z/OS C/C++ Run-Time Library Reference - Decimal Floating-Point Supplement

z/OS C/C++ Run-Time Library Reference - Decimal Floating-Point Supplement

Contents

Chapter 1. Header Files

float.h

The float.h header file contains definitions of constants listed in ANSI 2.2.4.2.2, that describe the characteristics of the internal representations of the three floating-point data types, float, double, and long double. The definitions are:

Table 1. Definitions in float.h

Constant	Description
FLT_MAXDIG10 DBL_MAXDIG10 LDBL_MAXDIG10	The number of base 10 digits required to ensure that values which differ by only one smallest unit in the last place (ulp) are always differentiated.

math.h

__STDC_WANT_DEC_FP__

acosd32()	acosd64()	acosd128()	acoshd32()	acoshd64()
acoshd128()	asind32()	asind64()	asind128()	asinhd32()
asinhd64()	asinhd128()	atand32()	atand64()	atand128()
atan2d32()	atan2d64()	atan2d128()	atanhd32()	atanhd64()
atanhd128()	__atanpid32()	__atanpid64()	__atanpid128()	ceild32()
ceild64()	ceild128()	copysignd32()	copysignd64()	copysignd128()
cosd32()	cosd64()	cosd128()	coshd32()	coshd64()
coshd128()	__cospid32()	__cospid64()	__cospid128()	erfd32()
erfd64()	erfd128()	erfcd32()	erfcd64()	erfcd128()
expd32()	expd64()	expd128()	fabsd32()	fabsd64()
fabsd128()	fdimd32()	fdimd64()	fdimd128()	floord32()
floord64()	floord128()	fmaxd32()	fmaxd64()	fmaxd128()
fmind32()	fmind64()	fmind128()	frexpd32()	frexpd64()
frexpd128()	ilogbd32()	ilogbd64()	ilogbd128()	ldexpd32()
ldexpd64()	ldexpd128()	lgammad32()	lgammad64()	lgammad128()
llrintd32()	llrintd64()	llrintd128()	llroundd32()	llroundd64()
llroundd128()	logd32()	logd64()	logd128()	log10d32()
log10d64()	log10d128()	logbd32()	logbd64()	logbd128()
lrintd32()	lrintd64()	lrintd128()	lroundd32()	lroundd64()
lroundd128()	modfd32()	modfd64()	modfd128()	nand32()
nand64()	nand128()	nearbyintd32()	nearbyintd64()	nearbyintd128()
nextafterd32()	nextafterd64()	nextafterd128()	nexttowardd32()	nexttowardd64()
nexttowardd128()	powd32()	powd64()	powd128()	quantized32()
quantized64()	quantized128()	remainderd32()	remainderd64()	remainderd128()
rintd32()	rintd64()	rintd128()	roundd32()	roundd64()
roundd128()	samequantumd32()	samequantumd64()	samequantumd128()	scalblnd32()
scalblnd64()	scalblnd128()	scalbnd32()	scalbnd64()	scalbnd128()
sind32()	sind64()	sind128()	sinhd32()	sinhd64()
sinhd128()	__sinpid32()	__sinpid64()	__sinpid128()	sqrtd32()
sqrtd64()	sqrtd128()	tand32()	tand64()	tand128()
tanhd32()	tanhd64()	tanhd128()	tgammad32()	tgammad64()
tgammad128()	truncd32()	truncd64()	truncd128()	

For C++ applications, the following functions are overloaded for _Decimal32, _Decimal64, and _Decimal128:

abs()	acos()	acosh()	asin()	asinh()
atan()	atan2()	atanh()	ceil()	copysign()
cos()	cosh()	erf()	erfc()	exp()
fabs()	fdim()	floor()	fmax()	fmin()
frexp()	ilogb()	ldexp()	lgamma()	llrint()
llround()	log()	log10()	logb()	lrint()
lround()	modf()	nearbyint()	nextafter()	nexttoward()
pow()	remainder()	rint()	round()	scalbn()
scalbln()	sin()	sinh()	sqrt()	tan()
tanh()	tgamma()	trunc()		

Chapter 2. Library Functions

acosd32(), acosd64(), acosd128() - Calculate Arccosine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  acosd32(_Decimal32 x);
_Decimal64  acosd64(_Decimal64 x);
_Decimal128 acosd128(_Decimal128 x);
_Decimal32  acos(_Decimal32 x);      /* C++ only */
_Decimal64  acos(_Decimal64 x);      /* C++ only */
_Decimal128 acos(_Decimal128 x);     /* C++ only */
```

General Description

Calculates the arccosine of *x*, expressed in radians, in the range 0 to pi.

The value of *x* must be between -1 and 1 inclusive.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, the function returns the arccosine of the argument x.

If *x* is less than -1 or greater than 1, the function sets errno to EDOM and returns NaNQ. No other errors will occur.

Example

CELEBA11

```
/* CELEBA11

   The example illustrates the acosd32() function.

