_Msu32LSS}(a, b, c) = a - b \cdot c$
MLIB_Msu32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction with saturation: $\mathrm{MLIB\_Msu32LSSSat}(a, b, c) = a - b \cdot c$
MLIB_MacRnd16SSS	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-accumulation with rounding to the nearest: $\mathrm{MLIB\_MacRnd16SSS}(a, b, c) = \mathrm{round}\{a + [(b \cdot c) \gg 16]\}$
MLIB_MacRnd16SSSSat	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-accumulation with rounding to the nearest with saturation: $\mathrm{MLIB\_MacRnd16SSSSat}(a, b, c) = \mathrm{round}\{a + [(b \cdot c) \gg 16]\}$
MLIB_MsuRnd16SSS	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-subtraction with rounding to the nearest: $\mathrm{MLIB\_MsuRnd16SSS}(a, b, c) = \mathrm{round}\{a - [(b \cdot c) \gg 16]\}$
MLIB_MsuRnd16SSSSat	Frac16 f16In Frac16 f16In Frac16 f16In	Frac16	Fractional multiplication-subtraction with roudning to the nearest saturation: $\mathrm{MLIB\_MsuRnd16SSS}(a, b, c) = \mathrm{round}\{a - [(b \cdot c) \gg 16]\}$
MLIB_MacRnd32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest: $\mathrm{MLIB\_MacRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MacRnd32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest with saturation: $\mathrm{MLIB\_MacRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MsuRnd32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction with rounding to the nearest: $\mathrm{MLIB\_MsuRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MsuRnd32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction with roudning to the nearest saturation: $\mathrm{MLIB\_MsuRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MacRnd32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest: $\mathrm{MLIB\_MacRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16$

Table 3-1. Function API summary

MLIB_MacRnd32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest with saturation: $\mathrm{MLIB\_MacRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MsuRnd32LSS	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction with rounding to the nearest: $\mathrm{MLIB\_MsuRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16$
MLIB_MsuRnd32LSSSat	Frac32 f32In Frac16 f16In Frac16 f16In	Frac32	Fractional multiplication-subtraction with roudning to the nearest saturation: $\mathrm{MLIB\_MsuRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16$
MLIB_Mac32LLS	Frac32 f32In Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest: $\mathrm{MLIB\_Mac32LLS}(a, b, c) = a + [(b \cdot c) \gg 16]$
MLIB_Mac32LLSSat	Frac32 f32In Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication-accumulation with rounding to the nearest with saturation: $\mathrm{MLIB\_Mac32LLSSat}(a, b, c) = a + [(b \cdot c) \gg 16]$
MLIB_Msu32LLS	Frac32 f32In Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication-subtraction with rounding to the nearest: $\mathrm{MLIB\_Msu32LLS}(a, b, c) = a - [(b \cdot c) \gg 16]$
MLIB_Msu32LLSSat	Frac32 f32In Frac32 f32In Frac16 f16In	Frac32	Fractional multiplication-subtraction with roudning to the nearest saturation: $\mathrm{MLIB\_Msu32LLSSat}(a, b, c) = a - [(b \cdot c) \gg 16]$
MLIB_Mac32LLL	Frac32 f32In Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication-accumulation with rounding to the nearest: $\mathrm{MLIB\_Mac32LLL}(a, b, c) = a + [(b \cdot c) \gg 32]$
MLIB_Mac32LLLSat	Frac32 f32In Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication-accumulation with rounding to the nearest with saturation: $\mathrm{MLIB\_Mac32LLLSat}(a, b, c) = a + [(b \cdot c) \gg 32]$
MLIB_Msu32LLL	Frac32 f32In Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication-subtraction with rounding to the nearest: $\mathrm{MLIB\_Msu32LLL}(a, b, c) = a - [(b \cdot c) \gg 32]$
MLIB_Msu32LLLSat	Frac32 f32In Frac32 f32In Frac32 f32In	Frac32	Fractional multiplication-subtraction with roudning to the nearest saturation: $\mathrm{MLIB\_Msu32LLLSat}(a, b, c) = a - [(b \cdot c) \gg 32]$
MLIB_Div1Q16SS	Frac16 f16In Frac16 f16In	Frac16	Single-quadrant fractional division: $\mathrm{MLIB\_Div1Q16SS}(a, b) = \frac{a \ll 16}{b}$
MLIB_Div4Q16SS	Frac16 f16In Frac16 f16In	Frac16	Four-quadrant fractional division: $\mathrm{MLIB\_Div4Q16SS}(a, b) = \frac{a \ll 16}{b}$
MLIB_Div1Q16LS	Frac32 f32In Frac16 f16In	Frac16	Single-quadrant fractional division: $\mathrm{MLIB\_Div1Q16LS}(a, b) = \frac{a}{b}$

Table 3-1. Function API summary

MLIB_Div4Q16LS	Frac32 f32In Frac16 f16In	Frac16	Four-quadrant fractional division: $\mathrm{MLIB\_Div4Q16LS}(a, b) = \frac{a}{b}$
MLIB_Div1Q32LS	Frac32 f32In Frac16 f16In	Frac32	Single-quadrant fractional division: $\mathrm{MLIB\_Div1Q32LS}(a, b) = \frac{a}{b} \ll 16$
MLIB_Div4Q32LS	Frac32 f32In Frac16 f16In	Frac32	Four-quadrant fractional division: $\mathrm{MLIB\_Div4Q32LS}(a, b) = \frac{a}{b} \ll 16$
MLIB_Rcp161Q	Frac16 f16In	Frac32	Single-quadrant 16-bit precision reciprocal: function: $\mathrm{MLIB\_Rcp161Q}(a) = \frac{1}{a}$
MLIB_Rcp164Q	Frac16 f16In	Frac32	Four-quadrant 16-bit precision reciprocal function: $\mathrm{MLIB\_Rcp324Q}(a) = \frac{1}{a}$
MLIB_Rcp321Q	Frac16 f16In	Frac32	Single-quadrant 32-bit precision reciprocal: function: $\mathrm{MLIB\_Rcp161Q}(a) = \frac{1}{a}$
MLIB_Rcp324Q	Frac16 f16In	Frac32	Four-quadrant 32-bit precision reciprocal function: $\mathrm{MLIB\_Rcp324Q}(a) = \frac{1}{a}$

3.1 MLIB_Abs16

This function performs an absolute value of the 16-bit argument.

3.1.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Abs16(Frac16 f16In)
```

3.1.2 Prototype

```
asm inline Frac16 MLIB_Abs16FAsmi(register Frac16 f16In)
```

3.1.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-2. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.1.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.1.5 Dependencies

The dependent files are:

- MLIB_AbsAsm.h
- MLIB_types.h

3.1.6 Description

The **MLIB_Abs16** function returns the absolute value of the argument.

$$\mathrm{MLIB\_Abs16}(a) = |a| \qquad \textit{Eqn. 3-1}$$

where:

- result - Frac16
- a - Frac16

3.1.7 Returns

The function returns the 16-bit absolute value of the input f16In.

3.1.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.1.9 Special Issues

If the input is 0x8000 the output is 0x8000 if the saturation mode is turned off.

The function **MLIB_Abs16** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Abs16Sat** has to be used instead.

3.1.10 Implementation

The **MLIB_Abs16** function is implemented as an inline function.

Example 3-1. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(-0.5);

        /* Absolute value */
        mf16Out = MLIB_Abs16(mf16In);
}
```

3.1.11 See Also

See **MLIB_Abs16Sat**, **MLIB_Abs32** and **MLIB_Abs32Sat** for more information.

3.1.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-3. Performance of the MLIB_Abs16 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	5 cycles
	Max	5 cycles

3.2 MLIB_Abs16Sat

This function returns an absolute value of the 16-bit argument. The function saturates the output if necessary.

3.2.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Abs16Sat(Frac16 f16In)
```

3.2.2 Prototype

```
inline Frac16 MLIB_Abs16SatFAsmi(register Frac16 f16In)
```

3.2.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-4. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000 0x7FFF	input value

3.2.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.2.5 Dependencies

The dependent files are:

- MLIB_AbsAsm.h
- MLIB_types.h

3.2.6 Description

The **MLIB_Abs16Sat** function returns the absolute value of the argument.

$$\mathrm{MLIB\_Abs16Sat}(a) = |a| \qquad \textit{Eqn. 3-2}$$

where:

- result - Frac16

a - Frac16

3.2.7 Returns

The function returns the 16-bit absolute value of the 16-bit input f16In. The function saturates the output if necessary.

3.2.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.2.9 Special Issues

If the input is 0x8000 the output is 0x7FFF.

The function **MLIB_Abs16Sat** does not require the saturation mode to be turned on.

3.2.10 Implementation

The **MLIB_Abs16Sat** function is implemented as an inline function.

Example 3-2. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(-0.5);

        /* Absolute value */
        mf16Out = MLIB_Abs16Sat(mf16In);
}
```

3.2.11 See Also

See **MLIB_Abs16**, **MLIB_Abs32** and **MLIB_Abs32Sat** for more information.

3.2.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-5. Performance of the MLIB_Abs16Sat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.3 MLIB_Abs32

This function performs an absolute value of the 32-bit argument.

3.3.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Abs32(Frac32 f32In)
```

3.3.2 Prototype

```
inline Frac32 MLIB_Abs32FAsmi(register Frac32 f32In)
```

3.3.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-6. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.3.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.3.5 Dependencies

The dependent files are:

- MLIB_AbsAsm.h
- MLIB_types.h

3.3.6 Description

The **MLIB_Abs32** function returns the absolute value of the argument.

$$\mathrm{MLIB\_Abs32}(a) = |a| \qquad \textbf{\textit{Eqn. 3-3}}$$

where:

- result - Frac32
- a - Frac32

3.3.7 Returns

The function returns the 32-bit absolute value of the 32-bit input f32In.

3.3.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.3.9 Special Issues

If the input is 0x8000 0000 the output is 0x8000 0000 if the saturation mode is turned off.

The function **MLIB_Abs32** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Abs32Sat** has to be used instead.

3.3.10 Implementation

The **MLIB_Abs32** function is implemented as an inline function.

Example 3-3. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Absolute value */
        mf32Out = MLIB_Abs32(mf32In);
}
```

3.3.11 See Also

See **MLIB_Abs16**, **MLIB_Abs16Sat** and **MLIB_Abs32Sat** for more information.

3.3.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-7. Performance of the MLIB_Abs32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	5 cycles
	Max	5 cycles

3.4 MLIB_Abs32Sat

This function returns an absolute value of the 32-bit argument. The function saturates the output if necessary.

3.4.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Abs32Sat(Frac32 f32In)
```

3.4.2 Prototype

```
inline Frac32 MLIB_Abs32SatFAsmi(register Frac32 f32In)
```

3.4.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-8. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.4.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.4.5 Dependencies

The dependent files are:

- MLIB_AbsAsm.h
- MLIB_types.h

3.4.6 Description

The **MLIB_Abs32Sat** function returns the absolute value of the argument.

$$\mathrm{MLIB\_Abs32Sat}(a) = |a| \qquad \textbf{\textit{Eqn. 3-4}}$$

where:

- result - Frac32
- a - Frac32

3.4.7 Returns

The function returns the 32-bit absolute value of the 32-bit input f32In. The function saturates the output if necessary.

3.4.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.4.9 Special Issues

If the input is 0x8000 0000 the output is 0x7FFF FFFF.

The function **MLIB_Abs32Sat** does not require the saturation mode to be turned on.

3.4.10 Implementation

The **MLIB_Abs32Sat** function is implemented as an inline function.

Example 3-4. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Absolute value */
        mf32Out = MLIB_Abs32Sat(mf32In);
}
```

3.4.11 See Also

See **MLIB_Abs16**, **MLIB_Abs16Sat** and **MLIB_Abs32** for more information.

3.4.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-9. Performance of the MLIB_Abs32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	8 cycles
	Max	8 cycles

3.5 MLIB_Rnd16

This function performs rounds the 16-bit argument.

3.5.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Rnd16(Frac32 f32In)
```

3.5.2 Prototype

```
asm inline Frac16 MLIB_Rnd16FAsmi(register Frac32 f32In)
```

3.5.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-10. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.5.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.5.5 Dependencies

The dependent files are:

- MLIB_RoundAsm.h
- MLIB_types.h

3.5.6 Description

The **MLIB_Rnd16** function returns the rounded argument. The rounding is to the nearest.

3.5.7 Returns

The function returns the 16-bit rounded input f16In.

3.5.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.5.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the input is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 if the saturation mode is turned off.

The function **MLIB_Rnd16** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Rnd16Sat** has to be used instead.

3.5.10 Implementation

The **MLIB_Rnd16** function is implemented as an inline function.

Example 3-5. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac16 mf16Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Rounding */
        mf16Out = MLIB_Rnd16(mf32In);
}
```

3.5.11 See Also

See **MLIB_Rnd16Sat**, **MLIB_Rnd32** and **MLIB_Rnd32Sat** for more information.

3.5.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-11. Performance of the MLIB_Rnd16 Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.6 MLIB_Rnd16Sat

This function performs rounds the 16-bit argument with saturation.

3.6.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_Rnd16Sat(Frac32 f32In)
```

3.6.2 Prototype

```
asm inline Frac16 MLIB_Rnd16SatFAsmi(register Frac32 f32In)
```

3.6.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-12. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.6.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.6.5 Dependencies

The dependent files are:

- MLIB_RoundAsm.h
- MLIB_types.h

3.6.6 Description

The **MLIB_Rnd16Sat** function returns the rounded argument. The rounding is to the nearest.

3.6.7 Returns

The function returns the 16-bit rounded input f16In. The function saturates the output if necessary.

3.6.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.6.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the input is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF.

The function **MLIB_Rnd16Sat** does not require the saturation mode to be turned on.

3.6.10 Implementation

The **MLIB_Rnd16Sat** function is implemented as an inline function.

Example 3-6. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac16 mf16Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Rounding */
        mf16Out = MLIB_Rnd16Sat(mf32In);
}
```

3.6.11 See Also

See **MLIB_Rnd16**, **MLIB_Rnd32** and **MLIB_Rnd32Sat** for more information.

3.6.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-13. Performance of the MLIB_Rnd16Sat Function

Code Size (words)	2
Data Size (words)	0

Table 3-13. Performance of the MLIB_Rnd16Sat Function

Execution Clock	Min	8 cycles
	Max	8 cycles

3.7 MLIB_Rnd32

This function rounds the upper 32 bits of the argument.

3.7.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Rnd32(Frac32 f32In)
```

3.7.2 Prototype

```
asm inline Frac16 MLIB_Rnd32FAsmi(register Frac32 f32In)
```

3.7.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-14. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.7.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.7.5 Dependencies

The dependent files are:

- MLIB_RoundAsm.h
- MLIB_types.h

3.7.6 Description

The **MLIB_Rnd32** function returns the rounded argument. The rounding is to the nearest.

3.7.7 Returns

The function returns the 32-bit rounded argument (f32In) to the upper 16 bits.

3.7.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.7.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the input is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 0000 if the saturation mode is turned off.

The function **MLIB_Rnd32** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Rnd32Sat** has to be used instead.

3.7.10 Implementation

The **MLIB_Rnd32** function is implemented as an inline function.

Example 3-7. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Rounding */
        mf16Out = MLIB_Rnd32(mf32In);
}
```

3.7.11 See Also

See **MLIB_Rnd16**, **MLIB_Rnd16Sat** and **MLIB_Rnd32Sat** for more information.

3.7.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-15. Performance of the MLIB_Rnd32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	6 cycles
	Max	6 cycles

3.8 MLIB_Rnd32Sat

This function rounds the upper 32 bits of the argument with saturation.

3.8.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Rnd32Sat(Frac32 f32In)
```

3.8.2 Prototype

```
asm inline Frac16 MLIB_Rnd32SatFAsmi(register Frac32 f32In)
```

3.8.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-16. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.8.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.8.5 Dependencies

The dependent files are:

- MLIB_RoundAsm.h
- MLIB_types.h

3.8.6 Description

The **MLIB_Rnd32Sat** function returns the rounded argument. The rounding is to the nearest.

3.8.7 Returns

The function returns 32-bit argument (f32In) rounded to the upper 16 bits. The function saturates the output if necessary.

3.8.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.8.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the input is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF FFFF if the saturation mode is turned off.

The function **MLIB_Rnd32Sat** does not require the saturation mode to be turned on.

3.8.10 Implementation

The **MLIB_Rnd32Sat** function is implemented as an inline function.

Example 3-8. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Rounding */
        mf16Out = MLIB_Rnd32Sat(mf32In);
}
```

3.8.11 See Also

See **MLIB_Rnd16**, **MLIB_Rnd16Sat** and **MLIB_Rnd32** for more information.

3.8.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-17. Performance of the MLIB_Rnd32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	8 cycles
	Max	8 cycles

3.9 MLIB_Neg16

This function negates the 16-bit argument.

3.9.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Neg16(Frac16 f16In)
```

3.9.2 Prototype

```
inline Frac16 MLIB_Neg16FAsmi(register Frac16 f16In)
```

3.9.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-18. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.9.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.9.5 Dependencies

The dependent files are:

- MLIB_NegAsm.h
- MLIB_types.h

3.9.6 Description

The **MLIB_Neg16** function returns the negated argument.

$$\mathrm{MLIB\_Neg16}(a) = -a \qquad \textbf{\textit{Eqn. 3-5}}$$

where:

- result - Frac16
- a - Frac16

3.9.7 Returns

The function returns the 16-bit negative value of the 16-bit input f16In.

3.9.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.9.9 Special Issues

If the input is 0x8000 the output is 0x8000 if the saturation mode is turned off.

The function **MLIB_Neg16** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Neg16Sat** has to be used instead.

3.9.10 Implementation

The **MLIB_Neg16** function is implemented as an inline function.

Example 3-9. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(-0.5);

        /* Negative value */
        mf16Out = MLIB_Neg16(mf16In);
}
```

3.9.11 See Also

See **MLIB_Neg16Sat**, **MLIB_Neg32** and **MLIB_Neg32Sat** for more information.

3.9.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-19. Performance of the MLIB_Neg16 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	5 cycles
	Max	5 cycles

3.10 MLIB_Neg16Sat

This function negates the 16-bit argument. The function saturates the output if necessary.

3.10.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Neg16Sat(Frac16 f16In)
```

3.10.2 Prototype

```
inline Frac16 MLIB_Neg16SatFAsmi(register Frac16 f16In)
```

3.10.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-20. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.10.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.10.5 Dependencies

The dependent files are:

- MLIB_NegAsm.h
- MLIB_types.h

3.10.6 Description

The **MLIB_Neg16Sat** function returns the negated argument.

$$\mathrm{MLIB\_Neg16Sat}(a) = -a \qquad \textbf{\textit{Eqn. 3-6}}$$

where:

- result - Frac16
- a - Frac16

3.10.7 Returns

The function returns the 16-bit negative value of the 16-bit input f16In. The function saturates the output if necessary.

3.10.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.10.9 Special Issues

If the input is 0x8000 the output is 0x7FFF.

The function **MLIB_Neg16Sat** does not require the saturation mode to be turned on.

3.10.10 Implementation

The **MLIB_Neg16Sat** function is implemented as an inline function.

Example 3-10. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(-0.5);

        /* Negative value */
        mf16Out = MLIB_Neg16Sat(mf16In);
}
```

3.10.11 See Also

See **MLIB_Neg16**, **MLIB_Neg32** and **MLIB_Neg32Sat** for more information.

