Doc No: WG14 N1051
Date: February 6, 2004
Reply to: P.J. Plauger
pjp@dinkumware.com

(Draft) Technical Report on Special Math Functions

Contents

Chapter 1

General [tr.intro]

This technical report describes extensions to the *C standard library* that is described in the International Standard for the C programming language ISO9899:1999.

This technical report is non-normative. Some of the library components in this technical report may be considered for standardization in a future version of C, but they are not currently part of any C standard. Some of the components in this technical report may never be standardized, and others may be standardized in a substantially changed form.

The goal of this technical report it to build more widespread existing practice for an expanded C standard library. It gives advice on extensions to those vendors who wish to provide them.

1.1 Method of description [tr.description]

The structure of clauses in this technical report, the elements that make up the subclauses, and the editorial conventions used to describe library components, are the same as described in clause 7 of the C standard.

1.2 Conditional inclusion [tr.inclusion]

Vendors should not simply add declarations to standard headers in a way that would be visible to users by default. [*Note:* That would fail to be standard conforming, because the new names could conflict with user macros. *—end note*] Users should be required to take explicit action to have access to library extensions.

It is recommended either that additional declarations in standard headers be protected with a macro that is not defined by default, or else that all extended headers, including both new headers and parallel versions of standard headers with nonstandard declarations, be placed in a separate directory that is not part of the default search path.

Chapter 2

Special math functions [tr.num.sf]

2.1 Additions to header <math.h> synopsis [tr.math.sf.syn]

The Table 2.1 summarizes the functions that are added to header <math.h>. The detailed signatures are given in the synopsis.

Table 2.1: Summary of additions to header <math.h>

Type		Name(s)	
Functions:			
assoc_laguerre	conf_hyperg	ellint_2	legendre
assoc_legendre	cyl_bessel_i	ellint_3	riemann_zeta
beta	cyl_bessel_j	expint	sph_bessel
comp_ellint_1	cyl_bessel_k	hermite	sph_legendre
comp_ellint_2	cyl_neumann	hyperg	sph_neumann
comp_ellint_3	ellint_1	laguerre	