   This example prompts for a value for x.
   It prints an error message if x is greater than 1 or
   less than -1; otherwise, it assigns the arccosine of
   x to y.

*/

#define __STDC_WANT_DEC_FP__
#include <stdio.h>
```

```
#include <stdlib.h>
#include <math.h>
#define MAX  1.0DF
#define MIN -1.0DF

int main(void)
{
   _Decimal32 x, y;

   printf( "Enter x\n" );
   scanf( "%Hf", &x );

   /* Output error if not in range */
   if ( x > MAX )
      printf( "Error: %f too large for acosd32\n", x );
   else if ( x < MIN )
      printf( "Error: %f too small for acosd32\n", x );
   else {
      y = acosd32( x );
      printf( "acosd32( %Hf ) = %Hf\n", x, y );
   }
}
```

Related Information

- math.h
- acoshd32(),acoshd64(),acoshd128() v Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

acoshd32(), acoshd64(), acoshd128() - Calculate Hyperbolic Arccosine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  acoshd32(_Decimal32 x);
_Decimal64  acoshd64(_Decimal64 x);
_Decimal128 acoshd128(_Decimal128 x);
_Decimal32  acosh(_Decimal32 x);      /* C++ only */
_Decimal64  acosh(_Decimal64 x);      /* C++ only */
_Decimal128 acosh(_Decimal128 x);     /* C++ only */
```

General Description

The acosh functions compute the (nonnegative) arc hyperbolic cosine of *x*. A domain error occurs for arguments less than 1.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, the function returns the hyperbolic arccosine of its argument x.

If *x* is less than 1.0, the function sets errno to EDOM and returns NaNQ.

Example

CELEBA12

```
/* CELEBA12

   This example illustrates the acoshd64() function.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal64 x, y;

   x = 100.0DD;
   y = acoshd64(x);

   printf("acoshd64(%Df) = %Df\n", x, y);
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

asind32(), asind64(), asind128() - Calculate Arcsine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  asind32(_Decimal32 x);
_Decimal64  asind64(_Decimal64 x);
_Decimal128 asind128(_Decimal128 x);
_Decimal32  asin(_Decimal32 x);      /* C++ only */
_Decimal64  asin(_Decimal64 x);      /* C++ only */
_Decimal128 asin(_Decimal128 x);     /* C++ only */
```

General Description

Calculates the arcsine of *x*, in the range -pi/2 to pi/2 radians.

The value of *x* must be between -1 and 1.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, the function returns the arcsine of its argument x.

If *x* is less than -1 or greater than 1, the function sets errno to EDOM and returns NaNQ. No other errors will occur.

Example

CELEBA13

```
/* CELEBA13

   This example illustrates the asind128() function.

   This example prompts for a value for x.
   It prints an error message if x is greater than 1 or
   less than -1; otherwise, it assigns the arcsine of
   x to y.

*/

#define  __STDC_WANT_DEC_FP__
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX  1.0DL
#define MIN -1.0DL

int main(void)
```

```
{
   _Decimal128 x, y;

   printf( "Enter x\n" );
   scanf( "%DDf", &x );

   /* Output error if not in range */
   if ( x > MAX )
      printf( "Error: %f too large for asind128\n", x );
   else if ( x < MIN )
      printf( "Error: %f too small for asind128\n", x );
   else {
      y = asind128( x );
      printf( "asind128( %DDf ) = %DDf\n", x, y );
   }
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate HyperbolicArctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanh64(), tanhd128() — Calculate Hyperbolic Tangent

asinhd32(), asinhd64(), asinhd128() - Calculate Hyperbolic Arcsine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  asinhd32(_Decimal32 x);
_Decimal64  asinhd64(_Decimal64 x);
_Decimal128 asinhd128(_Decimal128 x);
_Decimal32  asinh(_Decimal32 x);     /* C++ only */
_Decimal64  asinh(_Decimal64 x);     /* C++ only */
_Decimal128 asinh(_Decimal128 x);    /* C++ only */
```

General Description

The asinhd() functions return the hyperbolic arcsine of its argument *x*.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

asinhd() returns the hyperbolic arcsine of its argument *x*. The function is always successful.

