Math Scalar Library
Reference Manual (C/C++)
DD-00003-010

Jan Adelsbach

January 26, 2024

Contents

1 About this Guide

1.1 Legal Information

1.2 Feedback and Contact

For feedback on this document, please use the following email address:
techpubs@adelsbach-research.eu

Please include the page number or a link to the page.

For general contact details, please visit https://adelsbach-research.eu/contact.

1.3 Introduction

This manual describes the *Application Programming Interface* (API) of the *Math Scalar Library*.

1.4 Audience for This Guide

The audience of this guide is assumed to be C or C++ programmers who understand the basic concepts of at least one of the aforementioned programming languages.

Familiarity with the standard math libraries of the C or C++ language are recommended.

1.5 How to Use This Guide

This guide first describes some general programming details of the library and then documents each function individually.

The documentation for each function applies both the single and double precision versions. The former can be differentiated by a suffix letter `f`.

1.6 Conventions Used in This Guide

x
: Normal math typesetting represents a normal variable.

$\mathbf{x}$
: Bold math typesetting represents a vector.

`Mono`
: Monospace typesetting represents C function names, variables or data types.

2 Overview

2.1 Introduction

The *Math Scalar Library* is a function library with high performance scalar versions of standard mathematical functions. It is aimed at producing the same results as the *Math Vector Library* and as such allows consistency in numerical computations considering ULP errors.

2.2 Function Naming Scheme

All functions follow the standard naming scheme of the standard system math library for C/C++ as found in the headers `math.h` and `cmath` but with letter "`s`" as prefix.

2.3 Thread Safety

All routines in the library are completely thread-safe, as long as the data supplied in arguments is exclusive to the current thread.

2.4 FPU Signaling

All functions set FPU signals in the same manner as the standard system math library functions, this includes:

- denormalized
- overflow
- underflow
- inexact
- invalid (NaN)

3 Utility

3.1 `mscaver` - Version query

```
#include <msca.h>

void mscaver(int *major, int *minor);
```

Queries the version of the library and stores the *major* and *minor* version numbers in the respective arguments.

3.1.1 Parameters

`MAJOR` - **INTEGER** *EXIT:* The major version number of the library.

`MINOR` - **INTEGER** *EXIT:* The minor version number of the library

4 Rounding

4.1 sfloor - Round down to nearest integral part

```
#include <msca.h>

double sfloor (double x);
float  sfloorf(float x);
```

Given a value x this function rounds x to the nearest integral part less or equal than itself.

4.2 sceil - Round up to nearest integral part

```
#include <msca.h>

double sceil (double x);
float  sceilf(float x);
```

Given a value x this function rounds x to the nearest integral part greater or equal than itself.

4.3 strunc - Truncate to nearest integral part

```
#include <msca.h>

double strunc (double x);
float  struncf(float x);
```

Given a value x this function rounds x to the nearest integral part less or equal than $|x|$.

4.4 sround - Round to nearest integral part

```
#include <msca.h>

double sround (double x);
float  sroundf(float x);
```

Given a value x this function rounds x to the nearest integral part.

5 Roots

5.1 ssqrt - Square root $\sqrt{x}$

```
#include <msca.h>

double ssqrt (double x);
float  ssqrtf(float x);
```

Given a value x this function computes the square root of x.

5.2 srsqrt - Inverse square root $1/\sqrt{x}$

```
#include <msca.h>

double srsqrt (double x);
float  srsqrtf(float x);
```

Given a value x this function computes the inverse square root of x.

5.3 scbrt - Cube root $\sqrt[3]{x}$

```
#include <msca.h>

double scbrt (double x);
float  scnrtf(float x);
```

Given a value x this function computes the cube root of x.

5.4 srcbrt - Inverse cube root $1/\sqrt[3]{x}$

```
#include <msca.h>

double srcbrt (double x);
float  srcnrtf(float x);
```

Given a value x this function computes the inverse cube root of x.

6 Trigonometric Functions

6.1 ssin - Sine $\sin(x)$

```
#include <msca.h>

double ssin (double x);
float  ssinf(float x);
```

Given a value x this function computes the sine $\sin(x)$.