3.10.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-21. Performance of the MLIB_Neg16Sat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.11 MLIB_Neg32

This function negates the 32-bit argument.

3.11.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Neg32(Frac32 f32In)
```

3.11.2 Prototype

```
inline Frac32 MLIB_Neg32FAsmi(register Frac32 f32In)
```

3.11.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-22. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.11.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.11.5 Dependencies

The dependent files are:

- MLIB_NegAsm.h
- MLIB_types.h

3.11.6 Description

The **MLIB_Neg32** function returns the negated argument.

$$\mathrm{MLIB\_Neg32}(a) = -a \qquad \textit{Eqn. 3-7}$$

where:

- result - Frac32
- a - Frac32

3.11.7 Returns

The function returns the 32-bit negative value of 32-bit the input f32In.

3.11.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.11.9 Special Issues

If the input is 0x8000 0000 the output is 0x8000 0000 if the saturation mode is turned off.

The function **MLIB_Neg32** requires the saturation mode to be turned on for full-range correct operation or the **MLIB_Neg32Sat** has to be used instead.

3.11.10 Implementation

The **MLIB_Neg32** function is implemented as an inline function.

Example 3-11. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Negative value */
        mf32Out = MLIB_Neg32(mf32In);
}
```

3.11.11 See Also

See **MLIB_Neg16**, **MLIB_Neg16Sat** and **MLIB_Neg32Sat** for more information.

3.11.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-23. Performance of the MLIB_Neg32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	6 cycles
	Max	6 cycles

3.12 MLIB_Neg32Sat

This function negates the 32-bit argument. The function saturates the output if necessary.

3.12.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Neg32Sat(Frac32 f32In)
```

3.12.2 Prototype

```
inline Frac32 MLIB_Neg32SatFAsmi(register Frac32 f32In)
```

3.12.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-24. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.12.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.12.5 Dependencies

The dependent files are:

- MLIB_NegAsm.h
- MLIB_types.h

3.12.6 Description

The **MLIB_Neg32Sat** function returns the negative value of the argument.

$$\text{MLIB\_Neg32Sat}(a) = -a \qquad \textit{Eqn. 3-8}$$

where:

- result - Frac32
- a - Frac32

3.12.7 Returns

The function returns the 32-bit negated value of the 32-bit input f32In. The function saturates the output if necessary.

3.12.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.12.9 Special Issues

If the input is 0x8000 0000 the output is 0x7FFF FFFF.

The function **MLIB_Neg32Sat** does not require the saturation mode to be turned on.

3.12.10 Implementation

The **MLIB_Neg32Sat** function is implemented as an inline function.

Example 3-12. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(-0.5);

        /* Negative value */
        mf32Out = MLIB_Neg32Sat(mf32In);
}
```

3.12.11 See Also

See **MLIB_Neg16**, **MLIB_Neg16Sat** and **MLIB_Neg32** for more information.

3.12.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-25. Performance of the MLIB_Neg32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	8 cycles
	Max	8 cycles

3.13 MLIB_Add16

This function returns the 16-bit sum of two 16-bit inputs.

3.13.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Add16(Frac16 f16In1, Frac16 f16In2)
```

3.13.2 Prototype

```
inline Frac16 MLIB_Add16FAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.13.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-26. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.13.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.13.5 Dependencies

The dependent files are:

- MLIB_AddAsm.h
- MLIB_types.h

3.13.6 Description

The **MLIB_Add16** function returns the sum of two arguments. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_Add16}(a, b) = a + b \qquad \textbf{\textit{Eqn. 3-9}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.13.7 Returns

The function returns the 16-bit sum of the f16In1 and f16In2 inputs.

3.13.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.13.9 Special Issues

If the sum is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Add16** function requires the saturation mode to be turned on or the **MLIB_Add16Sat** has to be used instead.

3.13.10 Implementation

The **MLIB_Add16** function is implemented as an inline function.

Example 3-13. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16In1 + mf16In2 */
        mf16Out = MLIB_Add16(mf16In1, mf16In2);
}
```

3.13.11 See Also

See **MLIB_Add16Sat**, **MLIB_Add32** and **MLIB_Add32Sat** for more information.

3.13.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-27. Performance of the MLIB_Add16 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	8 cycles
	Max	8 cycles

3.14 MLIB_Add16Sat

This function returns the 16-bit sum of two 16-bit inputs with saturation.

3.14.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Add16Sat(Frac16 f16In1, Frac16 f16In2)
```

3.14.2 Prototype

```
inline Frac16 MLIB_Add16SatFAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.14.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-28. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.14.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.14.5 Dependencies

The dependent files are:

- MLIB_AddAsm.h
- MLIB_types.h

3.14.6 Description

The **MLIB_Add16Sat** function returns the sum of two arguments. The function saturates the output if necessary.

$$\mathrm{MLIB\_Add16Sat}(a, b) = a + b \quad \textbf{Eqn. 3-10}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.14.7 Returns

The function returns the 16-bit sum of thef16In1 and f16In2 inputs.

3.14.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.14.9 Special Issues

If the sum is greater than 0x7FFF the output is 0x7FFF. If the sum is smaller than 0x8000 the output is 0x8000.

The function **MLIB_Add16Sat** does not require the saturation mode to be turned on.

3.14.10 Implementation

The **MLIB_Add16Sat** function is implemented as an inline function.

Example 3-14. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16In1 + mf16In2 */
        mf16Out = MLIB_Add16Sat(mf16In1, mf16In2);
}
```

3.14.11 See Also

See **MLIB_Add16**, **MLIB_Add32** and **MLIB_Add32Sat** for more information.

3.14.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-29. Performance of the MLIB_Add16Sat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.15 MLIB_Add32

This function returns the 32-bit sum of two 32-bit inputs.

3.15.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Add32(Frac32 f32In1, Frac32 f32In2)
```

3.15.2 Prototype

```
inline Frac32 MLIB_Add32FAsmi(register Frac32 f32In1, register Frac32
f32In2)
```

3.15.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-30. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.15.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.15.5 Dependencies

The dependent files are:

- MLIB_AddAsm.h
- MLIB_types.h

3.15.6 Description

The **MLIB_Add32** function returns the sum of two arguments. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Add32}(a, b) = a + b \qquad \text{Eqn. 3-11}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.15.7 Returns

The function returns the 32-bit sum of the f32In1 and f32In2 inputs.

3.15.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.15.9 Special Issues

If the sum is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Add32** function requires the saturation mode to be turned on.

3.15.10 Implementation

The **MLIB_Add32** function is implemented as an inline function.

Example 3-15. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32In1 + mf32In2 */
        mf32Out = MLIB_Add32(mf32In1, mf32In2);
}
```

3.15.11 See Also

See **MLIB_Add16**, **MLIB_Add16Sat** and **MLIB_Add32Sat** for more information.

3.15.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-31. Performance of the MLIB_Add32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	9 cycles
	Max	9 cycles

3.16 MLIB_Add32Sat

This function returns the 32-bit sum of two 32-bit inputs with saturation.

3.16.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Add32Sat(Frac32 f32In1, Frac32 f32In2)
```

3.16.2 Prototype

```
inline Frac32 MLIB_Add32SatFAsmi(register Frac32 f32In1, register Frac32
f32In2)
```

3.16.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-32. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.16.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.16.5 Dependencies

The dependent files are:

- MLIB_AddAsm.h
- MLIB_types.h

3.16.6 Description

The **MLIB_Add32Sat** function returns the sum of two arguments. The function saturates the output if necessary.

$$\mathrm{MLIB\_Add32Sat}(a, b) = a + b \quad \textbf{\textit{Eqn. 3-12}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.16.7 Returns

The function returns the 32-bit sum of the f32In1 and f32In2 inputs.

3.16.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.16.9 Special Issues

If the sum is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the sum is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Add32Sat** does not require the saturation mode to be turned on.

3.16.10 Implementation

The **MLIB_Add32Sat** function is implemented as an inline function.

Example 3-16. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32In1 + mf32In2 */
        mf32Out = MLIB_Add32(mf32In1, mf32In2);
}
```

3.16.11 See Also

See **MLIB_Add16**, **MLIB_Add16Sat** and **MLIB_Add32** for more information.

3.16.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-33. Performance of the MLIB_Add32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.17 MLIB_Sub16

This function returns the 16-bit difference of two 16-bit inputs.

3.17.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Sub16(Frac16 f16In1, Frac16 f16In2)
```

3.17.2 Prototype

```
inline Frac16 MLIB_Sub16FAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.17.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-34. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.17.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.17.5 Dependencies

The dependent files are:

- MLIB_SubAsm.h
- MLIB_types.h

3.17.6 Description

The **MLIB_Sub16** function returns the difference of two arguments. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Sub16}(a, b) = a - b \qquad \text{Eqn. 3-13}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.17.7 Returns

The function returns the 16-bit difference of the f16In1 and f16In2 inputs.

3.17.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.17.9 Special Issues

If the result is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Sub16** function requires the saturation mode to be turned on or the **MLIB_Sub16Sat** has to be used instead.

3.17.10 Implementation

The **MLIB_Sub16** function is implemented as an inline function.

Example 3-17. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16In1 - mf16In2 */
        mf16Out = MLIB_Sub16(mf16In1, mf16In2);
}
```

3.17.11 See Also

See **MLIB_Sub16Sat**, **MLIB_Sub32** and **MLIB_Sub32Sat** for more information.

3.17.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-35. Performance of the MLIB_Sub16 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	9 cycles
	Max	9 cycles

3.18 MLIB_Sub16Sat

This function returns the 16-bit difference of two 16-bit inputs with saturation.

3.18.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_Sub16Sat(Frac16 f16In1, Frac16 f16In2)
```

3.18.2 Prototype

```
inline Frac16 MLIB_Sub16SatFAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.18.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-36. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.18.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.18.5 Dependencies

The dependent files are:

- MLIB_SubAsm.h
- MLIB_types.h

3.18.6 Description

The **MLIB_Sub16Sat** function returns the difference of two arguments. The function saturates the output if necessary.

$$\mathrm{MLIB\_Sub16Sat}(a, b) = a - b \qquad \textbf{\textit{Eqn. 3-14}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.18.7 Returns

The function returns the 16-bit difference of the f16In1 and f16In2 inputs.

3.18.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.18.9 Special Issues

If the result is greater than 0x7FFF the output is 0x7FFF. If the result is smaller than 0x8000 the output is 0x8000.

The **MLIB_Sub16Sat** does not require the saturation mode to be turned on.

3.18.10 Implementation

The **MLIB_Sub16Sat** function is implemented as an inline function.

Example 3-18. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
         mf16In1 = FRAC16(0.1);
         mf16In2 = FRAC16(-0.2);

         /* mf16Out = mf16In1 - mf16In2 */
         mf16Out = MLIB_Sub16Sat(mf16In1, mf16In2);
}
```

3.18.11 See Also

See **MLIB_Sub16**, **MLIB_Sub32** and **MLIB_Sub32Sat** for more information.

3.18.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-37. Performance of the MLIB_Sub16Sat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.19 MLIB_Sub32

This function returns the 32-bit difference of two 32-bit inputs.

3.19.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Sub32(Frac32 f32In1, Frac32 f32In2)
```

3.19.2 Prototype

```
inline Frac16 MLIB_Sub32FAsmi(register Frac32 f32In1, register Frac32
f32In2)
```

3.19.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-38. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.19.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.19.5 Dependencies

The dependent files are:

- MLIB_SubAsm.h
- MLIB_types.h

3.19.6 Description

The **MLIB_Sub32** function returns the difference of two arguments. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Sub32}(a, b) = a - b \qquad \textbf{\textit{Eqn. 3-15}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.19.7 Returns

The function returns the 32-bit difference of the f32In1 and f32In2 inputs.

3.19.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.19.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Sub32** function requires the saturation mode to be turned on or the **MLIB_Sub32Sat** has to be used instead.

3.19.10 Implementation

The **MLIB_Sub32** function is implemented as an inline function.

Example 3-19. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32In1 - mf32In2 */
        mf32Out = MLIB_Sub32(mf32In1, mf32In2);
}
```

3.19.11 See Also

See **MLIB_Sub16**, **MLIB_Sub16Sat** and **MLIB_Sub32Sat** for more information.

3.19.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-39. Performance of the MLIB_Sub32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	8 cycles
	Max	8 cycles

3.20 MLIB_Sub32Sat

This function returns the 32-bit difference of two 32-bit inputs with saturation.

3.20.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Sub32Sat(Frac32 f32In1, Frac32 f32In2)
```

3.20.2 Prototype

```
inline Frac16 MLIB_Sub32SatFAsmi(register Frac32 f32In1, register Frac32
f32In2)
```

3.20.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-40. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.20.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.20.5 Dependencies

The dependent files are:

- MLIB_SubAsm.h
- MLIB_types.h

3.20.6 Description

The **MLIB_Sub32Sat** function returns the difference of two arguments. The function saturates the output if necessary.

$$\mathrm{MLIB\_Sub32Sat}(a, b) = a - b \qquad \textit{Eqn. 3-16}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.20.7 Returns

The function returns the 32-bit difference of the f32In1 and f32In2 inputs.

3.20.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.20.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The **MLIB_Sub32Sat** does not require the saturation mode to be turned on..

3.20.10 Implementation

The **MLIB_Sub32Sat** function is implemented as an inline function.

Example 3-20. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32In1 - mf32In2 */
        mf32Out = MLIB_Sub32Sat(mf32In1, mf32In2);
}
```

3.20.11 See Also

See **MLIB_Sub16**, **MLIB_Sub16Sat** and **MLIB_Sub32** for more information.

3.20.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-41. Performance of the MLIB_Sub32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.21 MLIB_Sh1L16

This function performs one bit left shift of the 16-bit argument without saturation.

3.21.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Sh1L16(Frac16 f16In)
```

3.21.2 Prototype

```
inline Frac16 MLIB_Sh1L16FAsmi(register Frac16 f16In)
```

3.21.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-42. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.21.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.21.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.21.6 Description

The **MLIB_Sh1L16** function returns the value of the argument shifted one bit to the left without saturation.

$$\mathrm{MLIB\_Sh1L16}(a) = a \ll 1 \qquad \textbf{Eqn. 3-17}$$

where:

- result - Frac16
- a - Frac16

3.21.7 Returns

The function returns the 16-bit value of the f16In input shifted one bit to the left.

3.21.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.21.9 Special Issues

If the input is greater than 0x4000 or smaller than 0xC000, the result overflows.

The function **MLIB_Sh1L16** does not saturate the output. If the saturation is required the **MLIB_Sub16Sat** has to be used instead.

3.21.10 Implementation

The **MLIB_Sh1L16** function is implemented as an inline function.

Example 3-21. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(0.25); /* 0x2000 */

        /* Left shift by one bit */
        mf16Out = MLIB_Sh1L16(mf16In); /* 0x4000 */
}
```

3.21.11 See Also

See **MLIB_Sh1L16Sat**, **MLIB_Sh1R16**, **MLIB_Sh1L32**, **MLIB_Sh1L32Sat** and **MLIB_Sh1R32** for more information.

3.21.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-43. Performance of the MLIB_Sh1L16 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	5 cycles
	Max	5 cycles

3.22 MLIB_Sh1L16Sat

This function performs one bit left shift of the 16-bit argument. The function saturates the output if necessary.

3.22.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Sh1L16Sat(Frac16 f16In)
```

3.22.2 Prototype

```
inline Frac16 MLIB_Sh1L16SatFAsmi(register Frac16 f16In)
```

3.22.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-44. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.22.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.22.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.22.6 Description

The **MLIB_Sh1L16Sat** function returns the value of the argument shifted one bit to the left with saturation.

$$\mathrm{MLIB\_Sh1L16Sat}(a) = a \ll 1 \qquad \textbf{\textit{Eqn. 3-18}}$$

where:

- result - Frac16

- a - Frac16

3.22.7 Returns

The function returns the 16-bit value of the f16In input shifted one bit to the left.

3.22.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.22.9 Special Issues

If the input is greater than 0x4000, the result is 0x7FFF. If the input is smaller than 0xC000, the result is 0x8000.

The function **MLIB_Sh1L16Sat** saturates the output if necessary.

3.22.10 Implementation

The **MLIB_Sh1L16Sat** function is implemented as an inline function.

Example 3-22. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(0.25); /* 0x2000 */

        /* Left shift by one bit */
        mf16Out = MLIB_Sh1L16Sat(mf16In); /* 0x4000 */
}
```

3.22.11 See Also

See **MLIB_Sh1L16**, **MLIB_Sh1R16**, **MLIB_Sh1L32**, **MLIB_Sh1L32Sat** and **MLIB_Sh1R32** for more information.

3.22.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-45. Performance of the MLIB_Sh1L16Sat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.23 MLIB_Sh1R16

This function performs one bit right shift of the 16-bit argument.

3.23.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Sh1R16(Frac16 f16In)
```

3.23.2 Prototype

```
inline Frac16 MLIB_Sh1R16FAsmi(register Frac16 f16In)
```

3.23.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-46. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value

3.23.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.23.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.23.6 Description

The **MLIB_Sh1R16** function returns the value of the argument shifted one bit the right.

$$\mathrm{MLIB\_ShR16}(a, b) = a \gg b \qquad \textbf{\textit{Eqn. 3-19}}$$

where:

- result - Frac16
- a - Frac16

- b - Word16

3.23.7 Returns

The function returns the 16-bit value of the f16In input shifted one bit to the right.

3.23.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.23.9 Special Issues

None.

3.23.10 Implementation

The **MLIB_Sh1R16** function is implemented as an inline function.

Example 3-23. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;

void main(void)
{
        mf16In = FRAC16(0.5); /* 0x4000 */

        /* Right shift by one bit */
        mf16Out = MLIB_Sh1R16(mf16In); /* 0x2000 */
}
```

3.23.11 See Also

See **MLIB_Sh1L16**, **MLIB_Sh1L16Sat**, **MLIB_Sh1L32**, **MLIB_Sh1L32Sat** and **MLIB_Sh1R32** for more information.

3.23.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-47. Performance of the MLIB_Sh1R16 Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.24 MLIB_Sh1L32

This function performs one bit left shift of the 32-bit argument without saturation.