Example

CELEBA14

```
/* CELEBA14

   This example illustrates the asinhd32() function.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal32 x, y;

   x = 1.0DF;
   y = asinhd32(x);

   printf("asinhd32(%Hf) = %Hf\n", x, y);
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine

- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() - Calculate Arctangent

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  atand32(_Decimal32 x);
_Decimal64  atand64(_Decimal64 x);
_Decimal128 atand128(_Decimal128 x);
_Decimal32  atan(_Decimal32 x);     /* C++ only */
_Decimal64  atan(_Decimal64 x);     /* C++ only */
_Decimal128 atan(_Decimal128 x);    /* C++ only */

_Decimal32  atan2d32(_Decimal32 y, _Decimal32 x);
_Decimal64  atan2d64(_Decimal64 y, _Decimal64 x);
_Decimal128 atan2d128(_Decimal128 y, _Decimal128 x);
_Decimal32  atan2(_Decimal32 y, _Decimal32 x);   /* C++ only */
_Decimal64  atan2(_Decimal64 y, _Decimal64 x);   /* C++ only */
_Decimal128 atan2(_Decimal128 y, _Decimal128 x); /* C++ only */
```

General Description

The atan() and atan2() functions calculate the arctangent of x and *y/x*, respectively.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

Returns a value in the range -pi/2 to pi/2 radians.

If both arguments of atan2() are zero, the function sets errno to EDOM and returns 0. No other errors will occur.

Example

CELEBA15

```
/* CELEBA15

   This example illustrates the atand64() and atan2d64() functions.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
    _Decimal64 a,b,c,d;
```

```
    c = 0.45DD;
    d = 0.23DD;

    a = atand64(c);
    b = atan2d64(c,d);

    printf("atand64( %Df ) = %Df\n", c, a);
    printf("atan2d64( %Df, %Df ) = %Df\n", c, d, b);
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

atanhd32(), atanhd64(), atanhd128() - Calculate Hyperbolic Arctangent

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  atanhd32(_Decimal32 x);
_Decimal64  atanhd64(_Decimal64 x);
_Decimal128 atanhd128(_Decimal128 x);
_Decimal32  atanh(_Decimal32 x);      /* C++ only */
_Decimal64  atanh(_Decimal64 x);      /* C++ only */
_Decimal128 atanh(_Decimal128 x);     /* C++ only */
```

General Description

The atanh() function returns the hyperbolic arctangent of its argument *x*.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, the function returns the hyperbolic arctangent of its argument x.

If the absolute value of *x* is greater than 1.0, atanh() sets errno to EDOM and returns NaNQ. If the value of *x* is equal to 1.0, the function sets errno to ERANGE and returns +HUGE_VAL_D32, +HUGE_VAL_D64 or +HUGE_VAL_D128.

Example

CELEBA16

```
/* CELEBA16

   This example illustrates the atanhd32() function.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal32 x, y;

   x = 0.5DF;
   y = atanhd32(x);

   printf("atanhd32(%Hf) = %Hf\n", x, y);
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

__atanpid32(), __atanpid64(), __atanpid128() - Calculate Arctangent(x)/pi

Standards

Standards/Extensions	C or C++	Dependencies
Language Environment	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  __atanpid32(_Decimal32 x);
_Decimal64  __atanpid64(_Decimal64 x);
_Decimal128 __atanpid128(_Decimal128 x);
```

General Description

Calculates the value of arctangent(*x*)/pi.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

Returns the calculated value expressed in radians.

