EXAMPLES

```
z = log(+1.0);    /* z = +0.0 because e0 = 1 */
z = log(-1.0);    /* z = NAN because negative arguments are not
                     allowed. The invalid exception is raised. */
```

log10

You can use the `log10` function to compute the common logarithm of a real number.

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

`x` Any positive floating-point number.

DESCRIPTION

The `log10` function returns the common (base 10) logarithm of its argument.

$$\log 10(x) = \log_{10} x = y \text{ such that } x = 10^{y}$$

EXCEPTIONS

When x is finite and nonzero, the result of $\log 10(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x other than +1)
- invalid (when x is negative)

SPECIAL CASES

Table 10-15 shows the results when the argument to the `log10` function is a zero, a NaN, or an Infinity, plus other special cases for the `log10` function.

Table 10-15 Special cases for the `log10` function

Operation	Result	Exceptions raised
$\log 10(x)$ for $x < 0$	NaN	Invalid
$\log 10(+1)$	+0	None
$\log 10(+0)$	–∞	Divide-by-zero
$\log 10(-0)$	–∞	Divide-by-zero
$\log 10(\mathrm{NaN})$	NaN	None*
$\log 10(+\infty)$	+∞	None
$\log 10(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = log10(+1.0);  /* z = 0.0 because 10^0 = 1 */
z = log10(10.0);  /* z = 1.0 because 10^1 = 10. The inexact
                     exception is raised. */
z = log10(-1.0);  /* z = NAN because negative arguments are not
                     allowed. The invalid exception is raised. */
```

log1p

You can use the `log1p` function to compute the natural logarithm of 1 plus a real number.

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

`x` Any floating-point number greater than –1.

DESCRIPTION

The `log1p` function computes the natural logarithm of 1 plus its argument.

$$\log 1\mathrm{p}(x) = \log_e(x+1) = \ln(x+1) = y \quad \text{such that } 1 + x = 10^y$$

For small numbers, use the function call `log1p(x)` instead of the function call `log(1 + x)`. The call `log1p(x)` produces a more exact result because it avoids the roundoff error that might occur when the expression `1 + x` is computed.

EXCEPTIONS

When x is finite and nonzero, the result of $\log 1p(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of $x > -1$)
- invalid (when x is less than –1)
- divide-by-zero (when x is –1)

SPECIAL CASES

Table 10-16 shows the results when the argument to the `log1p` function is a zero, a NaN, or an Infinity, plus other special cases for the `log1p` function.

Table 10-16 Special cases for the `log1p` function

Operation	Result	Exceptions raised
$\log 1p(x)$ for $x < -1$	NaN	Invalid
$\log 1p(-1)$	$-\infty$	Divide-by-zero
$\log 1p(+0)$	+0	None
$\log 1p(-0)$	–0	None
$\log 1p(\text{NaN})$	NaN	None*
$\log 1p(+\infty)$	$+\infty$	None
$\log 1p(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = log1p(-1.0);  /* z = log(0) = -INFINITY. The divide-by-zero
                     and inexact exceptions are raised. */
z = log1p(0.0);   /* z = log(1) = 0.0 because e^0 = 1. */
z = log1p(-2.0);  /* z = log(-1) = NAN because logarithms of
                     negative numbers are not allowed. The
                     invalid exception is raised. */
```


10 Transcendental Functions

log2

You can use the `log2` function to compute the binary logarithm of a real number.

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

`x` Any positive floating-point number.

DESCRIPTION

The `log2` function returns the binary (base 2) logarithm of its argument.

$$\log 2(x) = \log_2 x = y \text{ such that } x = 2^y$$

The `exp2` function performs the inverse operation.

EXCEPTIONS

When x is finite and nonzero, the result of $\log 2(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x other than +1)
- invalid (when x is negative)

SPECIAL CASES

Table 10-17 shows the results when the argument to the `log2` function is a zero, a NaN, or an Infinity, plus other special cases for the `log2` function.