3.24.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Sh1L32(Frac32 f32In)
```

3.24.2 Prototype

```
inline Frac32 MLIB_Sh1L32FAsmi(register Frac32 f32In)
```

3.24.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-48. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.24.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.24.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.24.6 Description

The **MLIB_Sh1L32** function returns the value of the argument shifted one bit to the left without saturation.

$$\mathrm{MLIB\_Sh1L32}(a) = a \ll 1 \qquad \textbf{Eqn. 3-20}$$

where:

- result - Frac32
- a - Frac32

3.24.7 Returns

The function returns the 32-bit value of the f32In input shifted one bit to the left.

3.24.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.24.9 Special Issues

If the input is greater than 0x4000 0000 or smaller than 0xC000 0000, the result overflows.

The function **MLIB_Sh1L32** does not saturate output. If the saturation is required the **MLIB_Sub32Sat** has to be used instead.

3.24.10 Implementation

The **MLIB_Sh1L32** function is implemented as an inline function.

Example 3-24. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(0.25); /* 0x20000000 */

        /* Left shift by one bit */
        mf32Out = MLIB_Sh1L32(mf32In); /* 0x40000000 */
}
```

3.24.11 See Also

See **MLIB_Sh1L16**, **MLIB_Sh1L16Sat**, **MLIB_Sh1R16**, **MLIB_Sh1L32Sat** and **MLIB_Sh1R32** for more information.

3.24.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-49. Performance of the MLIB_Sh1L32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	6 cycles
	Max	6 cycles

3.25 MLIB_Sh1L32Sat

This function performs one bit left shift of the 32-bit argument with saturation.

3.25.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Sh1L32Sat(Frac32 f32In)
```

3.25.2 Prototype

```
inline Frac32 MLIB_Sh1L32SatFAsmi(register Frac32 f32In)
```

3.25.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-50. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.25.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.25.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.25.6 Description

The **MLIB_Sh1L32Sat** function returns the value of the argument shifted one bit to the left with saturation.

$$\mathrm{MLIB\_Sh1L32Sat}(a) = a \ll 1 \qquad \textbf{\textit{Eqn. 3-21}}$$

where:

- result - Frac32
- a - Frac32

3.25.7 Returns

The function returns the 32-bit value of the f32In input shifted one bit to the leftwith saturation.

3.25.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.25.9 Special Issues

If the input is greater than 0x4000 0000, the result is 0x7FFF FFFF. If the input is smaller than 0xC000 0000, the result is 0x8000 0000.

The function **MLIB_Sh1L32Sat** saturates the output if necessar

3.25.10 Implementation

The **MLIB_Sh1L32Sat** function is implemented as an inline function.

Example 3-25. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(0.25); /* 0x20000000 */

        /* Left shift by one bit */
        mf32Out = MLIB_Sh1L32Sat(mf32In); /* 0x40000000 */
}
```

3.25.11 See Also

See **MLIB_Sh1L16**, **MLIB_Sh1L16Sat**, **MLIB_Sh1R16**, **MLIB_Sh1L32** and **MLIB_Sh1R32** for more information.

3.25.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-51. Performance of the MLIB_Sh1L32Sat Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.26 MLIB_Sh1R32

This function performs one bit right shift of the 32-bit argument.

3.26.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Sh1R32(Frac32 f32In)
```

3.26.2 Prototype

```
inline Frac32 MLIB_Sh1R32FAsmi(register Frac32 f32In)
```

3.26.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-52. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.26.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.26.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.26.6 Description

The **MLIB_Sh1R32** function returns the value of the argument shifted one bit to the right.

$$\mathrm{MLIB\_Sh1R32}(a) = a \gg 1 \qquad \textbf{\textit{Eqn. 3-22}}$$

where:

- result - Frac32

a - Frac32

3.26.7 Returns

The function returns the 32-bit value of the f32In input shifted one bit to the right.

3.26.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.26.9 Special Issues

None

3.26.10 Implementation

The **MLIB_Sh1R32** function is implemented as an inline function.

Example 3-26. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;

void main(void)
{
        mf32In = FRAC32(0.5); /* 0x40000000 */

        /* Right shift by one bit */
        mf32Out = MLIB_Sh1R32(mf32In); /* 0x20000000 */
}
```

3.26.11 See Also

See **MLIB_Sh1L16**, **MLIB_Sh1L16Sat**, **MLIB_Sh1R16**, **MLIB_Sh1L32** and **MLIB_Sh1L32Sat** for more information.

3.26.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-53. Performance of the MLIB_Sh1R32 Function

Code Size (words)	1	
Data Size (words)	0	
Execution Clock	Min	7 cycles
	Max	7 cycles

3.27 MLIB_ShL16

This function performs multi-bit left shift of the 16-bit argument without saturation.

3.27.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_ShL16(Frac16 f16In, Word16 w16N)
```

3.27.2 Prototype

```
inline Frac16 MLIB_ShL16FAsmi(register Frac16 f16In, register Word16
w16N)
```

3.27.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-54. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value
w16N	In	SI16	-15...15	number of shifts to perform

3.27.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.27.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.27.6 Description

The **MLIB_ShL16** returns the argument shifted left by the specified number of bits. The function does not saturate the output. If the number of shifts is negative, the value is shifted to the right.

$$\mathrm{MLIB\_ShL16}(a, b) = a \ll b \qquad \textbf{Eqn. 3-23}$$

where:

- result - Frac16
- a - Frac16
- b - Word16

3.27.7 Returns

This function returns the f16In argument shifted to the left by the number of bits specified by the w16N argument.

3.27.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-15,15>. The output data value is in the range <-1, 1).

3.27.9 Special Issues

If the number of shifts is greater than the number of the input's leading bits, the result overflows.

The function **MLIB_ShL16** does not saturate the output. If the saturation is required the **MLIB_ShL16Sat** has to be used instead.

3.27.10 Implementation

The **MLIB_ShL16** function is implemented as an inline function.

Example 3-27. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;
static Word16 mw16N;

void main(void)
{
        mf16In = FRAC16(0.125); /* 0x1000 */
        mw16N = 2;

        /* Left shift by the mw16N bits */
        mf16Out = MLIB_ShL16(mf16In, mw16N); /* 0x4000 */
}
```

3.27.11 See Also

See **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShL32Sat**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.27.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-55. Performance of the MLIB_ShL16 Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.28 MLIB_ShL16Sat

This function performs mutli-bit left shift of the 16-bit argument with saturation.

3.28.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_ShL16Sat(Frac16 f16In, Word16 w16N)
```

3.28.2 Prototype

```
inline Frac16 MLIB_ShL16SatFAsmi(register Frac16 f16In, register Word16
w16N)
```

3.28.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-56. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value
w16N	In	SI16	-15...15	number of shifts to perform

3.28.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.28.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.28.6 Description

The **MLIB_ShL16Sat** returns the argument shifted to the left by the specified number of bits. The function saturates the output if necessary. If the number of shifts is negative, the value is shifted to the right. The function saturates the output if necessary.

$$\mathrm{MLIB\_ShL16Sat}(a, b) = a \ll b \qquad \textbf{Eqn. 3-24}$$

where:

- result - Frac16
- a - Frac16
- b - Word16

3.28.7 Returns

This function returns the f16In argument shifted to the left by the number of bits specified by the w16N argument.

3.28.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-15,15>. The output data value is in the range <-1, 1).

3.28.9 Special Issues

If the number of shifts is greater than the number of the input's leading bits, the result saturates to 0x7FFF (positive input) or to 0x8000 (negative input).

The function **MLIB_ShL16Sat** saturates the output if necessar

3.28.10 Implementation

The **MLIB_ShL16Sat** function is implemented as an inline function.

Example 3-28. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;
static Word16 mw16N;

void main(void)
{
        mf16In = FRAC16(0.125); /*0x1000*/
        mw16N = 2;

        /* Left shift by the mw16N bits */
        mf16Out = MLIB_ShL16Sat(mf16In, mw16N); /*0x4000*/
}
```

3.28.11 See Also

See **MLIB_ShL16**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShL32Sat**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.28.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-57. Performance of the MLIB_ShL16Sat Function

Code Size (words)	9	
Data Size (words)	0	
Execution Clock	Min	18 cycles
	Max	18 cycles

3.29 MLIB_ShR16

This function performs multi-bit right shift of the 16-bit argument without saturation.

3.29.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_ShR16(Frac16 f16In, Word16 w16N)
```

3.29.2 Prototype

```
inline Frac16 MLIB_ShR16FAsmi(register Frac16 f16In, register Word16
w16N)
```

3.29.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-58. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value
w16N	In	SI16	-15...15	number of shifts to perform

3.29.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.29.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.29.6 Description

The **MLIB_ShR16** returns the argument shifted to the right by the specified number of bits. The function does not saturate the output. If the number of shifts is negative, the value is shifted to the left.

$$\mathrm{MLIB\_ShR16}(a, b) = a \gg b \qquad \textit{Eqn. 3-25}$$

where:

- result - Frac16
- a - Frac16
- b - Word16

3.29.7 Returns

This function returns the f16In argument shifted to the right by the number of bits specified by the w16N argument.

3.29.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-15,15>. The output data value is in the range <-1, 1).

3.29.9 Special Issues

If the negative value of the number of shifts is greater than the number of the input's leading bits, the result overflows.

The function **MLIB_ShR16** does not saturate the output. If the saturation is required the **MLIB_ShR16Sat** has to be used instead.

3.29.10 Implementation

The **MLIB_ShR16** function is implemented as an inline function.

Example 3-29. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;
static Word16 mw16N;

void main(void)
{
         mf16In = FRAC16(0.5); /*0x4000*/
         mw16N = 2;

         /* Right shift by the mw16N bits */
         mf16Out = MLIB_ShR16(mf16In, mw16N); /*0x1000*/
}
```

3.29.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShL32Sat**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.29.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-59. Performance of the MLIB_ShR16 Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.30 MLIB_ShR16Sat

This function performs mulit-bit right shift of the 16-bit argument with saturation.

3.30.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_ShR16Sat(Frac16 f16In, Word16 w16N)
```

3.30.2 Prototype

```
inline Frac16 MLIB_ShR16SatFAsmi(register Frac16 f16In, register Word16
w16N)
```

3.30.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-60. Function Arguments

Name	In/Out	Format	Range	Description
f16In	In	SF16	0x8000... 0x7FFF	input value
w16N	In	SI16	-15...15	number of shifts to perform

3.30.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.30.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.30.6 Description

The **MLIB_ShR16Sat** returns the argument shifted to the right by the specified number of bits. The function saturates the output if necessary. If the number of shifts is negative, the value is shifted to the left.

$$\text{MLIB\_ShR16Sat}(a, b) = a \gg b \qquad \textbf{\textit{Eqn. 3-26}}$$

where:

- result - Frac16
- a - Frac16
- b - Word16

3.30.7 Returns

This function returns the 16-bit argument f16In shifted to the right by the number of bits specified by the w16N argument.

3.30.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-15,15>. The output data value is in the range <-1, 1).

3.30.9 Special Issues

If the negative value of the number of shifts is greater than the number of the input's leading bits, the result saturates to 0x7FFF (positive input) or to 0x8000 (negative input).

The function **MLIB_ShR16Sat** saturates the output if necessary.

3.30.10 Implementation

The **MLIB_ShR16Sat** function is implemented as an inline function.

Example 3-30. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In;
static Frac16 mf16Out;
static Word16 mw16N;

void main(void)
{
        mf16In = FRAC16(0.5); /*0x4000*/
        mw16N = 2;

        /* Right shift by the mw16N bits */
        mf16Out = MLIB_ShR16Sat(mf16In, mw16N); /*0x1000*/
}
```

3.30.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShL32**, **MLIB_ShL32Sat**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.30.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-61. Performance of the MLIB_ShR16Sat Function

Code Size (words)	9	
Data Size (words)	0	
Execution Clock	Min	17 cycles
	Max	17 cycles

3.31 MLIB_ShL32

This function performs multi-bit left shift of the 32-bit argument without saturation.

3.31.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_ShL32(Frac32 f32In, Word16 w16N)
```

3.31.2 Prototype

```
inline Frac32 MLIB_ShL32FAsmi(register Frac32 f32In, register Word16
w16N)
```

3.31.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-62. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
w16N	In	SI16	-31...31	number of shifts to perform

3.31.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.31.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.31.6 Description

The **MLIB_ShL32** returns the argument shifted to the left by the specified number of bits. The function does not saturate the output. If the number of shifts is negative, the value is shifted to the right.

$$\mathrm{MLIB\_ShL32}(a, b) = a \ll b \qquad \textbf{Eqn. 3-27}$$

where:

- result - Frac32
- a - Frac32
- b - Word16

3.31.7 Returns

This function returns the f32In argument shifted to the left by the number of bits specified by the w16N argument.

3.31.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-31,31>. The output data value is in the range <-1, 1).

3.31.9 Special Issues

If the number of shifts is greater than the number of the input's leading bits, the result overflows.

The function **MLIB_ShL32** does not saturate the output. If the saturation is required the **MLIB_ShL32Sat** has to be used instead.

3.31.10 Implementation

The **MLIB_ShL32** function is implemented as an inline function.

Example 3-31. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;
static Word16 mw16N;

void main(void)
{
        mf32In = FRAC32(0.125); /* 0x10000000 */
        mw16N = 2;

        /* Left shift by the mw16N bits */
        mf32Out = MLIB_ShL32(mf32In, mw16N); /* 0x40000000 */
}
```

3.31.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32Sat**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.31.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-63. Performance of the MLIB_ShL32 Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.32 MLIB_ShL32Sat

This function performs multi-bit left shift of the 32-bit argument with saturation.

3.32.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_ShL32Sat(Frac32 f32In, Word16 w16N)
```

3.32.2 Prototype

```
inline Frac32 MLIB_ShL32SatFAsmi(register Frac32 f32In, register Word16
w16N)
```

3.32.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-64. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
w16N	In	SI16	-31...31	number of shifts to perform

3.32.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.32.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.32.6 Description

The **MLIB_ShL32Sat** returns the argument shifted to the left by the specified number of bits. The function saturates the output if necessary. If the number of shifts is negative, the value is shifted to the right.

$$\text{MLIB\_ShL32Sat}(a, b) = a \ll b \quad \textit{Eqn. 3-28}$$

where:

- result - Frac32
- a - Frac32
- b - Word16

3.32.7 Returns

This function returns the f32In argument shifted to the left by the number of bits specified by the w16N argument.

3.32.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-31,31>. The output data value is in the range <-1, 1).

3.32.9 Special Issues

If the number of shifts is greater than the number of the input's leading bits, the result saturates to 0x7FFF FFFF (positive input) or to 0x8000 0000 (negative input).

The function **MLIB_ShL32Sat** saturates the output if necessary.

3.32.10 Implementation

The **MLIB_ShL32Sat** function is implemented as an inline function.

Example 3-32. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;
static Word16 mw16N;

void main(void)
{
        mf32In = FRAC32(0.125); /* 0x10000000 */
        mw16N = 2;

        /* Left shift by the mw16N bits */
        mf32Out = MLIB_ShL32Sat(mf32In, mw16N); /* 0x40000000 */
}
```

3.32.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShR32** and **MLIB_ShR32Sat** for more information.

3.32.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-65. Performance of the MLIB_ShL32Sat Function

Code Size (words)	8	
Data Size (words)	0	
Execution Clock	Min	16 cycles
	Max	16 cycles

3.33 MLIB_ShR32

This function performs multi-bit right shift of the 16-bit argument without saturation.

3.33.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_ShR32(Frac32 f32In, Word16 w16N)
```

3.33.2 Prototype

```
inline Frac32 MLIB_ShR32FAsmi(register Frac32 f32In, register Word16
w16N)
```

3.33.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-66. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
w16N	In	SI16	-31...31	number of shifts to perform

3.33.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.33.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.33.6 Description

The **MLIB_ShR32** returns the argument shifted to the right by the specified number of bits. The function does not saturate the output. If the number of shifts is negative, the value is shifted to the left.

$$\mathrm{MLIB\_ShR32}(a, b) = a \gg b \quad \textbf{\textit{Eqn. 3-29}}$$

where:

- result - Frac32
- a - Frac32
- b - Word16

3.33.7 Returns

This function returns the f32In argument shifted to the right by the number of bits specified by the w16N argument.

3.33.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-31,31>. The output data value is in the range <-1, 1).

3.33.9 Special Issues

If the negative value of the number of shifts is greater than the number of the input's leading bits, the result overflows.

The function **MLIB_ShR32** does not saturate the output. If the saturation is required the **MLIB_ShR32Sat** has to be used instead.

3.33.10 Implementation

The **MLIB_ShR32** function is implemented as an inline function.

Example 3-33. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;
static Word16 mw16N;

void main(void)
{
        mf32In = FRAC32(0.5); /* 0x40000000 */
        mw16N = 2;

        /* Right shift by the mw16N bits */
        mf32Out = MLIB_ShR32(mf32In, mw16N); /* 0x10000000 */
}
```

3.33.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShL32Sat** and **MLIB_ShR32Sat** for more information.

3.33.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-67. Performance of the MLIB_ShR32 Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.34 MLIB_ShR32Sat

This function performs multi-bit right shift of the 16-bit argument with saturation.

3.34.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_ShR32Sat(Frac32 f32In, Word16 w16N)
```

3.34.2 Prototype

```
inline Frac32 MLIB_ShR32SatFAsmi(register Frac32 f32In, register Word16
w16N)
```

3.34.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-68. Function Arguments

Name	In/Out	Format	Range	Description
f32In	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
w16N	In	SI16	-31...31	number of shifts to perform

3.34.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.34.5 Dependencies

The dependent files are:

- MLIB_ShiftAsm.h
- MLIB_types.h

3.34.6 Description

The **MLIB_ShR32Sat** returns the argument shifted to the right by the specified number of bits. The function saturates the output if necessary. If the number of shifts is negative, the value is shifted to the left.

$$\mathrm{MLIB\_ShR32Sat}(a, b) = a \gg b \qquad \textbf{Eqn. 3-30}$$

where:

- result - Frac32
- a - Frac32
- b - Word16

3.34.7 Returns

This function returns the f32In argument shifted to the right by the number of bits specified by the w16N argument. The function saturates the output if necessary. If the number of shifts is negative, the value is shifted to the left shift.

3.34.8 Range Issues

The input data value is in the range of <-1,1); the shift is in the range <-31,31>. The output data value is in the range <-1, 1).

3.34.9 Special Issues

If the negative value of the number of shifts is greater than the number of the input's leading bits, the result saturates to 0x7FFF FFFF (positive input) or to 0x8000 0000 (negative input).

The function **MLIB_ShR32Sat** saturates the output if necessary.

3.34.10 Implementation

The **MLIB_ShR32Sat** function is implemented as an inline function.

Example 3-34. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In;
static Frac32 mf32Out;
static Word16 mw16N;

void main(void)
{
        mf32In = FRAC32(0.5); /* 0x40000000 */
        mw16N = 2;

        /* Right shift by the mw16N bits */
        mf32Out = MLIB_ShR32Sat(mf32In, mw16N); /* 0x10000000 */
}
```

3.34.11 See Also

See **MLIB_ShL16**, **MLIB_ShL16Sat**, **MLIB_ShR16**, **MLIB_ShR16Sat**, **MLIB_ShL32**, **MLIB_ShL32Sat** and **MLIB_ShR32** for more information.