Example

CELEBA17

```
/* CELEBA17

   This example illustrates the __atanpid64() function.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal64 x, y;

   x = 5.0DD;
   y = __atanpid64(x);

   printf("__atanpid64(%Df) = %Df\n", x, y);
}
```

Related Information

- math.h
- __cospid32(), __cospid64(), __cospid128() — Calculate Cosine of pi *
- __sinpid32(), __sinpid64(), __sinpid128() — Calculate Sine of pi *

coshd32(), coshd64(), coshd128() - Calculate Hyperbolic Cosine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  coshd32(_Decimal32 x);
_Decimal64  coshd64(_Decimal64 x);
_Decimal128 coshd128(_Decimal128 x);
_Decimal32  cosh(_Decimal32 x);     /* C++ only */
_Decimal64  cosh(_Decimal64 x);     /* C++ only */
_Decimal128 cosh(_Decimal128 x);    /* C++ only */
```

General Description

Calculates the hyperbolic cosine of *x*. The value *x* is expressed in radians.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If the result overflows, the function returns +HUGE_VAL_D32, +HUGE_VAL_D64 or +HUGE_VAL_D128 and sets errno to ERANGE.

Example

CELEBC51

```
/* CELEBC51

   This example illustrates the coshd128() function.

   This example calculates y to be the hyperbolic cosine of x.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal128 x, y;

   x = 7.2DL;
   y = coshd128(x);

   printf("coshd128( %DDf ) = %DDf\n", x, y);
}
```

Related Information

- math.h
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

erfd32(), erfd64(), erfd128(), erfcd32(), erfcd64(), erfcd128() - Calculate Error and Complementary Error Functions

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  erfd32(_Decimal32 x);
_Decimal64  erfd64(_Decimal64 x);
_Decimal128 erfd128(_Decimal128 x);
_Decimal32  erf(_Decimal32 x);      /* C++ only */
_Decimal64  erf(_Decimal64 x);      /* C++ only */
_Decimal128 erf(_Decimal128 x);     /* C++ only */

_Decimal32  erfcd32(_Decimal32 x);
_Decimal64  erfcd64(_Decimal64 x);
_Decimal128 erfcd128(_Decimal128 x);
_Decimal32  erfc(_Decimal32 x);     /* C++ only */
_Decimal64  erfc(_Decimal64 x);     /* C++ only */
_Decimal128 erfc(_Decimal128 x);    /* C++ only */
```

General Description

Calculates the error and complementary error functions:

Because the erfc() function calculates the value of 1.0 - erf(x), it is used in place of erf() for large values of *x*.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

erf() and erfc() are always successful.

Example

CELEBE12

```
/* CELEBE12

   This example illustrates the erfd32() and erfcd32() functions.

*/

#define __STDC_WANT_DEC_FP__
#include <stdio.h>
#include <math.h>

_Decimal32 smallx, largex, value;
```

```
int main(void)
{
   smallx = 0.1DF;
   largex = 10.0DF;

   value = erfd32(smallx);
   printf("Error value for 0.1: %Hf\n", value);

   value = erfcd32(largex);
   printf("Error value for 10.0: %He\n", value);
}
```

Related Information

- math.h

lgammad32(), lgammad64(), lgammad128() - Log Gamma Function

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  lgammad32(_Decimal32 x);
_Decimal64  lgammad64(_Decimal64 x);
_Decimal128 lgammad128(_Decimal128 x);
_Decimal32  lgamma(_Decimal32 x);     /* C++ only */
_Decimal64  lgamma(_Decimal64 x);     /* C++ only */
_Decimal128 lgamma(_Decimal128 x);    /* C++ only */
```

General Description

The lgamma() function computes the

$$\log_e | \Gamma(x) |$$

is defined as

$$\int_0^\infty e^{-t}\ t^{(x-1)}\ dt$$

The sign of

$$\Gamma(x)$$

is returned in the external integer signgam. The argument *x* may not be a non-positive integer.

In a multithreaded process, each thread has its own instance of the *signgam* variable. Threads access their instances of the variable by calling the __signgam() function. The math.h header redefines the string *signgam* to an invocation of the __siggam function. The actual signgam external variable is used to store the signgam value for the IPT.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, lgamma() returns the above function of its argument.

lgamma() will fail under the following conditions:

- If the result overflows, the function will return +HUGE_VAL_D32, +HUGE_VAL_D64 or +HUGE_VAL_D128 and set errno to ERANGE.
- If *x* is a non-positive integer, lgamma() returns +HUGE_VAL_D32, +HUGE_VAL_D64 or +HUGE_VAL_D128 and sets errno to ERANGE.

Example

CELEBL26