Table 10-17 Special cases for the `log2` function

Operation	Result	Exceptions raised
$\log 2(x)$ for $x < 0$	NaN	Invalid
$\log 2(+1)$	+0	None
$\log 2(+0)$	$-\infty$	Divide-by-zero
$\log 2(-0)$	$-\infty$	Divide-by-zero
$\log 2(\mathrm{NaN})$	NaN	None*
$\log 2(+\infty)$	$+\infty$	None
$\log 2(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = log2(+1.0);   /* z = +0 because 2^0 = 1 */
z = log2(2.0);    /* z = 1 because 2^1 = 2. The inexact exception
                     is raised. */
z = log2(-1.0);   /* z = NAN because negative arguments are not
                     allowed. The invalid exception is raised. */
```

logb

You can use the `logb` function to determine the value in the exponent field of a floating-point number.

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

`x` Any floating-point number.

DESCRIPTION

The `logb` function returns the signed exponent of its argument x as a signed integer value.

$$\mathrm{logb}(x) = y \text{ such that } x = f \times 2^y$$

When the argument is a denormalized number, the exponent is determined as if the input argument had first been normalized.

Note that for a nonzero finite x, $1 \le \mathrm{fabs}(\mathrm{scalb}(x, -\mathrm{logb}(x))) < 2$.

That is, for a nonzero finite x, the magnitude of x taken to the power of its inverse exponent is between 1 and 2.

This function conforms to IEEE Standard 854, which differs from IEEE Standard 754 on the treatment of a denormalized argument x.

EXCEPTIONS

If x is finite and nonzero, the result of $\mathrm{logb}(x)$ is exact.

SPECIAL CASES

Table 10-18 shows the results when the argument to the `logb` function is a zero, a NaN, or an Infinity.

Table 10-18 Special cases for the `logb` function

Operation	Result	Exceptions raised
$\operatorname{logb}(+0)$	$-\infty$	Divide-by-zero
$\operatorname{logb}(-0)$	$-\infty$	Divide-by-zero
$\operatorname{logb}(\mathrm{NaN})$	NaN	None*
$\operatorname{logb}(+\infty)$	$+\infty$	None
$\operatorname{logb}(-\infty)$	$+\infty$	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = logb(789.9);  /* z = 9.0 because 789.9 ≈ 1.54 × 2^9 */
z = logb(21456789);/* z = 24.0 because 21456789 ≈ 1.28 × 2^24 */
```

modf

You can use the `modf` function to split a real number into a fractional part and an integer part.

```
float modff (float x, float *iptrf);
double modf (double x, double *iptr);
```

`x` Any floating-point number.

`iptr` A pointer to a floating-point variable in which the integer part can be stored upon return.

DESCRIPTION

The `modf` function splits its first argument into a fractional part and an integer part. This is an ANSI standard C function.

$$\operatorname{modf}(x, n) = f \text{ such that } |f| < 1.0 \text{ and } f + n = x$$

The fractional part is returned as the value of the function, and the integer part is stored as a floating-point number in the area pointed to by `iptr`. The fractional part and the integer part both have the same sign as the argument `x`.

EXCEPTIONS

If x is finite and nonzero, the result of $\operatorname{modf}(x, n)$ is exact.

SPECIAL CASES

Table 10-19 shows the results when the floating-point argument to the `modf` function is a zero, a NaN, or an Infinity.

Table 10-19 Special cases for the `modf` function

Operation	Result	Exceptions raised
$\operatorname{modf}(+0, n)$	$+0\ (n = 0)$	None
$\operatorname{modf}(-0, n)$	$-0\ (n = 0)$	None
$\operatorname{modf}(\mathrm{NaN}, n)$	$\mathrm{NaN}\ (n = \mathrm{NaN})$	None*
$\operatorname{modf}(+\infty, n)$	$+0\ (n = +\infty)$	None
$\operatorname{modf}(-\infty, n)$	$-0\ (n = -\infty)$	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = modf(1.0, n); /* z = 0.0 and n = 1.0 */
z = modf(+INFINITY, n); /* z = 0.0 and n = +INFINITY because the
                           value +∞ is an integer. */
```

Trigonometric Functions

MathLib provides the following **trigonometric functions:**

$\cos(x)$	Computes the cosine of x.
$\sin(x)$	Computes the sine of x.
$\tan(x)$	Computes the tangent of x.
$\operatorname{acos}(x)$	Computes the arc cosine of x.
$\operatorname{asin}(x)$	Computes the arc sine of x.
$\operatorname{atan}(x)$	Computes the arc tangent of x.
$\operatorname{atan2}(y, x)$	Computes the arc tangent of y/x.

The remaining trigonometric functions can be computed easily and efficiently from the transcendental functions provided.

The arguments for trigonometric functions (`cos`, `sin`, and `tan`) and return values for inverse trigonometric functions (`acos`, `asin`, `atan`, and `atan2`) are expressed in radians. The cosine, sine, and tangent functions use an argument reduction based on the `remainder` function (see page 6-11 in Chapter 6, "Numeric Operations and Functions") and the constant `pi`, where `pi` is the nearest approximation of π with 53 bits of precision. The cosine, sine, and tangent functions are periodic with respect to the constant `pi`, so their periods are different from their mathematical counterparts and diverge from their counterparts when their arguments become very large.

cos

You can use the `cos` function to compute the cosine of a real number.

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

`x` Any finite floating-point number.

DESCRIPTION

The `cos` function returns the cosine of its argument. The argument is the measure of an angle expressed in radians. This function is **symmetric** with respect to the y-axis ($\cos x = \cos -x$).

The `acos` function performs the inverse operation $(\arccos(y))$.

EXCEPTIONS

When x is finite and nonzero, $\cos(x)$ raises the inexact exception.

SPECIAL CASES

Table 10-20 shows the results when the argument to the `cos` function is a zero, a NaN, or an Infinity, plus other special cases for the `cos` function.

Table 10-20 Special cases for the `cos` function

Operation	Result	Exceptions raised
$\cos(\pi)$	–1	Inexact
$\cos(+0)$	1	None
$\cos(-0)$	1	None
$\cos(\text{NaN})$	NaN	None*
$\cos(+\infty)$	NaN	Invalid
$\cos(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = cos(0);    /* z = 1.0. */
z = cos(pi/2); /* z = –0.0. The inexact exception is raised. */
z = cos(pi);   /* z = –1.0. The inexact exception is raised. */
z = cos(-pi/2);/* z = 0.0. The inexact exception is raised. */
z = cos(–pi);  /* z = –1.0. The inexact exception is raised. */
```