3.34.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-69. Performance of the MLIB_ShR32Sat Function

Code Size (words)	8	
Data Size (words)	0	
Execution Clock	Min	17 cycles
	Max	17 cycles

3.35 MLIB_Mul16SS

This function returns the 16-bit fractional product of two 16-bit fractional inputs.

3.35.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Mul16SS(Frac16 f16In1, Frac16 f16In2)
```

3.35.2 Prototype

```
inline Frac16 MLIB_Mul16SSFAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.35.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-70. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.35.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.35.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.35.6 Description

The **MLIB_Mul16SS** function returns the fractional product of two fractional inputs. The result is the upper 16 bits of the resulted 32-bit product. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mul16SS}(a, b) = (a \cdot b) \gg 16 \qquad \textbf{Eqn. 3-31}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.35.7 Returns

The function returns the 16-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2.

3.35.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.35.9 Special Issues

If the both inputs are 0x8000, the output is 0x8000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mul16SS** function requires the saturation mode to be turned on or the **MLIB_Mul16SSSat** has to be used instead.

3.35.10 Implementation

The **MLIB_Mul16SS** function is implemented as an inline function.

Example 3-35. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16In1*mf16In2 */
        mf16Out = MLIB_Mul16SS(mf16In1, mf16In2);
}
```

3.35.11 See Also

See **MLIB_Mul16SSSat**, **MLIB_MulNeg16SS**, **MLIB_Mul32SS**, **MLIB_Mul32SSSat** and **MLIB_MulNeg32SS** for more information.

3.35.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-71. Performance of the MLIB_Mul16SS Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.36 MLIB_Mul16SSSat

This function returns the 16-bit fractional product of two 16-bit fractional inputs with saturation.

3.36.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Mul16SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.36.2 Prototype

```
inline Frac16 MLIB_Mul16SSSatFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.36.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-72. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.36.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.36.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.36.6 Description

The **MLIB_Mul16SSSat** function returns the fractional product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mul16SSSat}(a, b) = (a \cdot b) \gg 16 \qquad \textbf{\textit{Eqn. 3-32}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.36.7 Returns

The function returns the 16-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2.

3.36.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.36.9 Special Issues

If the both inputs are 0x8000, the output is 0x7FFF.

The function **MLIB_Mul16SSSat** does not require the saturation mode to be turned on.

3.36.10 Implementation

The **MLIB_Mul16SSSat** function is implemented as an inline function.

Example 3-36. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16In1*mf16In2 */
        mf16Out = MLIB_Mul16SSSat(mf16In1, mf16In2);
}
```

3.36.11 See Also

See **MLIB_Mul16SS**, **MLIB_MulNeg16SS**, **MLIB_Mul32SS**, **MLIB_Mul32SSSat** and **MLIB_MulNeg32SS** for more information.

3.36.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-73. Performance of the MLIB_Mul16SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.37 MLIB_MulNeg16SS

This function returns the 16-bit negative fractional product of two 16-bit fractional inputs.

3.37.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MulNeg16SS(Frac16 f16In1, Frac16 f16In2)
```

3.37.2 Prototype

```
inline Frac16 MLIB_MulNeg16SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.37.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-74. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.37.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.37.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.37.6 Description

The **MLIB_MulNeg16SS** function returns the fractioal negative product of two fractional inputs.

$$\mathrm{MLIB\_MulNeg16SS}(a, b) = (-a \cdot b) \gg 16 \qquad \textit{Eqn. 3-33}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.37.7 Returns

The function returns the 16-bit fractional negative product two 16-bit fractional inputs f16In1 and f16In2.

3.37.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.37.9 Special Issues

The function **MLIB_MulNeg16SS** is saration mode independent.

3.37.10 Implementation

The **MLIB_MulNeg16SS** function is implemented as an inline function.

Example 3-37. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = -mf16In1*mf16In2 */
        mf16Out = MLIB_MulNeg16SS(mf16In1, mf16In2);
}
```

3.37.11 See Also

See **MLIB_Mul16SS**, **MLIB_Mul16SSSat**, **MLIB_Mul32SS**, **MLIB_Mul32SSSat** and **MLIB_MulNeg32SS** for more information.

3.37.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-75. Performance of the MLIB_MulNeg16SS Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.38 MLIB_Mul32SS

This function returns the 32-bit fractional product of two 16-bit fractional inputs.

3.38.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mul32SS(Frac16 f16In1, Frac16 f16In2)
```

3.38.2 Prototype

```
inline Frac32 MLIB_Mul32SSFAsmi(register Frac16 f16In1, register Frac16
f16In2)
```

3.38.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-76. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.38.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.38.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.38.6 Description

The **MLIB_Mul32SS** function returns the fractional product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mul32SS}(a, b) = a \cdot b \qquad \textbf{\textit{Eqn. 3-34}}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.38.7 Returns

The function returns the 32-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2.

3.38.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.38.9 Special Issues

If the both inputs are 0x8000, the output is 0x8000 0000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mul32SS** function requires the saturation mode to be turned on or the **MLIB_Mul32SSSat** has to be used instead.

3.38.10 Implementation

The **MLIB_Mul32SS** function is implemented as an inline function.

Example 3-38. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf16In1*mf16In2 */
        mf32Out = MLIB_Mul32SS(mf16In1, mf16In2);
}
```

3.38.11 See Also

See **MLIB_Mul16SS**, **MLIB_Mul16SSSat**, **MLIB_MulNeg16SS**, **MLIB_Mul32SSSat** and **MLIB_MulNeg32SS** for more information.

3.38.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-77. Performance of the MLIB_Mul32SS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.39 MLIB_Mul32SSSat

This function returns the 32-bit fractional product of two 16-bit fractional inputs with saturation.

3.39.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Mul32SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.39.2 Prototype

```
inline Frac32 MLIB_Mul32SSSatFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.39.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-78. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.39.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.39.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.39.6 Description

The **MLIB_Mul32SSSat** function returns the fractional product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mul32SSSat}(a, b) = a \cdot b \quad \textbf{Eqn. 3-35}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.39.7 Returns

The function returns the 32-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2.

3.39.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.39.9 Special Issues

If the both inputs are 0x8000, the output is 0x7FFF FFFF.

The function **MLIB_Mul32SSSat** does not require the saturation mode to be turned on.

3.39.10 Implementation

The **MLIB_Mul32SSSat** function is implemented as an inline function.

Example 3-39. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf16In1*mf16In2 */
        mf32Out = MLIB_Mul32SSSat(mf16In1, mf16In2);
}
```

3.39.11 See Also

See **MLIB_Mul16SS**, **MLIB_Mul16SSSat**, **MLIB_MulNeg16SS**, **MLIB_Mul32SS** and **MLIB_MulNeg32SS** for more information.

3.39.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-79. Performance of the MLIB_Mul32SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.40 MLIB_MulNeg32SS

This function returns the 32-bit fractional negative product of 16-bit fractional inputs.

3.40.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MulNeg32SS(Frac16 f16In1, Frac16 f16In2)
```

3.40.2 Prototype

```
inline Frac32 MLIB_MulNeg32SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.40.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-80. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.40.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.40.5 Dependencies

The dependent files are:

- MLIB_Mul16Asm.h
- MLIB_types.h

3.40.6 Description

The **MLIB_MulNeg32SS** function returns the fractional negative product of two fractional inputs.

3.40.7 Returns

The function returns the 32-bit negative value of multiple of two 16-bit fractional inputs f16In1 and f16In2.

$$\mathrm{MLIB\_MulNeg32SS}(a, b) = -a \cdot b \quad \textit{Eqn. 3-36}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.40.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.40.9 Special Issues

The function **MLIB_MulNeg32SS** is saturation mode independent.

3.40.10 Implementation

The **MLIB_MulNeg32SS** function is implemented as an inline function.

Example 3-40. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = -mf16In1*mf16In2 */
        mf32Out = MLIB_MulNeg32SS(mf16In1, mf16In2);
}
```

3.40.11 See Also

See **MLIB_Mul16SS**, **MLIB_Mul16SSSat**, **MLIB_MulNeg16SS**, **MLIB_Mul32SS** and **MLIB_MulNeg32SS** for more information.

3.40.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-81. Performance of the MLIB_MulNeg32SS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	9 cycles
	Max	9 cycles

3.41 MLIB_MulRnd16SS

This function returns the rounded 16-bit fractional product of two 16-bit fractional inputs.

3.41.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MulRnd16SS(Frac16 f16In1, Frac16 f16In2)
```

3.41.2 Prototype

```
inline Frac16 MLIB_MulRnd16SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.41.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-82. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.41.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.41.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.41.6 Description

The **MLIB_MulRnd16SS** function returns the rounded fractional product of two fractional inputs. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_MulRnd16SS}(a, b) = \mathrm{round}\left(\frac{a \cdot b}{65536}\right) \quad \textbf{\textit{Eqn. 3-37}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.41.7 Returns

The function returns the rounded 16-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2.

3.41.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.41.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 if the saturation mode is turned off.

If the both inputs are 0x8000, the output is 0x8000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MulRnd16SS** function requires the saturation mode to be turned on or the **MLIB_MulRnd16SSSat** has to be used instead.

3.41.10 Implementation

The **MLIB_MulRnd16SS** function is implemented as an inline function.

Example 3-41. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);
```

```
        /* mf16Out = mf16In1*mf16In2, result is rounded */
        mf16Out = MLIB_MulRnd16SS(mf16In1, mf16In2);
}
```

3.41.11 See Also

See **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.41.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-83. Performance of the MLIB_MulRnd16SS Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.42 MLIB_MulRnd16SSSat

This function returns the rounded 16-bit fractional product of two 16-bit fractional inputs with saturation.

3.42.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MulRnd16SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.42.2 Prototype

```
inline Frac16 MLIB_MulRnd16SSSatFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.42.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-84. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.42.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.42.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.42.6 Description

The **MLIB_MulRnd16SSSat** returns the rounded fractional product of two fractional inputs. The result is rounded to the nearest. The function saturates the output if necessary.

$$\text{MLIB\_MulRnd16SSSat}(a, b) = \text{round}\left(\frac{a \cdot b}{65536}\right) \quad \textbf{\textit{Eqn. 3-38}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.42.7 Returns

The function returns the 16-bit rounded product of two 16-bit fractional inputs f16In1 and f16In2.

3.42.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.42.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF.

If the both inputs are 0x8000, the output is 0x7FFF.

The function **MLIB_MulRnd16SSSat** does not require the saturation mode to be turned on.

3.42.10 Implementation

The **MLIB_MulRnd16SSSat** function is implemented as an inline function.

Example 3-42. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
         mf16In1 = FRAC16(0.1);
         mf16In2 = FRAC16(-0.2);

         /* mf16Out = mf16In1*mf16In2, result is rounded */
```

```
        mf16Out = MLIB_MulRnd16SSSat(mf16In1, mf16In2);
}
```

3.42.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.42.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-85. Performance of the MLIB_MulRnd16SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.43 MLIB_MulNegRnd16SS

This function returns the rounded 16-bit fractional negative product of two 16-bit fractional inputs.

3.43.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MulNegRnd16SS(Frac16 f16In1, Frac16 f16In2)
```

3.43.2 Prototype

```
inline Frac16 MLIB_MulNegRnd16SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.43.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-86. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.43.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.43.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.43.6 Description

The **MLIB_MulNegRnd16SS** function returns the rounded fractional negative product of two fractional inputs. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_MulRnd16SS}(a, b) = \text{round}\left(\frac{-a \cdot b}{65536}\right) \qquad \textbf{Eqn. 3-39}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.43.7 Returns

The function returns the rounded 16-bit fractional negative product of two 16-bit fractional inputs f16In1 and f16In2.

3.43.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.43.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MulNegRnd16SS** function requires the saturation mode to be turned on or the **MLIB_MulNegRnd16SSSat** has to be used instead.

3.43.10 Implementation

The **MLIB_MulNegRnd16SS** function is implemented as an inline function.

Example 3-43. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = -mf16In1*mf16In2, result is rounded */
        mf16Out = MLIB_MulNegRnd16SS(mf16In1, mf16In2);
```

```
}
```

3.43.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.43.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-87. Performance of the MLIB_MulNegRnd16SS Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.44 MLIB_MulNegRnd16SSSat

This function returns the rounded 16-bit fractional negative product of two 16-bit fractional inputs with saturation.

3.44.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MulNegRnd16SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.44.2 Prototype

```
inline Frac16 MLIB_MulNegRnd16SSSatFAsmi(register Frac16 f16In1,
register Frac16 f16In2)
```

3.44.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-88. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.44.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.44.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.44.6 Description

The **MLIB_MulNegRnd16SSSat** function returns the rounded fractional negative product of two fractional inputs. The result is rounded to the nearest. The function saturates the output if necessary.

$$\mathrm{MLIB\_MulRnd16SSSat}(a, b) = \mathrm{round}\left(\frac{-a \cdot b}{65536}\right) \qquad \textit{Eqn. 3-40}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.44.7 Returns

The function returns the rounded 16-bit fractional negative product of two 16-bit fractional inputs f16In1 and f16In2. The function saturates the output if necessary.

3.44.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.44.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF.

The function **MLIB_MulNegRnd16SSSat** does not require the saturation mode to be turned on.

3.44.10 Implementation

The **MLIB_MulNegRnd16SSSat** function is implemented as an inline function.

Example 3-44. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = -mf16In1*mf16In2, result is rounded */
        mf16Out = MLIB_MulNegRnd16SSSat(mf16In1, mf16In2);
```

```
}
```

3.44.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.44.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-89. Performance of the MLIB_MulNegRnd16SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.45 MLIB_MulRnd32SS

This function returns the 32-bit product of two 16-bit fractional inputs rounded to the upper 16 bits.

3.45.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MulRnd32SS(Frac16 f16In1, Frac16 f16In2)
```

3.45.2 Prototype

```
inline Frac32 MLIB_MulRnd32SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.45.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-90. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.45.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.45.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.45.6 Description

The **MLIB_MulRnd32SS** function returns the product of two fractional inputs rounded to the upper 16 bits. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_MulRnd32SS}(a, b) = \left[\text{round}\left(\frac{a \cdot b}{65536}\right)\right] \ll 16 \quad \textbf{Eqn. 3-41}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.45.7 Returns

The function returns the 32-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2 rounded to the upper 16 bits.

3.45.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.45.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 0000 if the saturation mode is turned off.

If the both inputs are 0x8000, the output is 0x8000 0000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MulRnd32SS** function requires the saturation mode to be turned on or the **MLIB_MulRnd32SSSat** has to be used instead.

3.45.10 Implementation

The **MLIB_MulRnd32SS** function is implemented as an inline function.

Example 3-45. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);
```

```
        /* mf32Out = mf16In1*mf16In2, result is rounded */
        mf32Out = MLIB_MulRnd32SS(mf16In1, mf16In2);
}
```

3.45.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SSSat**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.45.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-91. Performance of the MLIB_MulRnd32SS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.46 MLIB_MulRnd32SSSat

This function returns the 32-bit product of two 16-bit fractional inputs rounded to the upper 16 bits with saturation.

3.46.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MulRnd32SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.46.2 Prototype

```
inline Frac32 MLIB_MulRnd32SSSatFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.46.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-92. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.46.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.46.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.46.6 Description

The **MLIB_MulRnd32SSSat** function returns the product of two fractional inputs rounded to the upper 16 bits. The result is rounded to the nearest. The function saturates the output if necessary.

$$\mathrm{MLIB\_MulRnd32SSSat}(a, b) = \left[\mathrm{round}\left(\frac{a \cdot b}{65536}\right)\right] \ll 16 \qquad \textbf{Eqn. 3-42}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.46.7 Returns

The function returns the 32-bit fractional product of two 16-bit fractional inputs f16In1 and f16In2 rounded to the upper 16 bits.

3.46.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.46.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF FFFF.

If the both inputs are 0x8000, the output is 0x7FFF FFFF.

The function **MLIB_MulRnd32SSSat** does not require the saturation mode to be turned on.

3.46.10 Implementation

The **MLIB_MulRnd32SSSat** function is implemented as an inline function.

Example 3-46. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf16In1*mf16In2, result is rounded */
```

```
        mf32Out = MLIB_MulRnd32SSSat(mf16In1, mf16In2);
}
```

3.46.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulNegRnd32SS** and **MLIB_MulNegRnd32SSSat** for more information.

3.46.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-93. Performance of the MLIB_MulRnd32SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.47 MLIB_MulNegRnd32SS

This function returns the 32-bit negative product of two 16-bit fractional inputs rounded to the upper 16 bits.

3.47.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_MulNegRnd32SS(Frac16 f16In1, Frac16 f16In2)
```

3.47.2 Prototype

```
inline Frac32 MLIB_MulNegRnd32SSFAsmi(register Frac16 f16In1, register
Frac16 f16In2)
```

3.47.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-94. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.47.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.47.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.47.6 Description

The **MLIB_MulNegRnd32SS** function returns the negative product of two fractional inputs rounded to the upper 16 bits. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_MulNegRnd32SS}(a, b) = \left[\text{round}\left(\frac{-a \cdot b}{65536}\right)\right] \ll 16 \qquad \textbf{\textit{Eqn. 3-43}}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.47.7 Returns

The function returns the 32-bit fractional negative product of two 16-bit fractional inputs f16In1 and f16In2 rounded to the upper 16 bits.

3.47.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.47.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 0000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MulNegRnd32SS** function requires the saturation mode to be turned on or the **MLIB_MulNegRnd32SSSat** has to be used instead.

3.47.10 Implementation

The **MLIB_MulNegRnd32SS** function is implemented as an inline function.

Example 3-47. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = -mf16In1*mf16In2, result is rounded */
        mf32Out = MLIB_MulNegRnd32SS(mf16In1, mf16In2);
```

```
}
```

3.47.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat** and **MLIB_MulNegRnd32SSSat** for more information.

3.47.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-95. Performance of the MLIB_MulNegRnd32SS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	10 cycles
	Max	10 cycles

3.48 MLIB_MulNegRnd32SSSat

This function returns the 32-bit negative product of two 16-bit fractional inputs rounded to the upper 16 bits with saturation.

3.48.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_MulNegRnd32SSSat(Frac16 f16In1, Frac16 f16In2)
```

3.48.2 Prototype

```
inline Frac16 MLIB_MulNegRnd32SSSatFAsmi(register Frac16 f16In1,
register Frac16 f16In2)
```

3.48.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-96. Function Arguments

Name	In/Out	Format	Range	Description
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.48.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.48.5 Dependencies

The dependent files are:

- MLIB_MulRnd16Asm.h
- MLIB_types.h

3.48.6 Description

The **MLIB_MulNegRnd32SSSat** function returns the negative product of two fractional inputs rounded to the upper 16 bits. The result is rounded to the nearest. The function saturates the output if necessary.

$$\text{MLIB\_MulNegRnd32SSSat}(a, b) = \left[\text{round}\left(\frac{-a \cdot b}{65536}\right)\right] \ll 16 \qquad \textit{Eqn. 3-44}$$

where:

- result - Frac32
- a - Frac16
- b - Frac16

3.48.7 Returns

The function returns the 32-bit fractional negative product of two 16-bit fractional inputs f16In1 and f16In2 rounded to the upper 16 bits. The function saturates the output if necessary.