```
/* CELEBL26

   This exaxmple illustrates the lgammad64() function.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>
int main(void)
{
   _Decimal64 x, y;

   x = 42.0DD;
   y = lgammad64(x);

   printf ("lgammad64(%Df) = %Df\n", x, y);
}
```

Related Information

- math.h
- expd32(), expd64(), expd128() — Calculate Exponential Function
- isnan() — Test for Nan
- __signgam() — Return signgam Reference

remainderd32(), remainderd64(), remainderd128() - Computes the remainder x REM y

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  remainderd32(_Decimal32 x, _Decimal32 y);
_Decimal64  remainderd64(_Decimal64 x, _Decimal64 y);
_Decimal128 remainderd128(_Decimal128 x, _Decimal128 y);
_Decimal32  remainder(_Decimal32 x,
                      _Decimal32 y);     /* C++ only */
_Decimal64  remainder(_Decimal64 x,
                      _Decimal64 y);     /* C++ only */
_Decimal128 remainder(_Decimal128 x,
                      _Decimal128 y);    /* C++ only */
```

General Description

The remainder() function returns the decimal floating-point remainder when *y* is nonzero and following the relation

The value *n* is the integral value nearest the exact value *x/y* and when then the value of n is even.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, remainder() returns the remainder of the division of *x* by *y*.

If *y* is zero, remainder() returns NaNQ and sets errno to EDOM.

Example

CELEBR23

```
/* CELEBR23

   This example illustrates the remainderd32() function.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

void main() {
  _Decimal32 number1=3.0DF, number2=3.5DF;
```

```
  printf("Illustrates the remainderd32() function\n");

  printf("remainderd32(%.2Hf,%.2Hf)=%.2Hf\n",number1,number2,remainderd32(number1,number2));
  number1=1.0DF; number2=2.0DF;
  printf("remainderd32(%.2Hf,%.2Hf)=%.2Hf\n",number1,number2,remainderd32(number1,number2));
  number1=1.0DF; number2=0.0DF;
  printf("remainderd32(%.2Hf,%.2Hf)=%.2Hf\n",number1,number2,remainderd32(number1,number2));
}
```

Related Information

- math.h

sinhd32(), sinhd64(), sinhd128() - Calculate Hyperbolic Sine

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  sinhd32(_Decimal32 x);
_Decimal64  sinhd64(_Decimal64 x);
_Decimal128 sinhd128(_Decimal128 x);
_Decimal32  sinh(_Decimal32 x);     /* C++ only */
_Decimal64  sinh(_Decimal64 x);     /* C++ only */
_Decimal128 sinh(_Decimal128 x);    /* C++ only */
```

General Description

Calculates the hyperbolic sine of *x*, with *x* expressed in radians.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

If successful, the function returns the hyperbolic sine of *x* with x expressed in radians.

If the result would overflow, the function returns ±HUGE_VAL_D32, ±HUGE_VAL_D64, or ±HUGE_VAL_D128 according to the value of *x*, and sets errno to ERANGE. No other errors can occur.

Example

CELEBS75

```
/* CELEBS75

   This example illustrates the sinhd64() function.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal64 pi, x, y;

   pi = 3.1415926535DD;
   x = pi/2.0DD;
```

```
    y = sinhd64(x);

    printf("sinhd64( %Df ) = %Df\n", x, y);
}
```

Related Information

- math.h
- atanh(), atanhf(), atanhl() — Calculate Hyperbolic Arctangent
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- tand32(), tand64(), tand128() — Calculate Tangent
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

tand32(), tand64(), tand128() - Calculate Tangent

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  tand32(_Decimal32 x);
_Decimal64  tand64(_Decimal64 x);
_Decimal128 tand128(_Decimal128 x);
_Decimal32  tan(_Decimal32 x);     /* C++ only */
_Decimal64  tan(_Decimal64 x);     /* C++ only */
_Decimal128 tan(_Decimal128 x);    /* C++ only */
```

General Description

Calculates the tangent of *x*, where *x* is expressed in radians. If x is large, a partial loss of significance in the result can occur.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

Returns the calculated tangent of *x*.

If the correct value would cause underflow, zero is returned. If the result overflows, ±HUGE_VAL_D32, ±HUGE_VAL_D64, or ±HUGE_VAL_D128 is returned. For both underflow and overflow, the value ERANGE is stored in errno.