sin

You can use the `sin` function to compute the sine of a real number.

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

`x` Any finite floating-point number.

DESCRIPTION

The `sin` function returns the sine of its argument. The argument is the measure of an angle expressed in radians. This function is **antisymmetric** with respect to the y-axis ($\sin x \neq \sin -x$).

The `asin` function performs the inverse operation $(\arcsin(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\sin(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-21 shows the results when the argument to the `sin` function is a zero, a NaN, or an Infinity, plus other special cases for the `sin` function.

Table 10-21 Special cases for the `sin` function

Operation	Result	Exceptions raised
$\sin(\pi)$	0	Inexact
$\sin(+0)$	+0	None
$\sin(-0)$	–0	None
$\sin(\mathrm{NaN})$	NaN	None*
$\sin(+\infty)$	NaN	Invalid
$\sin(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = sin(pi/2);    /* z = 1. The inexact exception is raised. */
z = sin(pi);      /* z = 0. The inexact exception is raised. */
z = sin(–pi/2);   /* z = –1. The inexact exception is raised. */
z = sin(–pi);     /* z = 0. The inexact exception is raised. */
```

tan

You can use the `tan` function to compute the tangent of a real number.

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

`x` Any finite floating-point number.

DESCRIPTION

The `tan` function returns the tangent of its argument. The argument is the measure of an angle expressed in radians. This function is antisymmetric.

The `atan` function performs the inverse operation $(\arctan(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\tan(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-22 shows the results when the argument to the `tan` function is a zero, a NaN, or an Infinity, plus other special cases for the `tan` function.

Table 10-22 Special cases for the `tan` function

Operation	Result	Exceptions raised
$\tan(\pi)$	0	Inexact
$\tan(\pi/2)$	$+\infty$	Inexact
$\tan(+0)$	+0	None
$\tan(-0)$	–0	None
$\tan(\mathrm{NaN})$	NaN	None*
$\tan(+\infty)$	NaN	Invalid
$\tan(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = tan(pi);   /* z = 0. The inexact exception is raised. */
z = tan(pi/2); /* z = +INFINITY. The inexact exception is
                  raised. */
z = tan(pi/4); /* z = 1. The inexact exception is raised. */
```

acos

You can use the `acos` function to compute the arc cosine of a real number between –1 and +1.

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

`x` Any floating-point number in the range $-1 \le x \le 1$.

DESCRIPTION

The `acos` function returns the arc cosine of its argument x. The return value is expressed in radians in the range [0, π].

$$\operatorname{acos}(x) = \arccos(x) = y \quad \text{such that} \quad \cos(y) = x \ \text{ for } -1 \le x \le 1$$

The `cos` function performs the inverse operation $(\cos(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\mathrm{acos}(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x other than 1)
- invalid (if $|x|>1$)

SPECIAL CASES

Table 10-23 shows the results when the argument to the `acos` function is a zero, a NaN, or an Infinity, plus other special cases for the `acos` function.