6.2 scos - Cosine $\cos(x)$

```
#include <msca.h>

double scos (double x);
float  scosf(float x);
```

Given a value x this function computes the sine $\cos(x)$.

6.3 stan - Tangent $\tan(x)$

```
#include <msca.h>

double stan (double x);
float  stanf(float x);
```

Given a value x this function computes the sine $\tan(x)$.

6.4 sasin - Arcsine $\sin^{-1}(x)$

```
#include <msca.h>

double sasin (double x);
float  sasinf(float x);
```

Given a value x this function computes the arcsine $\sin^{-1}(x)$.

6.5 sacos - Arccosine $\cos^{-1}(x)$

```
#include <msca.h>

double sacos (double x);
float  sacosf(float x);
```

Given a value x this function computes the arccosine $\cos^{-1}(x)$.

6.6 satan - Arctangent $\tan^{-1}(x)$

```
#include <msca.h>

double satan (double x);
float  satanf(float x);
```

Given a value x this function computes the arccosine $\tan^{-1}(x)$.

7 Hypergeometric Functions

7.1 ssinh - Hypergeometric sine $\sinh(x)$

```
#include <msca.h>

double ssinh (double x);
float  ssinhf(float x);
```

Given a value x this function computes the hypergeometric sine of x.

7.2 scosh - Hypergeometric cosine $\cosh(x)$

```
#include <msca.h>

double scosh (double x);
float  scoshf(float x);
```

Given a value x this function computes the hypergeometric cosine of x.

7.3 stanh - Hypergeometric tangent $\tanh(x)$

```
#include <msca.h>

double stanh (double x);
float  stanhf(float x);
```

Given a value x this function computes the hypergeometric tangent of x.

7.4 sasinh - Hypergeometric arcsine $\sinh^{-1}(x)$

```
#include <msca.h>

double sasinh (double x);
float  sasinhf(float x);
```

Given a value x this function computes the hypergeometric arcsine of x.

7.5 sacosh - Hypergeometric arccosine $\cosh^{-1}(x)$

```
#include <msca.h>

double sacosh (double x);
float  sacoshf(float x);
```

Given a value x this function computes the hypergeometric arccosine of x.

7.6 satanh - Hypergeometric arctangent $\tanh^{-1}(x)$

```
#include <msca.h>

double satanh (double x);
float  satanhf(float x);
```

Given a value x this function computes the hypergeometric arctangent of x.

8 Expoentials and Logarithms

8.1 `sexp` - Exponentiation e^x

```
#include <msca.h>

double sexp (double x);
float  sexpf(float x);
```

Given a value x this function raises e to x, that is e^x.

8.2 `sexpm1` - Exponentiation $e^x - 1$

```
#include <msca.h>

double sexpm1 (double x);
float  sexpm1f(float x);
```

Given a value x this function raises e to x and subtracts one, that is $e^x - 1$. This function is more precise than using the equivalent operations.

8.3 `sexp2` - Binary exponentiation 2^x

```
#include <msca.h>

double sexp2 (double x);
float  sexp2f(float x);
```

Given a value x this function raises 2 to x, that is 2^x.

8.4 `slog` - Natural logarithm $\log(x)$

```
#include <msca.h>

double slog (double x);
float  slogf(float x);
```

Given a value x this function computes the natural logarithm.

8.5 `slog2` - Base-2 logarithm $\log_2(x)$

```
#include <msca.h>

double slog2 (double x);
float  slog2f(float x);
```

Given a value x this function computes the base-2 logarithm.

8.6 `slog10` - Base-10 logarithm $\log_{10}(x)$

```
#include <msca.h>

double slog10 (double x);
float  slog10f(float x);
```

Given a value x this function computes the base-10 logarithm.

8.7 slog1p - Natural logarithm plus one $\log(x+1)$

```
#include <msca.h>

double slog1p (double x);
float  slog1pf(float x);
```

Given a value x this function computes the natural logarithm of x plus one, that is $\log(x+1)$. This function is more precise than using the equivalent operations.