Example

CELEBT22

```
/* CELEBT22

   This example illustrates the tand64() function.

   This example computes x as the tangent of PI/4.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal64 pi, x;

   pi = 3.1415926DD;
```

```
    x = tand64(pi/4.0DD);

    printf("tand64( %Df ) is %Df\n", pi/4.0DD, x);
}
```

Related Information

- math.h
- atanh(), atanhf(), atanhl() — Calculate Hyperbolic Arctangent
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine
- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tanhd32(), tanhd64(), tanhd128() — Calculate Hyperbolic Tangent

tanhd32(), tanhd64(), tanhd128() - Calculate Hyperbolic Tangent

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  tanhd32(_Decimal32 x);
_Decimal64  tanhd64(_Decimal64 x);
_Decimal128 tanhd128(_Decimal128 x);
_Decimal32  tanh(_Decimal32 x);     /* C++ only */
_Decimal64  tanh(_Decimal64 x);     /* C++ only */
_Decimal128 tanh(_Decimal128 x);    /* C++ only */
```

General Description

Calculates the hyperbolic tangent of *x*, where *x* is expressed in radians.

Returned Value

Returns the calculated value of the hyperbolic tangent of *x*.

If the result underflows, the function returns 0 and sets the errno to ERANGE.

Example

CELEBT23

```
/* CELEBT23

   This example illustrates the tanhd64() function.

   This example computes x as the hyperbolic tangent of PI/4.

*/

#define __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal64 pi, x;

   pi = 3.1415926DD;
   x = tanhd64(pi/4.0DD);

   printf("tanhd64( %Df ) = %Df\n", pi/4.0DD, x);
}
```

Related Information

- math.h
- atanh(), atanhf(), atanhl() — Calculate Hyperbolic Arctangent
- acosd32(), acosd64(), acosd128() — Calculate Arccosine
- acoshd32(), acoshd64(), acoshd128() — Calculate Hyperbolic Arccosine
- asind32(), asind64(), asind128() — Calculate Arcsine

- asinhd32(), asinhd64(), asinhd128() — Calculate Hyperbolic Arcsine
- atand32(), atand64(), atand128(), atan2d32(), atan2d64(), atan2d128() — Calculate Arctangent
- atanhd32(), atanhd64(), atanhd128() — Calculate Hyperbolic Arctangent
- cosd32(), cosd64(), cosd128() — Calculate Cosine
- coshd32(), coshd64(), coshd128() — Calculate Hyperbolic Cosine
- sind32(), sind64(), sind128() — Calculate Sine
- sinhd32(), sinhd64(), sinhd128() — Calculate Hyperbolic Sine
- tand32(), tand64(), tand128() — Calculate Tangent

tgammad32(), tgammad64(), tgammad128() - Calculate Gamma Function

Standards

Standards/Extensions	C or C++	Dependencies
C/C++ DFP	both	z/OS V1.10

Format

```
#define __STDC_WANT_DEC_FP__
#include <math.h>

_Decimal32  tgammad32(_Decimal32 x);
_Decimal64  tgammad64(_Decimal64 x);
_Decimal128 tgammad128(_Decimal128 x);
_Decimal32  tgamma(_Decimal32 x);     /* C++ only */
_Decimal64  tgamma(_Decimal64 x);     /* C++ only */
_Decimal128 tgamma(_Decimal128 x);    /* C++ only */
```

General Description

The tgamma() functions compute the gamma function of *x*.

These functions work in IEEE decimal floating-point format. See for more information.

Note: To use IEEE decimal floating-point, the hardware must have the Decimal Floating-Point Facility installed.

Returned Value

The tgamma functions return G(*x*).

A domain error occurs if *x* is a negative integer or when *x* is zero and the result cannot be represented. A range error occurs if the magnitude of *x* is too large or too small.

Example

CELEBT24

```
/* CELEBT24

   This example illustrates the tgammad128() function.

*/

#define  __STDC_WANT_DEC_FP__
#include <math.h>
#include <stdio.h>

int main(void)
{
   _Decimal128 x, y;

   x = 5.6DL;
   y = tgammad128(x);

   printf("tgammad128(%DDf) = %DDf\n", x, y);
}
```

Related Information

- math.h

Printed in USA