3.48.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.48.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the product before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x7FFF FFFF.

The function **MLIB_MulNegRnd32SSSat** does not require the saturation mode to be turned on.

3.48.10 Implementation

The **MLIB_MulNegRnd32SSSat** function is implemented as an inline function.

Example 3-48. Implementation Code

```
#include "mlib.h"

static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = -mf16In1*mf16In2, result is rounded */
        mf32Out = MLIB_MulNegRnd32SSSat(mf16In1, mf16In2);
```

```
}
```

3.48.11 See Also

See **MLIB_MulRnd16SS**, **MLIB_MulRnd16SSSat**, **MLIB_MulNegRnd16SS**, **MLIB_MulNegRnd16SSSat**, **MLIB_MulRnd32SS**, **MLIB_MulRnd32SSSat** and **MLIB_MulNegRnd32SS** for more information.

3.48.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-97. Performance of the MLIB_MulNegRnd32SSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	11 cycles
	Max	11 cycles

3.49 MLIB_Mul32LS

This function returns the 32-bit product of a 32-bit and a 16-bit fractional input.

3.49.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mul32LS(Frac32 f32In1, Frac16 f16In2)
```

3.49.2 Prototype

```
inline Frac32 MLIB_Mul32LSFAsmi(register Frac32 f32In1, register Frac16
f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mul32LSFAsmi(register Frac32 f32In1, register
Frac16 f16In2)
```

3.49.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-98. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.49.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.49.5 Dependencies

The dependent files are:

- MLIB_Mul24Asm.h
- MLIB_types.h

3.49.6 Description

The **MLIB_Mul32LS** function returns the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mul32LS}(a, b) = (a \cdot b) \gg 16 \qquad \textbf{\textit{Eqn. 3-45}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16

3.49.7 Returns

The function returns the upper 32 bits of the fractional product of a 32-bit (f32In1) and a 16-bit (f16In2) fractional input.

3.49.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.49.9 Special Issues

If the inputs are 0x8000 0000 and 0x8000, the output is 0x8000 0000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mul32LS** function requires the saturation mode to be turned on or the **MLIB_Mul32LSSat** has to be used instead.

3.49.10 Implementation

The **MLIB_Mul32LS** function is implemented as an inline function.

Example 3-49. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32In1*mf16In2 */
        mf32Out = MLIB_Mul32LS(mf32In1, mf16In2);
```

```
}
```

3.49.11 See Also

See **MLIB_Mul32LSSat**, **MLIB_MulNeg32LS**, **MLIB_Mul32LL**, **MLIB_Mul32LLSat** and **MLIB_MulNeg32LL** for more information.

3.49.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-99. Performance of the MLIB_Mul32LS Function

Code Size (words)	V2: 5, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 13, V3: 11 cycles
	Max	V2: 13, V3: 11 cycles

3.50 MLIB_Mul32LSSat

This function returns the 32-bit product of a 32-bit and a 16-bit fractional input with saturation.

3.50.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Mul32LSSat(Frac32 f32In1, Frac16 f16In2)
```

3.50.2 Prototype

```
inline Frac32 MLIB_Mul32LSSatFAsmi(register Frac32 f32In1, register
Frac16 f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mul32LSSatFAsmi(register Frac32 f32In1, register
Frac16 f16In2)
```

3.50.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-100. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.50.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.50.5 Dependencies

The dependent files are:

- MLIB_Mul24Asm.h
- MLIB_types.h

3.50.6 Description

The **MLIB_Mul32LSSat** function returns the product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mul32LSSat}(a, b) = (a \cdot b) \gg 16 \qquad \textit{Eqn. 3-46}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16

3.50.7 Returns

The function returns the upper 32 bits of the fractional product of a 32-bit (f32In1) and a 16-bit (f16In2) fractional input.

3.50.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.50.9 Special Issues

If the inputs are 0x8000 0000 and 0x8000, the output is 0x7FFF FFFF.

The function **MLIB_Mul32LSSat** does not require the saturation mode to be turned on.

3.50.10 Implementation

The **MLIB_Mul32LSSat** function is implemented as an inline function.

Example 3-50. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32In1*mf16In2 */
        mf32Out = MLIB_Mul32LSSat(mf32In1, mf16In2);
}
```

3.50.11 See Also

See **MLIB_Mul32LS**, **MLIB_MulNeg32LS**, **MLIB_Mul32LL**, **MLIB_Mul32LLSat** and **MLIB_MulNeg32LL** for more information.

3.50.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-101. Performance of the MLIB_Mul32LSSat Function

Code Size (words)	V2: 6, V3: 4	
Data Size (words)	0	
Execution Clock	Min	V2: 14, V3: 11 cycles
	Max	V2: 14, V3: 11 cycles

3.51 MLIB_MulNeg32LS

This function returns the 32-bit fractional negative product of a 32-bit and a 16-bit fractional input.

3.51.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MulNeg32LS(Frac32 f32In1, Frac16 f16In2)
```

3.51.2 Prototype

```
inline Frac32 MLIB_MulNeg32LSFAsmi(register Frac32 f32In1, register
Frac16 f16In2)

inline Frac32 MLIB_V3MulNeg32LSFAsmi(register Frac32 f32In1, register
Frac16 f16In2)
```

3.51.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-102. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.51.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.51.5 Dependencies

The dependent files are:

- MLIB_Mul24Asm.h
- MLIB_types.h

3.51.6 Description

The **MLIB_MulNeg32LS** function returns the negative product of two fractional inputs.

$$\text{MLIB\_MulNeg32LS}(a, b) = (-a \cdot b) \gg 16 \qquad \textit{Eqn. 3-47}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16

3.51.7 Returns

The function returns the upper 32 bits of the fractional product of a 32-bit (f32In1) and a 16-bit (f16In2) fractional input.

3.51.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.51.9 Special Issues

The **MLIB_MulNeg32LS** function is saturation mode independent.

3.51.10 Implementation

The **MLIB_MulNeg32LS** function is implemented as an inline function.

Example 3-51. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = -mf32In1*mf16In2 */
        mf32Out = MLIB_MulNeg32LS(mf32In1, mf16In2);
}
```

3.51.11 See Also

See **MLIB_Mul32LS**, **MLIB_Mul32LSSat**, **MLIB_Mul32LL**, **MLIB_Mul32LLSat** and **MLIB_MulNeg32LL** for more information.

3.51.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-103. Performance of the MLIB_MulNeg32LS Function

Code Size (words)	V2: 6, V3: 4	
Data Size (words)	0	
Execution Clock	Min	V2: 14, V3: 11 cycles
	Max	V2: 14, V3: 11 cycles

3.52 MLIB_Mul32LL

This function returns the 32-bit fractional product of two 32-bit fractional inputs.

3.52.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Mul32LL(Frac32 f32In1, Frac32 f32In2)
```

3.52.2 Prototype

```
inline Frac32 MLIB_Mul32LLFAsmi(register Frac32 f32In1, register Frac32
f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mul32LLFAsmi(register Frac32 f32In1, register
Frac32 f32In2)
```

3.52.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-104. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.52.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.52.5 Dependencies

The dependent files are:

- MLIB_Mul32Asm.h
- MLIB_types.h

3.52.6 Description

The **MLIB_Mul32LL** function returns the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mul32LL}(a, b) = (a \cdot b) \gg 32 \qquad \textbf{Eqn. 3-48}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.52.7 Returns

The function returns the upper 32 bits of the fractional product of two 32-bit fractional inputs f32In1 and f32In2.

3.52.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.52.9 Special Issues

If the both inputs are 0x8000 0000, the output is 0x8000 0000 if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mul32LL** function requires the saturation mode to be turned on or the **MLIB_Mul32LLSat** has to be used instead.

3.52.10 Implementation

The **MLIB_Mul32LL** function is implemented as an inline function.

Example 3-52. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
         mf32In1 = FRAC32(0.1);
         mf32In2 = FRAC32(-0.2);

         /* mf32Out = mf32In1*mf32In2 */
         mf32Out = MLIB_Mul32LL(mf32In1, mf32In2);
```

```
}
```

3.52.11 See Also

See **MLIB_Mul32LS**, **MLIB_Mul32LSSat**, **MLIB_MulNeg32LS**, **MLIB_Mul32LLSat** and **MLIB_MulNeg32LL** for more information.

3.52.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-105. Performance of the MLIB_Mul32LL Function

Code Size (words)	V2: 9, V3: 2	
Data Size (words)	0	
Execution Clock	Min	V2: 17, V3: 10 cycles
	Max	V2: 17, V3: 10 cycles

3.53 MLIB_Mul32LLSat

This function returns the 32-bit fractional product of two 32-bit fractional inputs with saturation.

3.53.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mul32LLSat(Frac32 f32In1, Frac32 f32In2)
```

3.53.2 Prototype

```
inline Frac32 MLIB_Mul32LLSatFAsmi(register Frac32 f32In1, register
Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mul32LLSatFAsmi(register Frac32 f32In1, register
Frac32 f32In2)
```

3.53.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-106. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.53.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.53.5 Dependencies

The dependent files are:

- MLIB_Mul32Asm.h
- MLIB_types.h

3.53.6 Description

The **MLIB_Mul32LLSat** function returns the product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mul32LLSat}(a, b) = (a \cdot b) \gg 32 \qquad \textbf{Eqn. 3-49}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.53.7 Returns

The function returns the upper 32 bits of the fractional product of two 32-bit fractional inputs f32In1 and f32In2.

3.53.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.53.9 Special Issues

If the both inputs are 0x8000 0000, the output is 0x7FFF FFFF.

The function **MLIB_Mul32LLSat** does not require the saturation mode to be turned on.

3.53.10 Implementation

The **MLIB_Mul32LLSat** function is implemented as an inline function.

Example 3-53. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32In1*mf32In2 */
        mf32Out = MLIB_Mul32LLSat(mf32In1, mf32In2);
}
```

3.53.11 See Also

See **MLIB_Mul32LS**, **MLIB_Mul32LSSat**, **MLIB_MulNeg32LS**, **MLIB_Mul32LL** and **MLIB_MulNeg32LL** for more information.

3.53.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-107. Performance of the MLIB_Mul32LLSat Function

Code Size (words)	V2: 10, V3:	
Data Size (words)	0	
Execution Clock	Min	V2: 18, V3: 11 cycles
	Max	V2: 18, V3: 11 cycles

3.54 MLIB_MulNeg32LL

This function returns the 32-bit fractional negative product of two 32-bit fractional inputs.

3.54.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MulNeg32LL(Frac32 f32In1, Frac32 f32In2)
```

3.54.2 Prototype

```
inline Frac32 MLIB_MulNeg32LLFAsmi(register Frac32 f32In1, register
Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3MulNeg32LLFAsmi(register Frac32 f32In1, register
Frac32 f32In2)
```

3.54.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-108. Function Arguments

Name	In/Out	Format	Range	Description
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.54.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.54.5 Dependencies

The dependent files are:

- MLIB_Mul32Asm.h
- MLIB_types.h

3.54.6 Description

The **MLIB_MulNeg32LL** function returns the negative product of two fractional inputs.

$$\mathrm{MLIB\_Mul32NegLL}(a, b) = (-a \cdot b) \gg 32 \qquad \textbf{\textit{Eqn. 3-50}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32

3.54.7 Returns

The function returns the upper 32 bits of the fractional negative product of two 32-bit fractional inputs f32In1 and f32In2.

3.54.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.54.9 Special Issues

The **MLIB_MulNeg32LL** function is saturation mode independent.

3.54.10 Implementation

The **MLIB_MulNeg32LL** function is implemented as an inline function.

Example 3-54. Implementation Code

```
#include "mlib.h"

static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = -mf32In1*mf32In2 */
        mf32Out = MLIB_MulNeg32LL(mf32In1, mf32In2);
}
```

3.54.11 See Also

See **MLIB_Mul32LS**, **MLIB_Mul32LSSat**, **MLIB_MulNeg32LS**, **MLIB_Mul32LL** and **MLIB_Mul16SSSat** for more information.

3.54.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-109. Performance of the MLIB_MulNeg32LL Function

Code Size (words)	V2: 10, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 18, V3: 11 cycles
	Max	V2: 18, V3: 11 cycles

3.55 MLIB_Mac16SSS

This function returns the 16-bit sum of the 16-bit fractional accumulator and the product of two 16-bit fractional inputs.

3.55.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Mac16SSS(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.55.2 Prototype

```
inline Frac16 MLIB_Mac16SSSFAsmi(register Frac16 f16Acc, register Frac16
f16In1, register Frac16 f16In2)
```

3.55.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-110. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.55.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.55.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.55.6 Description

The **MLIB_Mac16SSS** returns the sum of the accumulator and the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_Mac16SSS}(a, b, c) = a + [(b \cdot c) \gg 16] \quad \textbf{\textit{Eqn. 3-51}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.55.7 Returns

The function returns the 16-bit sum of a 16-bit fractional accumulator (f16Acc) and the product of two 16-bit fractional inputs (f16In1 and f16In2).

3.55.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.55.9 Special Issues

If the result is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mac16SSS** function requires the saturation mode to be turned on or the **MLIB_Mac16SSSSat** has to be used instead.

3.55.10 Implementation

The **MLIB_Mac16SSS** function is implemented as an inline function.

Example 3-55. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16Acc = FRAC16(0.3);
```

```
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc + mf16In1*mf16In2 */
        mf16Out = MLIB_Mac16SSS(mf16Acc, mf16In1, mf16In2);
}
```

3.55.11 See Also

See **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.55.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-111. Performance of the MLIB_Mac16SSS Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.56 MLIB_Mac16SSSSat

This function returns the 16-bit sum of the 16-bit fractional accumulator and the product of two 16-bit fractional inputs with saturation.

3.56.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Mac16SSSSat(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.56.2 Prototype

```
inline Frac16 MLIB_Mac16SSSSatFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.56.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-112. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.56.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.56.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.56.6 Description

The **MLIB_Mac16SSSSat** returns the sum of the accumulator and the product of two fractional inputs. The function saturates the output if necessary.

$$\text{MLIB\_Mac16SSSSat}(a, b, c) = a + [(b \cdot c) \gg 16] \qquad \textbf{Eqn. 3-52}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.56.7 Returns

The function returns the 16-bit sum of a 16-bit fractional accumulator (f16Acc) and the product of two 16-bit fractional inputs f16In1 and f16In2.

3.56.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.56.9 Special Issues

If the result is greater than 0x7FFF the output is 0x7FFF. If the result is smaller than 0x8000 the output is 0x8000.

The function **MLIB_Mac16SSSSat** does not require the saturation mode to be turned on.

3.56.10 Implementation

The **MLIB_Mac16SSSSat** function is implemented as an inline function.

Example 3-56. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16Acc = FRAC16(0.3);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc + mf16In1*mf16In2 */
        mf16Out = MLIB_Mac16SSSSat(mf16Acc, mf16In1, mf16In2);
}
```

3.56.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.56.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-113. Performance of the MLIB_Mac16SSSSat Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.57 MLIB_Msu16SSS

This function returns the 16-bit value of the product of two 16-bit fractional inputs subtracted from the 16-bit fractional accumulator.

3.57.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Msu16SSS(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.57.2 Prototype

```
inline Frac16 MLIB_Msu16SSSFAsmi(register Frac16 f16Acc, register Frac16
f16In1, register Frac16 f16In2)
```

3.57.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-114. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.57.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.57.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.57.6 Description

The **MLIB_Msu16SSS** returns the product of two fractional inputs subtracted from the fractional accumulator. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_Msu16SSS}(a, b, c) = a - [(b \cdot c) \gg 16] \qquad \textit{Eqn. 3-53}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.57.7 Returns

This function returns the 16-bit product of two 16-bit fractional inputs (f16In1 and f16In2) sustracted from the 16-bit fractional accumulator (f16Acc).

3.57.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.57.9 Special Issues

If the result is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Msu16SSS** function requires the saturation mode to be turned on or the **MLIB_Msu16SSSSat** has to be used instead.

3.57.10 Implementation

The **MLIB_Msu16SSS** function is implemented as an inline function.

Example 3-57. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc, mf16In1, mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16Acc = FRAC16(0.3);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);
```

```
        /* mf16Out = mf16Acc - mf16In1*mf16In2 */
        mf16Out = MLIB_Msu16SSS(mf16Acc, mf16In1, mf16In2);
}
```

3.57.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.57.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-115. Performance of the MLIB_Msu16SSS Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.58 MLIB_Msu16SSSSat

This function returns the 16-bit value of the product of two 16-bit fractional inputs subtracted from the 16-bit fractional accumulator with saturation.

3.58.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_Msu16SSSSat(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.58.2 Prototype

```
inline Frac16 MLIB_Msu16SSSSatFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.58.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-116. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.58.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.58.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.58.6 Description

The **MLIB_Msu16SSSSat** returns the product of two fractional inputs subtracted from the fractional accumulator. The function saturates the output if necessary.

$$\mathrm{MLIB\_Msu16SSSSat}(a, b, c) = a - [(b \cdot c) \gg 16] \qquad \textbf{\textit{Eqn. 3-54}}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.58.7 Returns

This function returns the 16-bit value of multiple of two 16-bit fractional inputs f16In1 and f16In2 subtracted from 16-bit fractional input f16Acc.

3.58.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.58.9 Special Issues

If the result is greater than 0x7FFF the output is 0x7FFF. If the result is smaller than 0x8000 the output is 0x8000.

The function **MLIB_Msu16SSSSat** does not require the saturation mode to be turned on.

3.58.10 Implementation

The **MLIB_Msu16SSSSat** function is implemented as an inline function.

Example 3-58. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16Acc = 0;
        mf16In1 = FRAC16(0.1);
```

```
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc - mf16In1*mf16In2 */
        mf16Out = MLIB_Msu16SSSSat(mf16Acc, mf16In1, mf16In2);
}
```

3.58.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.58.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-117. Performance of the MLIB_Msu16SSSSat Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.59 MLIB_Mac32LSS

This function returns the 32-bit sum of the 32-bit fractional accumulator and the product of two 16-bit fractional inputs.

3.59.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mac32LSS(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.59.2 Prototype

```
inline Frac32 MLIB_Mac32LSSFAsmi(register Frac32 f32Acc, register Frac16
f16In1, register Frac16 f16In2)
```

3.59.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-118. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.59.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.59.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.59.6 Description

The **MLIB_Mac32LSS** returns the sum of the accumulator and the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mac32LSS}(a, b, c) = a + b \cdot c \qquad \textit{Eqn. 3-55}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.59.7 Returns

The function returns the 32-bit sum of 32-bit fractional accumulator (f32Acc) and the product of two 16-bit fractional inputs f16In1 and f16In2.

3.59.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.59.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mac32LSS** function requires the saturation mode to be turned on or the **MLIB_Mac32LSSSat** has to be used instead.

3.59.10 Implementation

The **MLIB_Mac32LSS** function is implemented as an inline function.

Example 3-59. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.3);
```

```
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf16In1*mf16In2 */
        mf32Out = MLIB_Mac32LSS(mf32Acc, mf16In1, mf16In2);
}
```

3.59.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSSSat**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.59.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-119. Performance of the MLIB_Mac32LSS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.60 MLIB_Mac32LSSSat

This function returns the 32-bit sum of the 32-bit fractional accumulator and the product of two 16-bit fractional inputs.