8.8 spow - Raise x^y

```
#include <msca.h>

double spow (double x, double y);
float  spowf(float x, float y);
```

Given values x and y this function computes x raised by y, that is x^y.

8.9 sfmod - Remainder mod(x,y)

```
#include <msca.h>

double sfmod (double x, double y);
float  sfmodf(float x, float y);
```

Given values x and y this function computes the remainder mod(x,y).

9 Special Functions

9.1 serf - Error function erf(x)

```
#include <msca.h>

double serf (double x);
float  serff(float x);
```

Given a value x this function computes the error function:

$$\frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$$

9.2 serfc - Complementary error function erfc(x)

```
#include <msca.h>

double serfc (double x);
float  serfcf(float x);
```

Given a value x this function computes the complemenary error function:

$$1 - \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$$

9.3 sj0 - Bessel function $J_0(x)$

```
#include <msca.h>

double sj0 (double x);
float  sj0f(float x);
```

Given a value x this function computes the Bessel function $J_0(x)$.

9.4 sy0 - Bessel function $Y_0(x)$

```
#include <msca.h>

double sy0 (double x);
float  sy0f(float x);
```

Given a value x this function computes the Bessel function $Y_0(x)$.

9.5 sj1 - Bessel function $J_1(x)$

```
#include <msca.h>

double sj1 (double x);
float  sj1f(float x);
```

Given a value x this function computes the Bessel function $J_1(x)$.

9.6 sy1 - Bessel function $Y_1(x)$

```
#include <msca.h>

double sy1 (double x);
float  sy1f(float x);
```

Given a value x this function computes the Bessel function $Y_1(x)$.

9.7 sjn - Bessel function $J_n(x)$

```
#include <msca.h>

double sjn (int n, double x);
float  sjnf(int n, float x);
```

Given a value x and an order n, this function computes the Bessel function $J_n(x)$.

9.8 syn - Bessel function $Y_n(x)$

```
#include <msca.h>

double syn (int n, double x);
float  synf(int n, float x);
```

Given a value x and an order n, this function computes the Bessel function $Y_n(x)$.

9.9 slgamma - Log-Gamma function $\log\Gamma(x)$

```
#include <msca.h>

double slgamma (double x);
float  slgammaf(float x);
```

Given a value x this function computes the Log-Gamma function $\log\Gamma(x)$.

10 Other Functions

10.1 sfabs - Absolute value $|x|$

```
#include <msca.h>

double sfabs (double x);
float  sfabsf(float x);
```

Given a value x this function computes the absolute value $|x|$.

10.2 scopysign - Copy sign

```
#include <msca.h>

double scopysign (double x, double y);
float  scopysignf(float x, float y);
```

Given values x and y this function returns the value of x with its sign changed to that of y. This is the equivalent of:

$$\text{sgn(y)} \times |\text{x}|$$

10.3 shypot - Euclidean distance $\sqrt{x^2+y^2}$

```
#include <msca.h>

double shypot (double x, double y);
float  shypotf(float x, float y);
```

Given values x and y this function computes the euclidean distance:

$$\sqrt{x^2+y^2}$$

10.4 sremainder - Remainder

```
#include <msca.h>

double sremainder (double x, double y);
float  sremainderf(float x, float y);
```

Given values x and y this function computes the remainder of dividing x by y and returns the result.

11 Complex Numbers

11.1 screal - Complex real component Re(x)

```
#include <msca.h>

double screal (double complex x);
float  screalf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex absolute value:

$$\text{Re(x)}$$

11.2 scimag - Complex imaginary component Im(x)

```
#include <msca.h>

double scimag (double complex x);
float  scimagf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex absolute value:

$$\text{Im(x)}$$

11.3 scabs - Complex absolute value $|x|$

```
#include <msca.h>

double scabs (double complex x);
float  scabsf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex absolute value:

$$|x|$$

11.4 scarg - Complex argument arg(x)