Table 10-23 Special cases for the `acos` function

Operation	Result	Exceptions raised
$\mathrm{acos}(x)$ for $|x| > 1$	NaN	Invalid
$\mathrm{acos}(-1)$	π	Inexact
$\mathrm{acos}(+1)$	+0	None
$\mathrm{acos}(+0)$	$\pi/2$	Inexact
$\mathrm{acos}(-0)$	$\pi/2$	Inexact
$\mathrm{acos}(\mathrm{NaN})$	NaN	None*
$\mathrm{acos}(+\infty)$	NaN	Invalid
$\mathrm{acos}(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = acos(1.0);    /* z = arccos (1) = 0.0 */
z = acos(-1.0);   /* z = arccos (-1) = π. The inexact exception is
                     raised. */
```

asin

You can use the `asin` function to compute the arc sine of a real number between –1 and 1.

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

`x` Any floating-point number in the range $-1 \le x \le 1$.

DESCRIPTION

The `asin` function returns the arc sine of its argument. The return value is expressed in radians in the range $[-\pi/2, +\pi/2]$. This function is antisymmetric.

$$\operatorname{asin}(x) = \arcsin(x) = y \quad \text{such that} \quad \sin(y) = x \quad \text{for } -1 \le x \le 1$$

The `sin` function performs the inverse operation $(\sin(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\operatorname{asin}(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- invalid (if $|x| > 1$)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-24 shows the results when the argument to the `asin` function is a zero, a NaN, or an Infinity, plus other special cases for the `asin` function.

Table 10-24 Special cases for the `asin` function

Operation	Resul t	Exceptions raised
$\operatorname{asin}(x)$ for $\lvert x\rvert > 1$	NaN	Invalid
$\operatorname{asin}(-1)$	$-\pi/2$	Inexact
$\operatorname{asin}(+1)$	$\pi/2$	Inexact
$\operatorname{asin}(+0)$	+0	None
$\operatorname{asin}(-0)$	–0	None
$\operatorname{asin}(\mathrm{NaN})$	NaN	None*
$\operatorname{asin}(+\infty)$	NaN	Invalid
$\operatorname{asin}(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = asin(1.0);     /* z = arcsin 1 = π/2. The inexact exception is
                      raised. */
z = asin(-1.0);    /* z = arcsin -1 = -π/2. The inexact exception
                      is raised. */
```

atan

You can use the `atan` function to compute the arc tangent of a real number.

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

`x` Any floating-point number.

DESCRIPTION

The `atan` function returns the arc tangent of its argument. The return value is expressed in radians in the range $[-\pi/2, +\pi/2]$. This function is antisymmetric.

$$\operatorname{atan}(x) = \arctan(x) = y \quad \text{such that} \quad \tan(y) = x \quad \text{for all } x$$

The `tan` function performs the inverse operation $(\tan(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\operatorname{atan}(x)$ might raise one of the following exceptions:

- inexact (for all nonzero values of x)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-25 shows the results when the argument to the `atan` function is a zero, a NaN, or an Infinity.

Table 10-25 Special cases for the `atan` function

Operation	Result	Exceptions raised
$\operatorname{atan}(+0)$	+0	None
$\operatorname{atan}(-0)$	–0	None
$\operatorname{atan}(\mathrm{NaN})$	NaN	None*
$\operatorname{atan}(+\infty)$	$+\pi/2$	Inexact
$\operatorname{atan}(-\infty)$	$-\pi/2$	Inexact