3.60.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Mac32LSSSat(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.60.2 Prototype

```
inline Frac32 MLIB_Mac32LSSSatFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.60.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-120. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.60.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.60.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.60.6 Description

The **MLIB_Mac32LSSSat** eturns the sum of the accumulator and the product of two fractional inputs. The function saturates the output if necessary.

$$\text{MLIB\_Mac32LSSSat}(a, b, c) = a + b \cdot c \qquad \textbf{Eqn. 3-56}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.60.7 Returns

The function returns the 32-bit sum of 32-bit fractional accumulator (f32Acc) and the product of two 16-bit fractional inputs f16In1 and f16In2.

3.60.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.60.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Mac32LSSSat** does not require the saturation mode to be turned on.

3.60.10 Implementation

The **MLIB_Mac32LSSSat** function is implemented as an inline function.

Example 3-60. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.3);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf16In1*mf16In2 */
        mf32Out = MLIB_Mac32LSSSat(mf32Acc, mf16In1, mf16In2);
}
```

3.60.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Msu32LSS** and **MLIB_Msu32LSSSat** for more information.

3.60.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-121. Performance of the MLIB_Mac32LSSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	13 cycles
	Max	13 cycles

3.61 MLIB_Msu32LSS

This function returns the 32-bit product of two 16-bit fractional inputs subtracted from the 32-bit fractional accumulator.

3.61.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Msu32LSS(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.61.2 Prototype

```
inline Frac32 MLIB_Msu32LSSFAsmi(register Frac32 f32Acc, register Frac16
f16In1, register Frac16 f16In2)
```

3.61.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-122. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.61.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.61.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.61.6 Description

The **MLIB_Msu32LSS** returns the product of two fractional inputs subtracted from the fractional accumulator. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Msu32LSS}(a, b, c) = a - b \cdot c \qquad \textbf{Eqn. 3-57}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.61.7 Returns

This function returns the 32-bit product of two 16-bit fractional inputs (f16In1 and f16In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.61.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.61.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Msu32LSS** function requires the saturation mode to be turned on or the **MLIB_Msu32LSSSat** has to be used instead.

3.61.10 Implementation

The **MLIB_Msu32LSS** function is implemented as an inline function.

Example 3-61. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1, mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.4);
        mf16In1 = FRAC16(0.1);
```

```
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf16In1*mf16In2 */
        mf32Out = MLIB_Msu32LSS(mf32Acc, mf16In1, mf16In2);
}
```

3.61.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat** and **MLIB_Msu32LSSSat** for more information.

3.61.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-123. Performance of the MLIB_Msu32LSS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.62 MLIB_Msu32LSSSat

This function returns the 32-bit product of two 16-bit fractional inputs subtracted from the 32-bit fractional accumulator with saturation.

3.62.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Msu32LSSSat(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.62.2 Prototype

```
inline Frac32 MLIB_Msu32LSSSatFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.62.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-124. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.62.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.62.5 Dependencies

The dependent files are:

- MLIB_Mac16Asm.h
- MLIB_types.h

3.62.6 Description

The **MLIB_Msu32LSSSat** returns the product of two fractional inputs subtracted from the fractional accumulator. The function saturates the output if necessary.

$$\mathrm{MLIB\_Msu32LSSSat}(a, b, c) = a - b \cdot c \qquad \textbf{Eqn. 3-58}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.62.7 Returns

This function returns the 32-bit product of two 16-bit fractional inputs (f16In1 and f16In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.62.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.62.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Msu32LSSSat** does not require the saturation mode to be turned on.

3.62.10 Implementation

The **MLIB_Msu32LSSSat** function is implemented as an inline function.

Example 3-62. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.4);
        mf16In1 = FRAC16(0.1);
```

```
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf16In1*mf16In2 */
        mf32Out = MLIB_Msu32LSSSat(mf32Acc, mf16In1, mf16In2);
}
```

3.62.11 See Also

See **MLIB_Mac16SSS**, **MLIB_Mac16SSSSat**, **MLIB_Msu16SSS**, **MLIB_Msu16SSSSat**, **MLIB_Mac32LSS**, **MLIB_Mac32LSSSat** and **MLIB_Msu32LSS** for more information.

3.62.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-125. Performance of the MLIB_Msu32LSSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	13 cycles
	Max	13 cycles

3.63 MLIB_MacRnd16SSS

This function returns the rounded 16-bit sum of the 16-bit fractional accumulator and the product of two 16-bit fractional inputs.

3.63.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_MacRnd16SSS(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.63.2 Prototype

```
inline Frac16 MLIB_MacRnd16SSSFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.63.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-126. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.63.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.63.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.63.6 Description

The **MLIB_MacRnd16SSS** returns the rounded sum of the accumulator and the product of two fractional inputs. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_MacRnd16SSS}(a, b, c) = \mathrm{round}\{a + [(b \cdot c) \gg 16]\} \qquad \textit{Eqn. 3-59}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.63.7 Returns

The function returns the 16-bit rounded sum of a 16-bit fractional accumulator (f16Acc) and the product of two 16-bit fractional inputs (f16In1 and f16In2).

3.63.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.63.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the result before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 if the saturation mode is turned off.

If the result is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MacRnd16SSS** function requires the saturation mode to be turned on or the **MLIB_MacRnd16SSSat** has to be used instead.

3.63.10 Implementation

The **MLIB_MacRnd16SSS** function is implemented as an inline function.

Example 3-63. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc;
```

```
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;

void main(void)
{
        mf16Acc = FRAC16(0.5);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc + mf16In1*mf16In2, result is rounded*/
        mf16Out = MLIB_MacRnd16SSS(mf16Acc, mf16In1, mf16In2);
}
```

3.63.11 See Also

See **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.63.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-127. Performance of the MLIB_MacRnd16SSS Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.64 MLIB_MacRnd16SSSSat

This function returns the rounded 16-bit sum of the 16-bit fractional accumulator and the product of two 16-bit fractional inputs with saturation.

3.64.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_MacRnd16SSSSat(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.64.2 Prototype

```
inline Frac16 MLIB_MacRnd16SSSSatFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.64.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-128. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.64.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.64.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.64.6 Description

The **MLIB_MacRnd16SSSSat** function returns the rounded fractional product of two fractional inputs. The result is rounded to the nearest. The function saturates the output if necessary.

$$\mathrm{MLIB\_MacRnd16SSSSat}(a, b, c) = \mathrm{round}\{a + [(b \cdot c) \gg 16]\} \qquad \textit{Eqn. 3-60}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.64.7 Returns

The function returns the 16-bit rounded sum of a 16-bit fractional accumulator (f16Acc) and the product of two 16-bit fractional inputs (f16In1 and f16In2).

3.64.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.64.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the result is greater than 0x7FFF the output is 0x7FFF. If the result is smaller than 0x8000 the output is 0x8000.

The function **MLIB_MacRnd16SSSSat** does not require the saturation mode to be turned on.

3.64.10 Implementation

The **MLIB_MacRnd16SSSSat** function is implemented as an inline function.

Example 3-64. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac16 mf16Out;
```

```
void main(void)
{
        mf16Acc = FRAC16(0.4);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc + mf16In1*mf16In2, result is rounded*/
        mf16Out = MLIB_MacRnd16SSSSat(mf16Acc, mf16In1, mf16In2);
}
```

3.64.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.64.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-129. Performance of the MLIB_MacRnd16SSSSat Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.65 MLIB_MsuRnd16SSS

This function returns the rounded 16-bit fractional product of two 16-bit fractional inputs subtracted from the 16-bit fractional accumulator.

3.65.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_MsuRnd16SSS(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.65.2 Prototype

```
inline Frac16 MLIB_MsuRnd16SSSFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.65.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-130. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.65.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.65.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.65.6 Description

The **MLIB_MsuRnd16SSS** returns the rounded product of two fractional inputs subtracted from the fractional accumulator. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\text{MLIB\_MsuRnd16SSS}(a, b, c) = \text{round}\{a - [(b \cdot c) \gg 16]\} \qquad \textbf{Eqn. 3-61}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.65.7 Returns

This function returns the rounded 16-bit product 16-bit fractional inputs (f16In1 and f16In2) subtracted from the 16-bit fractional accumulator (f16Acc).

3.65.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.65.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the result before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 if the saturation mode is turned off.

If the result is greater than 0x7FFF or smaller than 0x8000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MsuRnd16SSS** function requires the saturation mode to be turned on or the **MLIB_MsuRnd16SSSat** has to be used instead.

3.65.10 Implementation

The **MLIB_MsuRnd16SSS** function is implemented as an inline function.

Example 3-65. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
```

```
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf16Acc = FRAC16(0.2);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc - mf16In1*mf16In2, result is rounded */
        mf16Out = MLIB_MsuRnd16SSS(mf16Acc, mf16In1, mf16In2);
}
```

3.65.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.65.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-131. Performance of the MLIB_MsuRnd16SSS Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.66 MLIB_MsuRnd16SSSSat

This function returns the rounded 16-bit fractional product of two 16-bit fractional inputs subtracted from the 16-bit fractional accumulator with saturation.

3.66.1 Synopsis

```
#include ”mlib.h”
Frac16 MLIB_MsuRnd16SSSSat(Frac16 f16Acc, Frac16 f16In1, Frac16 f16In2)
```

3.66.2 Prototype

```
inline Frac16 MLIB_MsuRnd16SSSSatFAsmi(register Frac16 f16Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.66.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-132. Function Arguments

Name	In/Out	Format	Range	Description
f16Acc	In	SF16	0x8000... 0x7FFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.66.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.66.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.66.6 Description

The **MLIB_MsuRnd16SSSSat** returns the rounded product of two fractional inputs subtracted from the fractional accumulator. The result is rounded to the nearest. The function saturates the output if necessary.

$$\text{MLIB\_MsuRnd16SSS}(a, b, c) = \text{round}\{a - [(b \cdot c) \gg 16]\} \qquad \textit{Eqn. 3-62}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16
- c - Frac16

3.66.7 Returns

This function returns the rounded 16-bit product 16-bit fractional inputs (f16In1 and f16In2) subtracted from the 16-bit fractional accumulator (f16Acc).

3.66.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.66.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the result is greater than 0x7FFF the output is 0x7FFF. If the result is smaller than 0x8000 the output is 0x8000.

The function **MLIB_MsuRnd16SSSSat** does not require the saturation mode to be turned on.

3.66.10 Implementation

The **MLIB_MsuRnd16SSSSat** function is implemented as an inline function.

Example 3-66. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;
```

```
void main(void)
{
        mf16Acc = FRAC16(0.1);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf16Out = mf16Acc - mf16In1*mf16In2, result is rounded */
        mf16Out = MLIB_MsuRnd16SSSSat(mf16Acc, mf16In1, mf16In2);
}
```

3.66.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.66.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-133. Performance of the MLIB_MsuRnd16SSSSat Function

Code Size (words)	4	
Data Size (words)	0	
Execution Clock	Min	14 cycles
	Max	14 cycles

3.67 MLIB_MacRnd32LSS

This function returns the 32-bit sum rounded to the upper 16 bits of the 32-bit fractional accumulator and the product of two 16-bit fractional inputs.

3.67.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_MacRnd32LSS(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.67.2 Prototype

```
inline Frac32 MLIB_MacRnd32LSSFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.67.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-134. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.67.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.67.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.67.6 Description

The **MLIB_MacRnd32LSS** returns the rounded sum of the accumulator and the product of two fractional inputs. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_MacRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16 \qquad \textbf{\textit{Eqn. 3-63}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.67.7 Returns

The function returns the 32-bit sum rounded to the upper 16 bits of the 32-bit fractional accumulator (f32Acc) and the product of two 16-bit fractional inputs (f16In1 and f16In2). The function does not saturate the output if the saturation mode is turned off.

3.67.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.67.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two’s compelment.)

If the result before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 0000 if the saturation mode is turned off.

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MacRnd32LSS** function requires the saturation mode to be turned on or the **MLIB_MacRnd32LSSSat** has to be used instead.

3.67.10 Implementation

The **MLIB_MacRnd32LSS** function is implemented as an inline function.

Example 3-67. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.6);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf16In1*mf16In2, result is rounded*/
        mf32Out = MLIB_MacRnd32LSS(mf32Acc, mf16In1, mf16In2);
}
```

3.67.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSSSat**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.67.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-135. Performance of the MLIB_MacRnd32LSS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.68 MLIB_MacRnd32LSSSat

This function returns the 32-bit sum rounded to the upper 16 bits of the 32-bit fractional accumulator and the product of two 16-bit fractional inputs with saturation.

3.68.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_MacRnd32LSSSat(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.68.2 Prototype

```
inline Frac32 MLIB_MacRnd32LSSSatFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.68.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-136. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.68.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.68.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.68.6 Description

The **MLIB_MacRnd32LSSSat** returns the rounded sum of the accumulator and the product of two fractional inputs. The result is rounded to the nearest. The function saturates the output if necessary.

$$\mathrm{MLIB\_MacRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a + b \cdot c}{65536}\right)\right] \ll 16 \qquad \textbf{\textit{Eqn. 3-64}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.68.7 Returns

The function returns the 32-bit sum rounded to the upper 16 bits of the 32-bit fractional accumulator (f32Acc) and the product of two 16-bit fractional inputs (f16In1 and f16In2).

3.68.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.68.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the result before roudning is greater than 0x7FFF 8000 the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_MacRnd32LSSSat** does not require the saturation mode to be turned on.

3.68.10 Implementation

The **MLIB_MacRnd32LSSSat** function is implemented as an inline function.

Example 3-68. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
```

```
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.6);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf16In1*mf16In2, result is rounded*/
        mf32Out = MLIB_MacRnd32LSSSat(mf32Acc, mf16In1, mf16In2);
}
```

3.68.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MsuRnd32LSS** and **MLIB_MsuRnd32LSSSat** for more information.

3.68.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-137. Performance of the MLIB_MacRnd32LSSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	13 cycles
	Max	13 cycles

3.69 MLIB_MsuRnd32LSS

This function returns the rounded 32-bit value rounded to the upper 16 bits of the product of two 16-bit fractional inputs subtracted from the 32-bit fractional accumulator.

3.69.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_MsuRnd32LSS(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.69.2 Prototype

```
inline Frac32 MLIB_MsuRnd32LSSFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.69.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-138. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.69.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.69.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.69.6 Description

The **MLIB_MsuRnd32LSS** returns the rounded product of two fractional inputs subtracted from the fractional accumulator. The result is rounded to the nearest. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_MsuRnd32LSS}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16 \qquad \textbf{\textit{Eqn. 3-65}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.69.7 Returns

This function returns the 32-bit value rounded to the upper 16 bits of the product of two 16-bit fractional inputs (f16In1 and f16In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.69.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.69.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the result before rounding is 0x7FFF 8000 to 0x7FFF FFF the output is 0x8000 0000 if the saturation mode is turned off.

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_MsuRnd32LSS** function requires the saturation mode to be turned on or the **MLIB_MsuRnd32LSSSat** has to be used instead.

3.69.10 Implementation

The **MLIB_MsuRnd32LSS** function is implemented as an inline function.

Example 3-69. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC16(0.7);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf16In1*mf16In2, result is rounded */
        mf32Out = MLIB_MsuRnd32LSS(mf32Acc, mf16In1, mf16In2);
}
```

3.69.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat** and **MLIB_MsuRnd32LSSSat** for more information.

3.69.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-139. Performance of the MLIB_MsuRnd32LSS Function

Code Size (words)	2	
Data Size (words)	0	
Execution Clock	Min	12 cycles
	Max	12 cycles

3.70 MLIB_MsuRnd32LSSSat

This function returns the rounded 32-bit value rounded to the upper 16 bits of the product of two 16-bit fractional inputs subtracted from the 32-bit fractional accumulator with saturation.

3.70.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_MsuRnd32LSSSat(Frac32 f32Acc, Frac16 f16In1, Frac16 f16In2)
```

3.70.2 Prototype

```
inline Frac32 MLIB_MsuRnd32LSSSatFAsmi(register Frac32 f32Acc, register
Frac16 f16In1, register Frac16 f16In2)
```

3.70.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-140. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f16In1	In	SF16	0x8000... 0x7FFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.70.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.70.5 Dependencies

The dependent files are:

- MLIB_MacRnd16Asm.h
- MLIB_types.h

3.70.6 Description

The **MLIB_MsuRnd32LSSSat** returns the rounded value of multiple of two fractional inputs substracted from the fractional value of accumulator. The result is rounded to the nearest. The function saturates the output if necessary.

$$\mathrm{MLIB\_MsuRnd32LSSSat}(a, b, c) = \left[\mathrm{round}\left(\frac{a - b \cdot c}{65536}\right)\right] \ll 16 \qquad \textit{Eqn. 3-66}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16
- c - Frac16

3.70.7 Returns

This function returns the 32-bit value rounded to the upper 16 bits of the product of two 16-bit fractional inputs (f16In1 and f16In2) substracted from the 32-bit fractional accumulator (f32Acc).

3.70.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.70.9 Special Issues

The rounding of the result where the lower 16 bits are equalt to exactly 0.5 (0x8000) depends on the rounding mode of the core (convergent or two's compelment.)

If the result before roudning is greater than 0x7FFF 8000 the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_MsuRnd32LSSSat** does not require the saturation mode to be turned on.

3.70.10 Implementation

The **MLIB_MsuRnd32LSSSat** function is implemented as an inline function.