```
#include <msca.h>

double scarg (double complex x);
float  scargf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex argument:

$$\arg(x)$$

11.5 sconj - Complex conjugate $\overline{x}$

```
#include <msca.h>

double sconj (double complex x);
float  sconjf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex conjugate:

$$\overline{x}$$

11.6 scproj - Complex Riemann sphere projection proj(x)

```
#include <msca.h>

double scproj (double complex x);
float  scprojf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex Riemann sphere projection:

$$\text{proj}(x)$$

11.7 scexp - Complex exponentiation exp(x)

```
#include <msca.h>

double scexp (double complex x);
float  scexpf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex exponentiation:

$$\exp(x)$$

11.8 sclog - Complex logarithm log(x)

```
#include <msca.h>

double sclog (double complex x);
float  sclogf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex logarithm:

$$\log(x)$$

11.9 scsin - Complex sine sin(x)

```
#include <msca.h>

double scsin (double complex x);
float  scsinf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex sine:

$$\sin(x)$$

11.10 sccos - Complex cosine cos(x)

```
#include <msca.h>

double sccos (double complex x);
float  sccosf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex cosine:

$$\cos(x)$$

11.11 sctan - Complex tangent tan(x)

```
#include <msca.h>

double sctan (double complex x);
float  sctanf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex tangent:

$$\tan(x)$$

11.12 scasin - Complex arcsine $\sin^{-1}(x)$

```
#include <msca.h>

double scasin (double complex x);
float  scasinf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex arcsine:

$$\sin^{-1}(x)$$

11.13 scacos - Complex arccosine $\cos^{-1}(x)$

```
#include <msca.h>

double scacos (double complex x);
float  scacosf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex arccosine:

$$\cos^{-1}(x)$$

11.14 scatan - Complex arctangent $\tan^{-1}(x)$

```
#include <msca.h>

double scatan (double complex x);
float  scatanf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex arctangent:

$$\tan^{-1}(x)$$

11.15 scsinh - Complex hyperbolic sine sinh(x)

```
#include <msca.h>

double scsinh (double complex x);
float  scsinhf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic sine:

$$\sinh(x)$$

11.16 sccosh - Complex hyperbolic cosine $\cosh(x)$

```
#include <msca.h>

double sccosh (double complex x);
float  sccoshf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic cosine:

$$\cosh(x)$$

11.17 sctanh - Complex hyperbolic tangent $\tanh(x)$

```
#include <msca.h>

double sctanh (double complex x);
float  sctanhf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic tangent:

$$\tanh(x)$$

11.18 scasinh - Complex hyperbolic arcsine $\sinh^{-1}(x)$

```
#include <msca.h>

double scasinh (double complex x);
float  scasinhf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic arcsine:

$$\sinh^{-1}(x)$$

11.19 scacosh - Complex hyperbolic arccosine $\cosh^{-1}(x)$

```
#include <msca.h>

double scacosh (double complex x);
float  scacoshf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic cosine:

$$\cosh^{-1}(x)$$

11.20 scatanh - Complex hyperbolic arctangent $\tanh^{-1}(x)$

```
#include <msca.h>

double scatanh (double complex x);
float  scatanhf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex hyperbolic arctangent:

$$\tanh^{-1}(x)$$

11.21 scsqrt - Complex square root $\sqrt{x}$

```
#include <msca.h>

double scsqrt (double complex x);
float  scsqrtf(float complex x);
```

Given a value $x \in \mathbb{C}$ this function computes the complex square root:

$$\sqrt{x}$$

11.22 scpow - Complex power x^y

```
#include <msca.h>

double scpow (double complex x, double complex y);
float  scpowf(float complex x, float complex y);
```

Given values $x, y \in \mathbb{C}$ this function computes the complex power:

$$x^y$$

12 Acknowledgements

Parts of this product include or are derived from code under the following licences:

```
/*
 * Copyright (c) 2003, Steven G. Kargl
 * Copyright (c) 2005, 2011 David Schultz <das@FreeBSD.ORG>
 * Copyright (c) 2005 Bruce D. Evans and Steven G. Kargl
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

/*
 * Copyright (c) 2008 Stephen L. Moshier <steve@moshier.net>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */
```