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = atan(1.0);    /* z = arctan 1 = π/4 */
z = atan(–1.0);   /* z = arctan –1 = –π/4. The inexact exception
                     is raised. */
```

atan2

You can use the `atan2` function to compute the arc tangent of a real number divided by another real number.

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

`y`	Any floating-point number.
`x`	Any floating-point number.

DESCRIPTION

The `atan2` function returns the arc tangent of its first argument divided by its second argument. The return value is expressed in radians in the range [–π, +π], using the signs of its operands to determine the quadrant.

$$\mathrm{atan2}(y, x) = \arctan(y/x) = z \quad \text{such that} \quad \tan(z) = y/x$$

EXCEPTIONS

When x and y are finite and nonzero, the result of $\mathrm{atan2}(y, x)$ might raise one of the following exceptions:

- inexact (if either x or y is any finite, nonzero value)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-26 shows the results when one of the arguments to the `atan2` function is a zero, a NaN, or an Infinity. In this table, x and y are finite, nonzero floating-point numbers.

Table 10-26 Special cases for the `atan2` function

Operation	Result	Exceptions raised
$\text{atan2}(+0, x)$	$+0 \quad x > 0$	None
	$+\pi \quad x < 0$	None
$\text{atan2}(y, +0)$	$+\pi/2 \quad y > 0$	None
	$-\pi/2 \quad y < 0$	None
$\text{atan2}(\pm 0, +0)$	± 0	None
$\text{atan2}(-0, x)$	$-0 \quad x > 0$	Inexact
	$-\pi \quad x < 0$	Inexact
$\text{atan2}(y, -0)$	$+\pi/2 \quad y > 0$	None
	$-\pi/2 \quad y < 0$	None
$\text{atan2}(\pm 0, -0)$	$\pm\pi$	Inexact
$\text{atan2}(\text{NaN}, x)$	NaN*	None†
$\text{atan2}(y, \text{NaN})$	NaN	None†
$\text{atan2}(+\infty, x)$	$\pi/2$	Inexact
$\text{atan2}(y, +\infty)$	± 0	None
$\text{atan2}(\pm\infty, +\infty)$	$\pm 3\pi/4$	Inexact
$\text{atan2}(-\infty, x)$	$-\pi/2$	Inexact
$\text{atan2}(y, -\infty)$	$\pm\pi$	None
$\text{atan2}(\pm\infty, -\infty)$	$\pm 3\pi/4$	Inexact

* If both arguments are NaNs, it is undefined which one `atan2` returns.
† If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = atan2(1.0, 1.0); /* z = arctan 1/1 = arctan 1 = π/4. The
                        inexact exception is raised. */
z = atan2(3.5, 0.0); /* z = arctan 3.5/0 = arctan ∞ = π/2 */
```

Hyperbolic Functions

MathLib provides hyperbolic and inverse hyperbolic functions.

$\cosh(x)$	Hyperbolic cosine of x.
$\sinh(x)$	Hyperbolic sine of x.
$\tanh(x)$	Hyperbolic tangent of x.
$\mathrm{acosh}(x)$	Inverse hyperbolic cosine of x.
$\mathrm{asinh}(x)$	Inverse hyperbolic sine of x.
$\mathrm{atanh}(x)$	Inverse hyperbolic tangent of x.

These functions are based on other transcendental functions and defer most exception generation to the core functions they use.

cosh

You can use the `cosh` function to compute the hyperbolic cosine of a real number.

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

`x` Any floating-point number.

DESCRIPTION

The `cosh` function returns the hyperbolic cosine of its argument. This function is symmetric.

The `acosh` function performs the inverse operation $(\mathrm{arccosh}(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\cosh(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- overflow (if the result is outside the range of the data type)

SPECIAL CASES

Table 10-27 shows the results when the argument to the `cosh` function is a zero, a NaN, or an Infinity.

Table 10-27 Special cases for the `cosh` function

Operation	Result	Exceptions raised
$\cosh(+0)$	+1	None
$\cosh(-0)$	+1	None
$\cosh(\mathrm{NaN})$	NaN	None*
$\cosh(+\infty)$	$+\infty$	None
$\cosh(-\infty)$	$+\infty$	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = cosh(1.0);    /* z ≈ 1.54308. The inexact exception is
                     raised. */
z = cosh(-1.0);   /* z ≈ 1.54308. The inexact exception is
                     raised. */
```

sinh

You can use the `sinh` function to compute the hyperbolic sine of a real number.

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

`x` Any floating-point number.

DESCRIPTION

The `sinh` function returns the hyperbolic sine of its argument. This function is antisymmetric.

The `asinh` function performs the inverse operation $(\operatorname{arcsinh}(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\sinh(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- overflow (if the result is outside the range of the data type)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-28 shows the results when the argument to the `sinh` function is a zero, a NaN, or an Infinity.

Table 10-28 Special cases for the `sinh` function

Operation	Result	Exceptions raised
$\sinh(+0)$	+0	None
$\sinh(-0)$	–0	None
$\sinh(\mathrm{NaN})$	NaN	None*
$\sinh(+\infty)$	$+\infty$	None
$\sinh(-\infty)$	$-\infty$	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
sinh(1.0);  /* z ≈ 1.175201. The inexact exception is raised. */
sinh(–1.0); /* z ≈ –1.175201. The inexact exception is raised. */
```

tanh

You can use the `tanh` function to compute the hyperbolic tangent of a real number.

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

`x` Any floating-point number.