Example 3-70. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
```

```
static Frac16 mf16In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC16(0.3);
        mf16In1 = FRAC16(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf16In1*mf16In2, result is rounded */
        mf32Out = MLIB_MsuRnd32LSSSat(mf32Acc, mf16In1, mf16In2);
}
```

3.70.11 See Also

See **MLIB_MacRnd16SSS**, **MLIB_MacRnd16SSSSat**, **MLIB_MsuRnd16SSS**, **MLIB_MsuRnd16SSSSat**, **MLIB_MacRnd32LSS**, **MLIB_MacRnd32LSSSat** and **MLIB_MsuRnd32LSS** for more information.

3.70.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-141. Performance of the MLIB_MsuRnd32LSSSat Function

Code Size (words)	3	
Data Size (words)	0	
Execution Clock	Min	13 cycles
	Max	13 cycles

3.71 MLIB_Mac32LLS

This function returns the 32-bit sum of the 32-bit fractional accumulator and the upper 32 bits of the product of a 32-bit and a 16-bit fractional inputs.

3.71.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mac32LLS(Frac32 f32Acc, Frac32 f32In1, Frac16 f16In2)
```

3.71.2 Prototype

```
inline Frac32 MLIB_Mac32LLSFAsmi(register Frac32 f32Acc, register Frac32
f32In1, register Frac16 f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mac32LLSFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

3.71.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-142. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.71.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.71.5 Dependencies

The dependent files are:

- MLIB_Mac24Asm.h
- MLIB_types.h

3.71.6 Description

The **MLIB_Mac32LLS** returns the sum of the accumulator and the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mac32LLS}(a, b, c) = a + [(b \cdot c) \gg 16] \qquad \textbf{\textit{Eqn. 3-67}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac16

3.71.7 Returns

The function returns the 32-bit sum of the 32-bit fractional accumulator (f32Acc) and the upper 32 bits of the product of a 32-bit fractional input (f32In1) and 16-bit fractional input (f16In2).

3.71.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.71.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mac32LLS** function requires the saturation mode to be turned on or the **MLIB_Mac32LLSSat** has to be used instead.

3.71.10 Implementation

The **MLIB_Mac32LLS** function is implemented as an inline function.

Example 3-71. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
```

```
        mf32Acc = FRAC32(0.7);
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf32In1*mf16In2 */
        mf32Out = MLIB_Mac32LLS(mf32Acc, mf32In1, mf16In2);
}
```

3.71.11 See Also

See **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.71.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-143. Performance of the MLIB_Mac32LLS Function

Code Size (words)	V2: 6, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 16, V3: 12 cycles
	Max	V2: 16, V3: 12 cycles

3.72 MLIB_Mac32LLSSat

This function returns the 32-bit sum of the 32-bit fractional accumulator and the upper 32 bits of the product of a 32-bit and a 16-bit fractional inputs with saturation.

3.72.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mac32LLSSat(Frac32 f32Acc, Frac32 f32In1, Frac16 f16In2)
```

3.72.2 Prototype

```
inline Frac32 MLIB_Mac32LLSSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mac32LLSSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

3.72.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-144. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.72.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.72.5 Dependencies

The dependent files are:

- MLIB_Mac24Asm.h
- MLIB_types.h

3.72.6 Description

The **MLIB_Mac32LLSSat** returns the sum of the accumulator and the product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mac32LLSSat}(a, b, c) = a + [(b \cdot c) \gg 16] \qquad \textbf{\textit{Eqn. 3-68}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac16

3.72.7 Returns

The function returns the 32-bit sum of the 32-bit fractional accumulator (f32Acc) and the upper 32 bits of the product of a 32-bit fractional input (f32In1) and 16-bit fractional input (f16In2).

3.72.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.72.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Mac32LLSSat** does not require the saturation mode to be turned on.

3.72.10 Implementation

The **MLIB_Mac32LLSSat** function is implemented as an inline function.

Example 3-72. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.7);
        mf32In1 = FRAC32(0.1);
```

```
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc + mf32In1*mf16In2 */
        mf32Out = MLIB_Mac32LLSSat(mf32Acc, mf32In1, mf16In2);
}
```

3.72.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.72.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-145. Performance of the MLIB_Mac32LLSSat Function

Code Size (words)	V2: 7, V3: 4	
Data Size (words)	0	
Execution Clock	Min	V2: 17, V3: 14 cycles
	Max	V2: 17, V3: 14 cycles

3.73 MLIB_Msu32LLS

This function returns the 32-bit value of the upper 32 bits of the product of a 32-bit and 16-bit fractional input subtracted from the 32-bit fractional accumulator.

3.73.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Msu32LLS(Frac32 f32Acc, Frac32 f32In1, Frac16 f16In2)
```

3.73.2 Prototype

```
inline Frac32 MLIB_Msu32LLSFAsmi(register Frac32 f32Acc, register Frac32
f32In1, register Frac16 f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mac32LLSSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

3.73.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-146. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.73.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.73.5 Dependencies

The dependent files are:

- MLIB_Mac24Asm.h
- MLIB_types.h

3.73.6 Description

The **MLIB_Msu32LLS** returns the product of two fractional inputs subtracted from the fractional accumulator. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Msu32LLS}(a, b, c) = a - [(b \cdot c) \gg 16] \qquad \textbf{\textit{Eqn. 3-69}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac16

3.73.7 Returns

This function returns the 32-bit value of the upper 32 bits of the product of a 32-bit fractional input (f32In1) and a 16-bit fractional input (f16In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.73.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.73.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Msu32LLS** function requires the saturation mode to be turned on or the **MLIB_Msu32LLSSat** has to be used instead.

3.73.10 Implementation

The **MLIB_Msu32LLS** function is implemented as an inline function.

Example 3-73. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
```

```
        mf32Acc = FRAC32(0.8);
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf32In1*mf16In2 */
        mf32Out = MLIB_Msu32LLS(mf32Acc, mf32In1, mf16In2);
}
```

3.73.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.73.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-147. Performance of the MLIB_Msu32LLS Function

Code Size (words)	V2: 6, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 16, V3: 12 cycles
	Max	V2: 16, V3: 12 cycles

3.74 MLIB_Msu32LLSSat

This function returns the 32-bit value of the upper 32 bits of the product of a 32-bit and 16-bit fractional input subtracted from the 32-bit fractional accumulator with saturation

3.74.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Msu32LLSSat(Frac32 f32Acc, Frac32 f32In1, Frac16 f16In2)
```

3.74.2 Prototype

```
inline Frac32 MLIB_Msu32LLSSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Msu32LLSSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac16 f16In2)
```

3.74.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-148. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f16In2	In	SF16	0x8000... 0x7FFF	input value

3.74.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.74.5 Dependencies

The dependent files are:

- MLIB_Mac24Asm.h
- MLIB_types.h

3.74.6 Description

The **MLIB_Msu32LLSSat** returns the product of two fractional inputs subtracted from the fractional accumulator. The function saturates the output if necessary.

$$\text{MLIB\_Msu32LLSSat}(a, b, c) = a - [(b \cdot c) \gg 16] \qquad \textit{Eqn. 3-70}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac16

3.74.7 Returns

This function returns the 32-bit value of the upper 32 bits of the product of a 32-bit fractional input (f32In1) and a 16-bit fractional input (f16In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.74.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.74.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Msu32LLSSat** does not require the saturation mode to be turned on.

3.74.10 Implementation

The **MLIB_Msu32LLSSat** function is implemented as an inline function.

Example 3-74. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac16 mf16In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.6);
```

```
        mf32In1 = FRAC32(0.1);
        mf16In2 = FRAC16(-0.2);

        /* mf32Out = mf32Acc - mf32In1*mf16In2 */
        mf32Out = MLIB_Msu32LLSSat(mf32Acc, mf32In1, mf16In2);
}
```

3.74.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.74.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-149. Performance of the MLIB_Msu32LLSSat Function

Code Size (words)	V2: 7, V3: 4	
Data Size (words)	0	
Execution Clock	Min	V2: 17, V3: 13 cycles
	Max	V2: 17, V3: 13 cycles

3.75 MLIB_Mac32LLL

This function returns the 32-bit sum of the 32-bit fractional accumulator and the upper 32 bits of the product of two 32-bit fractional inputs.

3.75.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Mac32LLL(Frac32 f32Acc, Frac32 f32In1, Frac32 f32In2)
```

3.75.2 Prototype

```
inline Frac32 MLIB_Mac32LLLFAsmi(register Frac32 f32Acc, register Frac32
f32In1, register Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mac32LLLFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

3.75.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-150. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.75.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.75.5 Dependencies

The dependent files are:

- MLIB_Mac32Asm.h
- MLIB_types.h

3.75.6 Description

The **MLIB_Mac32LLL** returns the sum of the accumulator and the product of two fractional inputs. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Mac32LLL}(a, b, c) = a + [(b \cdot c) \gg 32] \qquad \textit{Eqn. 3-71}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac32

3.75.7 Returns

The function returns the 32-bit sum of the 32-bit fractional accumulator (f32Acc) and the upper 32 bits of the product of two 32-bit fractional inputs (f32In1 and f32In2).

3.75.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.75.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Mac32LLL** function requires the saturation mode to be turned on or the **MLIB_Mac32LLLSat** has to be used instead.

3.75.10 Implementation

The **MLIB_Mac32LLL** function is implemented as an inline function.

Example 3-75. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
```

```
        mf32Acc = FRAC32(0.5);
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32Acc + mf32In1*mf32In2 */
        mf32Out = MLIB_Mac32LLL(mf32Acc, mf32In1, mf32In2);
}
```

3.75.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLLSat**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.75.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-151. Performance of the MLIB_Mac32LLL Function

Code Size (words)	V2: 10, V3: 2	
Data Size (words)	0	
Execution Clock	Min	V2: 20, V3: 12 cycles
	Max	V2: 20, V3: 12 cycles

3.76 MLIB_Mac32LLLSat

This function returns the 32-bit sum of the 32-bit fractional accumulator and the upper 32 bits of the product of two 32-bit fractional inputs with saturation.

3.76.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Mac32LLLSat(Frac32 f32Acc, Frac32 f32In1, Frac32 f32In2)
```

3.76.2 Prototype

```
inline Frac32 MLIB_Mac32LLLSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Mac32LLLSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

3.76.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-152. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.76.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.76.5 Dependencies

The dependent files are:

- MLIB_Mac32Asm.h
- MLIB_types.h

3.76.6 Description

The **MLIB_Mac32LLLSat** returns the sum of the accumulator and the product of two fractional inputs. The function saturates the output if necessary.

$$\mathrm{MLIB\_Mac32LLLSat}(a, b, c) = a + [(b \cdot c) \gg 32] \qquad \textit{Eqn. 3-72}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac32

3.76.7 Returns

The function returns the 32-bit sum of the 32-bit fractional accumulator (f32Acc) and the upper 32 bits of the product of two 32-bit fractional inputs (f32In1 and f32In2).

3.76.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.76.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Mac32LLLSat** does not require the saturation mode to be turned on.

3.76.10 Implementation

The **MLIB_Mac32LLLSat** function is implemented as an inline function.

Example 3-76. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
        mf32Acc = FRAC32(0.6);
        mf32In1 = FRAC32(0.1);
```

```
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32Acc + mf32In1*mf32In2 */
        mf32Out = MLIB_Mac32LLLSat(mf32Acc, mf32In1, mf32In2);
}
```

3.76.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Msu32LLL** and **MLIB_Msu32LLLSat** for more information.

3.76.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-153. Performance of the MLIB_Mac32LLLSat Function

Code Size (words)	V2: 11, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 21, V3: 13 cycles
	Max	V2: 21, V3: 13 cycles

3.77 MLIB_Msu32LLL

This function returns the 32-bit value of the upper 32 bits of the product of two 32-bit fractional inputs subtracted from the 32-bit fractional accumulator.

3.77.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Msu32LLL(Frac32 f32Acc, Frac32 f32In1, Frac32 f32In2)
```

3.77.2 Prototype

```
inline Frac32 MLIB_Msu32LLLFAsmi(register Frac32 f32Acc, register Frac32
f32In1, register Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Msu32LLLFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

3.77.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-154. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.77.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.77.5 Dependencies

The dependent files are:

- MLIB_Mac32Asm.h
- MLIB_types.h

3.77.6 Description

The **MLIB_Msu32LLL** returns the product of two fractional inputs subtracted from the fractional accumulator. The function does not saturate the output if the saturation mode is turned off.

$$\mathrm{MLIB\_Msu32LLL}(a, b, c) = a - [(b \cdot c) \gg 32] \quad \textit{Eqn. 3-73}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac32

3.77.7 Returns

This function returns the 32-bit value of the upper 32 bits of the product of two 32-bit fractional inputs (f32In1 and f32In2) subtracted from the 32-bit fractional accumulator (f32Acc).

3.77.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.77.9 Special Issues

If the result is greater than 0x7FFF FFFF or smaller than 0x8000 0000, the output overflows if the saturation mode is turned off.

In case of desired saturation, the **MLIB_Msu32LLL** function requires the saturation mode to be turned on or the **MLIB_Msu32LLLSat** has to be used instead.

3.77.10 Implementation

The **MLIB_Msu32LLL** function is implemented as an inline function.

Example 3-77. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
```

```
        mf32Acc = FRAC32(0.6);
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32Acc - mf32In1*mf32In2 */
        mf32Out = MLIB_Msu32LLL(mf32Acc, mf32In1, mf32In2);
}
```

3.77.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat** and **MLIB_Msu32LLLSat** for more information.

3.77.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-155. Performance of the MLIB_Msu32LLL Function

Code Size (words)	V2: 10, V3: 2	
Data Size (words)	0	
Execution Clock	Min	V2: 20, V3: 12 cycles
	Max	V2: 20, V3: 12 cycles

3.78 MLIB_Msu32LLLSat

This function returns the 32-bit value of the upper 32 bits of the product of two 32-bit fractional inputs substracted from the 32-bit fractional accumulator with saturation.

3.78.1 Synopsis

```
#include ”mlib.h”
Frac32 MLIB_Msu32LLLSat(Frac32 f32Acc, Frac32 f32In1, Frac32 f32In2)
```

3.78.2 Prototype

```
inline Frac32 MLIB_Msu32LLLSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

V3 core version:

```
inline Frac32 MLIB_V3Msu32LLLSatFAsmi(register Frac32 f32Acc, register
Frac32 f32In1, register Frac32 f32In2)
```

3.78.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-156. Function Arguments

Name	In/Out	Format	Range	Description
f32Acc	In	SF32	0x8000 0000... 0x7FFF FFFF	accumulator value
f32In1	In	SF32	0x8000 0000... 0x7FFF FFFF	input value
f32In2	In	SF32	0x8000 0000... 0x7FFF FFFF	input value

3.78.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.78.5 Dependencies

The dependent files are:

- MLIB_Mac32Asm.h
- MLIB_types.h

3.78.6 Description

The **MLIB_Msu32LLLSat** returns the product of two fractional inputs substracted from the fractional accumulator. The function saturates the output if necessary.

$$\text{MLIB\_Msu32LLLSat}(a, b, c) = a - [(b \cdot c) \gg 32] \qquad \textbf{\textit{Eqn. 3-74}}$$

where:

- result - Frac32
- a - Frac32
- b - Frac32
- c - Frac32

3.78.7 Returns

This function returns the 32-bit value of the upper 32 bits of the product of two 32-bit fractional inputs (f32In1 and f32In2) substracted from the 32-bit fractional accumulator (f32Acc). The function saturates the output if necessary.

3.78.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-1, 1).

3.78.9 Special Issues

If the result is greater than 0x7FFF FFFF the output is 0x7FFF FFFF. If the result is smaller than 0x8000 0000 the output is 0x8000 0000.

The function **MLIB_Msu32LLLSat** does not require the saturation mode to be turned on.

3.78.10 Implementation

The **MLIB_Msu32LLLSat** function is implemented as an inline function.

Example 3-78. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Acc;
static Frac32 mf32In1;
static Frac32 mf32In2;
static Frac32 mf32Out;

void main(void)
{
         mf32Acc = FRAC32(-0.3);
```

```
        mf32In1 = FRAC32(0.1);
        mf32In2 = FRAC32(-0.2);

        /* mf32Out = mf32Acc - mf32In1*mf32In2 */
        mf32Out = MLIB_Msu32LLLSat(mf32Acc, mf32In1, mf32In2);
}
```

3.78.11 See Also

See **MLIB_Mac32LLS**, **MLIB_Mac32LLSSat**, **MLIB_Msu32LLS**, **MLIB_Msu32LLSSat**, **MLIB_Mac32LLL**, **MLIB_Mac32LLLSat** and **MLIB_Msu32LLL** for more information.

3.78.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-157. Performance of the MLIB_Msu32LLLSat Function

Code Size (words)	V2: 11, V3: 3	
Data Size (words)	0	
Execution Clock	Min	V2: 21, V3: 13 cycles
	Max	V2: 21, V3: 13 cycles

3.79 MLIB_Div1Q16SS

This function performs the single-quadrant division of two 16-bit non-negative fractional inputs with the 16-bit result.

3.79.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div1Q16SS(Frac16 f16Num, Frac16 f16Denom)
```

3.79.2 Prototype

```
inline Frac16 MLIB_Div1Q16SSFAsmi(register Frac16 f16Num, register
Frac16 f16Denom)
```

3.79.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-158. Function Arguments

Name	In/Out	Format	Range	Description
f16Num	In	SF16	0x8000... 0x7FFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.79.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.79.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.79.6 Description

The **MLIB_Div1Q16SS** function returns the single quadrant division of two non-negative fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\text{MLIB\_Div1Q16SS}(a, b) = \frac{a \ll 16}{b} \quad \textbf{Eqn. 3-75}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.79.7 Returns

The function divides a 16-bit non-negative fractional numerator (f16Num) by a 16-bit non-negative fractional denominator (f16Denom) with the 16-bit fractional result.

3.79.8 Range Issues

The input data value is in the range of <0,1). The output data value is in the range <0, 1).

3.79.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff.

The function **MLIB_Div1Q16SS** does not require the saturation mode to be turned on.

3.79.10 Implementation

The **MLIB_Div1Q16SS** function is implemented as an inline function.

Example 3-79. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Num, mf16Denom, mf16Out;

void main(void)
{
        mf16Num = FRAC16(0.1);
        mf16Denom = FRAC16(0.2);

        /* mf16Out = mf16Num/mf16Denom */
        mf16Out = MLIB_Div1Q16SS(mf16Num, mf16Denom);
}
```

3.79.11 See Also

See **MLIB_Div4Q16SS**, **MLIB_Div1Q16LS**, **MLIB_Div4Q16LS**, **MLIB_Div1Q32LS** and **MLIB_Div4Q32LS** for more information.

3.79.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-159. Performance of the MLIB_Div1Q16SS Function

Code Size (words)	20	
Data Size (words)	0	
Execution Clock	Min	49 cycles
	Max	49 cycles

3.80 MLIB_Div4Q16SS

This function performs the four-quadrant division of two 16-bit non-negative fractional inputs with the 16-bit result.

3.80.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div4Q16SS(Frac16 f16Num, Frac16 f16Denom)
```

3.80.2 Prototype

```
inline Frac16 MLIB_Div4Q16SSFAsmi(register Frac16 f16Num, register
Frac16 f16Denom)
```

3.80.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-160. Function Arguments

Name	In/Out	Format	Range	Description
f16Num	In	SF16	0x8000... 0x7FFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.80.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.80.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.80.6 Description

The **MLIB_Div4Q16SS** function returns the division of two fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\mathrm{MLIB\_Div4Q16SS}(a, b) = \frac{a \ll 16}{b} \qquad \textbf{Eqn. 3-76}$$

where:

- result - Frac16
- a - Frac16
- b - Frac16

3.80.7 Returns

The function divides a 16-bit fractional numerator (f16Num) by a 16-bit fractional denominator (f16Denom) with the 16-bit fractional result.

3.80.8 Range Issues

The input data value is in the range of <1,1). The output data value is in the range <1, 1).

3.80.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff if the numerator is greater than 0, otherwise 0x8000.

The function **MLIB_Div4Q16SS** does not require the saturation mode to be turned on.

3.80.10 Implementation

The **MLIB_Div4Q16SS** function is implemented as an inline function.

Example 3-80. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Num;
static Frac16 mf16Denom;
static Frac16 mf16Out;

void main(void)
{
        mf16Num = FRAC16(-0.1);
        mf16Denom = FRAC16(0.2);

        /* mf16Out = mf16Num/mf16Denom */
        mf16Out = MLIB_Div4Q16SS(mf16Num, mf16Denom);
}
```

3.80.11 See Also

See **MLIB_Div1Q16SS**, **MLIB_Div1Q16LS**, **MLIB_Div4Q16LS**, **MLIB_Div1Q32LS** and **MLIB_Div4Q32LS** for more information.

3.80.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-161. Performance of the MLIB_Div4Q16SS Function

Code Size (words)	26	
Data Size (words)	0	
Execution Clock	Min	56 cycles
	Max	56 cycles

3.81 MLIB_Div1Q16LS

This function performs the single-quadrant division of a 32-bit non-negative fractional numerator and a 16-bit non-negative fractional denominator with the 16-bit fractional result.

3.81.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div1Q16LS(Frac32 f32Num, Frac16 f16Denom)
```

3.81.2 Prototype

```
inline Frac16 MLIB_Div1Q16LSFAsmi(register Frac32 f32Num, register
Frac16 f16Denom)
```

3.81.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-162. Function Arguments

Name	In/Out	Format	Range	Description
f32Num	In	SF32	0x8000 0000... 0x7FFF FFFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.81.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.81.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.81.6 Description

The **MLIB_Div1Q16LS** function returns the single-quadrant division of two non-negative fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\text{MLIB\_Div1Q16LS}(a, b) = \frac{a}{b} \qquad \textbf{\textit{Eqn. 3-77}}$$

where:

- result - Frac16
- a - Frac32
- b - Frac16

3.81.7 Returns

The function divides a 32-bit non-negative fractional numerator (f32Num) by a 16-bit non-negative fractional denominator (f16Denom) and returns the 16-bit fractional result.

3.81.8 Range Issues

The input data value is in the range of <0,1). The output data value is in the range <0, 1).

3.81.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff.

The function **MLIB_Div1Q16LS** does not require the saturation mode to be turned on.

3.81.10 Implementation

The **MLIB_Div1Q16LS** function is implemented as an inline function.

Example 3-81. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Num;
static Frac16 mf16Denom, mf16Out;

void main(void)
{
         mf32Num = FRAC32(0.1);
         mf16Denom = FRAC16(0.2);

         /* mf16Out = mf32Num/mf16Denom */
         mf16Out = MLIB_Div1Q16LS(mf32Num, mf16Denom);
}
```

3.81.11 See Also

See **MLIB_Div1Q16SS**, **MLIB_Div4Q16SS**, **MLIB_Div4Q16LS**, **MLIB_Div1Q32LS** and **MLIB_Div4Q32LS** for more information.

3.81.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-163. Performance of the MLIB_Div1Q16LS Function

Code Size (words)	22	
Data Size (words)	0	
Execution Clock	Min	50 cycles
	Max	50 cycles

3.82 MLIB_Div4Q16LS

This function performs the four-quadrant division of a 32-bit fractional numerator and a 16-bit fractional denominator with the 16-bit fractional result.

3.82.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div4Q16LS(Frac32 f32Num, Frac16 f16Denom)
```

3.82.2 Prototype

```
inline Frac16 MLIB_Div1Q16LSFAsmi(register Frac32 f32Num, register
Frac16 f16Denom)
```

3.82.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-164. Function Arguments

Name	In/Out	Format	Range	Description
f32Num	In	SF32	0x8000 0000... 0x7FFF FFFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.82.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.82.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.82.6 Description

The **MLIB_Div4Q16LS** function returns the four-quadrant division of two non-negative fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\mathrm{MLIB\_Div4Q16LS}(a, b) = \frac{a}{b} \qquad \textbf{\textit{Eqn. 3-78}}$$

where:

- result - Frac16
- a - Frac32
- b - Frac16

3.82.7 Returns

The function divides a 32-bit fractional numerator (f32Num) by a 16-bit fractional denominator (f16Denom) and returns the 16-bit fractional result.

3.82.8 Range Issues

The input data value is in the range of <1,1). The output data value is in the range <1, 1).

3.82.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff if the numerator is greater than 0, otherwise 0x8000.

The function **MLIB_Div4Q16LS** does not require the saturation mode to be turned on.

3.82.10 Implementation

The **MLIB_Div4Q16LS** function is implemented as an inline function.

Example 3-82. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Num;
static Frac16 mf16Denom, mf16Out;

void main(void)
{
        mf32Num = FRAC32(0.1);
        mf16Denom = FRAC16(0.2);

        /* mf16Out = mf32Num/mf16Denom */
        mf16Out = MLIB_Div4Q16LS(mf32Num, mf16Denom);
}
```

3.82.11 See Also

See **MLIB_Div1Q16SS**, **MLIB_Div4Q16SS**, **MLIB_Div1Q16LS**, **MLIB_Div1Q32LS** and **MLIB_Div4Q32LS** for more information.

3.82.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-165. Performance of the MLIB_Div4Q16LS Function

Code Size (words)	28	
Data Size (words)	0	
Execution Clock	Min	56 cycles
	Max	56 cycles

3.83 MLIB_Div1Q32LS

This function performs the single-quadrant division of 32-bit non-negative fractional numerator and a 16-bit non-negative fractional denominator with the 32-bit fractional result.

3.83.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div1Q32LS(Frac32 f32Num, Frac16 f16Denom)
```

3.83.2 Prototype

```
inline Frac16 MLIB_Div1Q16LSFAsmi(register Frac32 f32Num, register
Frac16 f16Denom)
```

3.83.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-166. Function Arguments

Name	In/Out	Format	Range	Description
f32Num	In	SF32	0x8000 0000... 0x7FFF FFFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.83.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.83.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.83.6 Description

The **MLIB_Div1Q32LS** function returns the single quadrant division of two non-negative fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\text{MLIB\_Div1Q32LS}(a, b) = \frac{a}{b} \ll 16 \qquad \textbf{Eqn. 3-79}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16

3.83.7 Returns

The function divides a 32-bit non-negative fractional numerator (f32Num) by a 16-bit non-negative fractional denominator (f16Denom) and returns the 32-bit fractional result.

3.83.8 Range Issues

The input data value is in the range of <0,1). The output data value is in the range <0, 1).

3.83.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff.

The function **MLIB_Div1Q32LS** does not require the saturation mode to be turned on.

3.83.10 Implementation

The **MLIB_Div1Q32LS** function is implemented as an inline function.

Example 3-83. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Num, mf32Out;
static Frac16 mf16Denom;

void main(void)
{
        mf32Num = FRAC32(0.1);
        mf16Denom = FRAC16(0.2);

        /* mf32Out = mf32Num/mf16Denom */
        mf32Out = MLIB_Div1Q32LS(mf32Num, mf16Denom);
}
```

3.83.11 See Also

See **MLIB_Div1Q16SS**, **MLIB_Div4Q16SS**, **MLIB_Div1Q16LS**, **MLIB_Div1Q16LS** and **MLIB_Div4Q32LS** for more information.

3.83.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-167. Performance of the MLIB_Div1Q32LS Function

Code Size (words)	26	
Data Size (words)	0	
Execution Clock	Min	70 cycles
	Max	70 cycles

3.84 MLIB_Div4Q32LS

This function performs the four-quadrant division of 32-bit fractional numerator and a 16-bit fractional denominator with the 32-bit fractional result.

3.84.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Div4Q32LS(Frac32 f32Num, Frac16 f16Denom)
```

3.84.2 Prototype

```
inline Frac16 MLIB_Div1Q16LSFAsmi(register Frac32 f32Num, register
Frac16 f16Denom)
```

3.84.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-168. Function Arguments

Name	In/Out	Format	Range	Description
f32Num	In	SF32	0x8000 0000... 0x7FFF FFFF	numerator value
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.84.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.84.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.84.6 Description

The **MLIB_Div4Q32LS** function returns the four-quadrant division of two fractional inputs. The function normalizes the inputs to get higher precision of division.

$$\text{MLIB\_Div4Q32LS}(a, b) = \frac{a}{b} \ll 16 \qquad \textbf{Eqn. 3-80}$$

where:

- result - Frac32
- a - Frac32
- b - Frac16

3.84.7 Returns

The function divides a 32-bit fractional numerator (f32Num) by a 16-bit non-negative fractional denominator (f16Denom) and returns the 32-bit fractional result.

3.84.8 Range Issues

The input data value is in the range of <1,1). The output data value is in the range <1, 1).

3.84.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff if the numerator is greater than 0, otherwise 0x8000 0000.

The function **MLIB_Div4Q32LS** does not require the saturation mode to be turned on.

3.84.10 Implementation

The **MLIB_Div4Q32LS** function is implemented as an inline function.

Example 3-84. Implementation Code

```
#include "mlib.h"

static Frac32 mf32Num, mf32Out;
static Frac16 mf16Denom;

void main(void)
{
         mf32Num = FRAC32(0.1);
         mf16Denom = FRAC16(0.2);

         /* mf32Out = mf32Num/mf16Denom */
         mf32Out = MLIB_Div4Q32LS(mf32Num, mf16Denom);
}
```

3.84.11 See Also

See **MLIB_Div1Q16SS**, **MLIB_Div4Q16SS**, **MLIB_Div1Q16LS**, **MLIB_Div1Q16LS** and **MLIB_Div1Q32LS** for more information.

3.84.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-169. Performance of the MLIB_Div4Q32LS Function

Code Size (words)	32	
Data Size (words)	0	
Execution Clock	Min	77 cycles
	Max	77 cycles

3.85 MLIB_Rcp161Q

This function calculates the the single-quadrant 16-bit precision reciprocal function of the 16-bit fractional input with the 32-bit result.

3.85.1 Synopsis

```
#include "mlib.h"
Frac16 MLIB_Rcp161Q(Frac16 f16Denom)
```

3.85.2 Prototype

```
inline Frac16 MLIB_Rcp161QFAsmi(register Frac16 f16Denom)
```

3.85.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-170. Function Arguments

Name	In/Out	Format	Range	Description
f16Denom	In	SF16	0x0... 0x7FFF	denominator value

3.85.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.85.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.85.6 Description

The **MLIB_Rcp161Q** function returns the single-quadrant division of fractional 1 by a non-negative fractional input. The function normalizes the inputs to get higher precision of division. The 16-bit precision division is performed.

$$\text{MLIB\_Rcp161Q}(a) = \frac{1}{a} \quad \textbf{\textit{Eqn. 3-81}}$$

where:

- result - Frac32
  — upper 16 bits: Word16 (scale: 0x4000 to 32768)
  — lower 16 bits: Frac16 (scale: 0x8000 corresponds to 1)
- a - Frac16

3.85.7 Returns

The function divides the fractional 1 by a 16-bit non-negative fractional denominator (f16Denom) with the 32-bit result.

3.85.8 Range Issues

The input data value is in the range of <0,1). The output data value is in the range <0, 32768).

3.85.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff.

The result precision is below 3 LSB if the input is 4096 and greater. If higher precision for lower inputs is required, the **MLIB_Rcp321Q** function has to be used.

The function **MLIB_Rcp161Q** does not require the saturation mode to be turned on.

3.85.10 Implementation

The **MLIB_Rcp161Q** function is implemented as an inline function.

Example 3-85. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Denom;
static Frac32 mf132Out;

void main(void)
{
        mf16Denom = FRAC16(0.2);

        /* mf32Out = 1 / mf16Denom */
        mf32Out = MLIB_Rcp161Q(mf16Denom);
}
```

3.85.11 See Also

See **MLIB_Rcp164Q**, **MLIB_Rcp321Q** and **MLIB_Rcp324Q** for more information.

3.85.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-171. Performance of the MLIB_Rcp161Q Function

Code Size (words)	18	
Data Size (words)	0	
Execution Clock	Min	44 cycles
	Max	44 cycles

3.86 MLIB_Rcp164Q

This function calculates the the four-quadrant 16-bit precision reciprocal function of the 16-bit fractional input with the 32-bit result.

3.86.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Rcp164Q(Frac16 f16Denom)
```

3.86.2 Prototype

```
inline Frac32 MLIB_Rcp164QFAsmi(register Frac16 f16Denom)
```

3.86.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-172. Function Arguments

Name	In/Out	Format	Range	Description
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.86.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.86.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.86.6 Description

The **MLIB_Rcp164Q** function returns the four-quadrant division of fractional 1 by a fractional input. The function normalizes the inputs to get higher precision of division. The 16-bit precision division is performed.

$$\mathrm{MLIB\_Rcp321Q}(a) = \frac{1}{a} \qquad \textit{Eqn. 3-82}$$

where:

- result - Frac32
  — upper 16 bits: Word16 (scale: 0x4000 to 32768)
  — lower 16 bits: Frac16 (scale: 0x8000 corresponds to 1)
- a - Frac16

3.86.7 Returns

The function divides the fractional 1 by a 16-bit non-negative fractional denominator (f16Denom) with the 32-bit result.

3.86.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-32768, 32768).

3.86.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff.

The result precision is below 3 LSB if the input is 4096 and greater. If higher precision for lower inputs is required, the **MLIB_Rcp324Q** function has to be used.

The function **MLIB_Rcp164Q** does not require the saturation mode to be turned on.

3.86.10 Implementation

The **MLIB_Rcp164Q** function is implemented as an inline function.

Example 3-86. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Denom;
static Frac32 mf132Out;

void main(void)
{
         mf16Denom = FRAC16(0.2);

         /* mf32Out = 1 / mf16Denom */
         mf32Out = MLIB_Rcp164Q(mf16Denom);
}
```

3.86.11 See Also

See **MLIB_Rcp161Q**, **MLIB_Rcp321Q** and **MLIB_Rcp324Q** for more information.

3.86.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-173. Performance of the MLIB_Rcp164Q Function

Code Size (words)	22	
Data Size (words)	0	
Execution Clock	Min	45 cycles
	Max	45 cycles

3.87 MLIB_Rcp321Q

This function calculates the the single-quadrant 32-bit precision reciprocal function of the 16-bit fractional input with the 32-bit result.

3.87.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Rcp321Q(Frac16 f16Denom)
```

3.87.2 Prototype

```
inline Frac32 MLIB_Rcp321QFAsmi(register Frac16 f16Denom)
```

3.87.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-174. Function Arguments

Name	In/Out	Format	Range	Description
f16Denom	In	SF16	0x0... 0x7FFF	denominator value

3.87.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.87.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.87.6 Description

The **MLIB_Rcp321Q** function returns the single-quadrant division of fractional 1 by a non-negative fractional input. The function normalizes the inputs to get higher precision of division. The 32-bit precision division is performed.

$$\mathrm{MLIB\_Rcp161Q}(a) = \frac{1}{a} \qquad \textit{Eqn. 3-83}$$

where:

- result - Frac32
  - upper 16 bits: Word16 (scale: 0x4000 to 32768)
  - lower 16 bits: Frac16 (scale: 0x8000 corresponds to 1)
- a - Frac16

3.87.7 Returns

The function divides the fractional 1 by a 16-bit non-negative fractional denominator (f16Denom) with the 32-bit result.

3.87.8 Range Issues

The input data value is in the range of <0,1). The output data value is in the range <0, 32768).

3.87.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff.

The function **MLIB_Rcp321Q** does not require the saturation mode to be turned on.

3.87.10 Implementation

The **MLIB_Rcp321Q** function is implemented as an inline function.

Example 3-87. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Denom;
static Frac32 mf132Out;

void main(void)
{
        mf16Denom = FRAC16(0.2);

        /* mf32Out = 1 / mf16Denom */
        mf32Out = MLIB_Rcp321Q(mf16Denom);
}
```

3.87.11 See Also

See **MLIB_Rcp161Q**, **MLIB_Rcp164Q** and **MLIB_Rcp324Q** for more information.

3.87.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-175. Performance of the MLIB_Rcp321Q Function

Code Size (words)	23	
Data Size (words)	0	
Execution Clock	Min	62 cycles
	Max	62 cycles

3.88 MLIB_Rcp324Q

This function calculates the the four-quadrant 32-bit precision reciprocal function of the 16-bit fractional input with the 32-bit result.

3.88.1 Synopsis

```
#include "mlib.h"
Frac32 MLIB_Rcp324Q(Frac16 f16Denom)
```

3.88.2 Prototype

```
inline Frac32 MLIB_Rcp324QFAsmi(register Frac16 f16Denom)
```

3.88.3 Arguments

This subsection describes the input/output arguments to a function or a macro. It explains the algorithms being used by the functions or macro.

Table 3-176. Function Arguments

Name	In/Out	Format	Range	Description
f16Denom	In	SF16	0x8000... 0x7FFF	denominator value

3.88.4 Availability

This library module is available in the C-callable interface assembly version format.

This library module is targeted for the 56800E and 56800Ex platforms.

3.88.5 Dependencies

The dependent files are:

- MLIB_DivAsm.h
- MLIB_types.h

3.88.6 Description

The **MLIB_Rcp324Q** function returns the four-quadrant division of fractional 1 by a fractional input. The function normalizes the inputs to get higher precision of division. The 32-bit precision division is performed.

$$\mathrm{MLIB\_Rcp324Q}(a) = \frac{1}{a} \quad \text{Eqn. 3-84}$$

where:

- result - Frac32
  - upper 16 bits: Word16 (scale: 0x4000 to 32768)
  - lower 16 bits: Frac16 (scale: 0x8000 corresponds to 1)
- a - Frac16

3.88.7 Returns

The function divides the fractional 1 by a 16-bit fractional denominator (f16Denom) with the 32-bit result.

3.88.8 Range Issues

The input data value is in the range of <-1,1). The output data value is in the range <-32768, 32768).

3.88.9 Special Issues

If the denominator is equal to 0, the result is 0x7fff ffff.

The function **MLIB_Rcp324Q** does not require the saturation mode to be turned on.

3.88.10 Implementation

The **MLIB_Rcp324Q** function is implemented as an inline function.

Example 3-88. Implementation Code

```
#include "mlib.h"

static Frac16 mf16Denom;
static Frac32 mf132Out;

void main(void)
{
        mf16Denom = FRAC16(0.2);

        /* mf32Out = 1 / mf16Denom */
        mf32Out = MLIB_Rcp324Q(mf16Denom);
}
```

3.88.11 See Also

See **MLIB_Rcp161Q**, **MLIB_Rcp164Q** and **MLIB_Rcp321Q** for more information.

3.88.12 Performance

This section specifies actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

Table 3-177. Performance of the MLIB_Rcp324Q Function

Code Size (words)	27	
Data Size (words)	0	
Execution Clock	Min	66 cycles
	Max	66 cycles

Appendix A Revision History

Table 0-1. Revision history

Revision number	Date	Subsequent changes
0	02/2014	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

56800Ex_MLIB
Rev. 0, 02/2014