DESCRIPTION

The `tanh` function returns the hyperbolic tangent of its argument. The return value is in the range [–1, +1]. This function is antisymmetric.

The `atanh` function performs the inverse operation $(\operatorname{arctanh}(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\tanh(x)$ raises the following exception:

- inexact (for all finite, nonzero values of x)

SPECIAL CASES

Table 10-29 shows the results when the argument to the `tanh` function is a zero, a NaN, or an Infinity.

Table 10-29 Special cases for the `tanh` function

Operation	Result	Exceptions raised
$\tanh(+0)$	+0	None
$\tanh(-0)$	–0	None
$\tanh(\mathrm{NaN})$	NaN	None*
$\tanh(+\infty)$	+1	None
$\tanh(-\infty)$	–1	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = tanh(1.0);     /* z ≈ 0.761594. The inexact exception is
                      raised. */
z = tanh(-1.0);    /* z ≈ 0.761594. The inexact exception is
                      raised. */
```

acosh

You can use the `acosh` function to compute the inverse hyperbolic cosine of a real number.

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

`x` Any floating-point number in the range $1 \le x \le +\infty$.

DESCRIPTION

The `acosh` function returns the inverse hyperbolic cosine of its argument. This function is antisymmetric.

$$\operatorname{acosh}(x) = \operatorname{arccosh} x = y \quad \text{such that } \cosh y = x$$

The `cosh` function performs the inverse operation $(\cosh(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\operatorname{acosh}(x)$ might raise one of the following exceptions:

- inexact (for all finite values of $x > 1$)
- invalid (if $x < 1$)

SPECIAL CASES

Table 10-30 shows the results when the argument to the `acosh` function is a zero, a NaN, or an Infinity, plus other special cases for the `acosh` function.

Table 10-30 Special cases for the `acosh` function

Operation	Result	Exceptions raised
$\operatorname{acosh}(x)$ for $x < 1$	NaN	Invalid
$\operatorname{acosh}(1)$	+0	None
$\operatorname{acosh}(+0)$	NaN	Invalid
$\operatorname{acosh}(-0)$	NaN	Invalid
$\operatorname{acosh}(\mathrm{NaN})$	NaN	None*
$\operatorname{acosh}(+\infty)$	+∞	None
$\operatorname{acosh}(-\infty)$	NaN	Invalid

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = acosh(1.0);   /* z = +0 */
z = acosh(0.0);   /* z = NAN. The invalid exception is raised. */
```

asinh

You can use the `asinh` function to compute the inverse hyperbolic sine of a real number.

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

`x` Any floating-point number.

DESCRIPTION

The `asinh` function returns the inverse hyperbolic sine of its argument. This function is antisymmetric.

$$\operatorname{asinh}(x) = \operatorname{arcsinh} x = y \quad \text{such that } \sinh y = x$$

The `sinh` function performs the inverse operation $(\sinh(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\operatorname{asinh}(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)

SPECIAL CASES

Table 10-31 shows the results when the argument to the `asinh` function is a zero, a NaN, or an Infinity.

Table 10-31 Special cases for the `asinh` function

Operation	Result	Exceptions raised
$\operatorname{asinh}(+0)$	+0	None
$\operatorname{asinh}(-0)$	–0	None
$\operatorname{asinh}(\mathrm{NaN})$	NaN	None*
$\operatorname{asinh}(+\infty)$	+∞	None
$\operatorname{asinh}(-\infty)$	–∞	None

* If the NaN is a signaling NaN, the invalid exception is raised.

EXAMPLES

```
z = asinh(1.0);   /* z ≈ 0.881374. The inexact exception is
                     raised. */
z = asinh(-1.0);  /* z ≈ 0.881374. The inexact exception is
                     raised. */
```

atanh

You can use the `atanh` function to perform the inverse hyperbolic tangent of a real number.

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

`x` Any floating-point number in the range $-1 \le x \le 1$.

DESCRIPTION

The `atanh` function returns the inverse hyperbolic tangent of its argument. This function is antisymmetric.

$$\operatorname{atanh}(x) = \operatorname{arctanh} x = y \quad \text{such that } \tanh y = x$$

The `tanh` function performs the inverse operation $(\tanh(y))$.

EXCEPTIONS

When x is finite and nonzero, the result of $\operatorname{atanh}(x)$ might raise one of the following exceptions:

- inexact (for all finite, nonzero values of x other than +1 and –1)
- invalid (if $|x| > 1$)
- underflow (if the result is inexact and must be represented as a denormalized number or 0)