Each of these functions is provided for arguments of type float, double, and long double. The signatures added to header <math.h> are:

```
// associated Laguerre polynomials:
double       assoc_laguerre(unsigned n, unsigned m, double x);
float        assoc_laguerref(unsigned n, unsigned m, float x);
long double  assoc_laguerrel(unsigned n, unsigned m, long double x);

// associated Legendre functions:
double       assoc_legendre(unsigned l, unsigned m, double x)
float        assoc_legendref(unsigned l, unsigned m, float x)
long double  assoc_legendrel(unsigned l, unsigned m, long double x)
```

```
// beta function:
double       beta(double x, double y);
float        betaf(float x, float y);
long double  betal(long double x, long double y);

// (complete) elliptic integral of the first kind:
double       comp_ellint_1(double k);
float        comp_ellint_1f(float k);
long double  comp_ellint_1l(long double k);

// (complete) elliptic integral of the second kind:
double       comp_ellint_2(double k);
float        comp_ellint_2f(float k);
long double  comp_ellint_2l(long double k);

// (complete) elliptic integral of the third kind:
double       comp_ellint_3(double k, double nu);
float        comp_ellint_3f(float k, float nu);
long double  comp_ellint_3l(long double k, long double nu);

// confluent hypergeometric functions:
double       conf_hyperg(double a, double c, double x);
float        conf_hypergf(float a, float c, float x);
long double  conf_hypergl(long double a, long double c, long double x);

// regular modified cylindrical Bessel functions:
double       cyl_bessel_i(double nu, double x);
float        cyl_bessel_if(float nu, float  x);
long double  cyl_bessel_il(long double nu, long double x);

// cylindrical Bessel functions (of the first kind):
double       cyl_bessel_j(double nu, double x);
float        cyl_bessel_jf(float nu, float  x);
long double  cyl_bessel_jl(long double nu, long double x);

// irregular modified cylindrical Bessel functions:
double       cyl_bessel_k(double nu, double x);
float        cyl_bessel_kf(float nu, float  x);
long double  cyl_bessel_kl(long double nu, long double x);

// cylindrical Neumann functions;
// cylindrical Bessel functions (of the second kind):
double       cyl_neumann(double nu, double x);
float        cyl_neumannf(float nu, float  x);
long double  cyl_neumannl(long double nu, long double x);
```

```
// (incomplete) elliptic integral of the first kind:
double       ellint_1(double k, double phi);
float        ellint_1f(float k, float phi);
long double  ellint_1l(long double k, long double phi);

// (incomplete) elliptic integral of the second kind:
double       ellint_2(double k, double phi);
float        ellint_2f(float k, float phi);
long double  ellint_2l(long double k, long double phi);

// (incomplete) elliptic integral of the third kind:
double       ellint_3(double k, double nu, double phi);
float        ellint_3f(float k, float nu, float phi);
long double  ellint_3l(long double k, long double nu, long double phi);

// exponential integral:
double       expint(double x);
float        expintf(float x);
long double  expintl(long double x);

// Hermite polynomials:
double       hermite(unsigned n, double x);
float        hermitef(unsigned n, float x);
long double  hermitel(unsigned n, long double x);

// hypergeometric functions:
double       hyperg(double a, double b, double c,  double x);
float        hypergf(float a, float b, float c,  float x);
long double  hypergl(long double a, long double b, long double c,
                     long double x);

// Laguerre polynomials:
double       laguerre(unsigned n, double x);
float        laguerref(unsigned n, float x);
long double  laguerrel(unsigned n, long double x);

// Legendre polynomials:
double       legendre(unsigned l, double x)
float        legendref(unsigned l, float x)
long double  legendrel(unsigned l, long double x)

// Riemann zeta function:
double       riemann_zeta(double);
float        riemann_zetaf(float);
long double  riemann_zetal(long double);
```

```
// spherical Bessel functions (of the first kind):
double       sph_bessel(unsigned n, double x);
float        sph_besself(unsigned n, float  x);
long double  sph_bessell(unsigned n, long double x);

// spherical associated Legendre functions:
double       sph_legendre(unsigned l, unsigned m, double theta)
float        sph_legendref(unsigned l, unsigned m, float theta)
long double  sph_legendrel(unsigned l, unsigned m, long double theta)

// spherical Neumann functions;
// spherical Bessel functions (of the second kind):
double       sph_neumann(unsigned n, double x);
float        sph_neumannf(unsigned n, float  x);
long double  sph_neumannl(unsigned n, long double x);
```

2.1.1 associated Laguerre polynomials [tr.math.sf.Lnm]

Synopsis

```
double       assoc_laguerre(unsigned n, unsigned m, double x);
float        assoc_laguerref(unsigned n, unsigned m, float x);
long double  assoc_laguerrel(unsigned n, unsigned m, long double x);
```

Description

These functions compute the associated Laguerre polynomials of their respective arguments `n`, `m`, and `x`.

Returns

The `assoc_laguerre` functions return

$$\mathsf{L}_n^m(x) = e^x \frac{\mathsf{d}^m}{\mathsf{d}x^m} \mathsf{L}_n(x) \, .$$

2.1.2 associated Legendre functions [tr.math.sf.Plm]

Synopsis

```
double       assoc_legendre(unsigned l, unsigned m, double x)
float        assoc_legendref(unsigned l, unsigned m, float x)
long double  assoc_legendrel(unsigned l, unsigned m, long double x)
```

Description

These functions compute the associated Legendre functions of their respective arguments `l`, `m`, and `x`. A domain error occurs if `m` is greater than `l`. A domain error may occur if the magnitude of `x` is greater than one.

Returns The `assoc_legendre` functions return

$$\mathsf{P}_l^m(x) = (1-x^2)^{m/2}\,\frac{\mathsf{d}^m}{\mathsf{d}x^m}\,\mathsf{P}_l(x)\ .$$

2.1.3 beta function [tr.math.sf.beta]

Synopsis

```
double       beta(double x, double y);
float        betaf(float x, float y);
long double  betal(long double x, long double y);
```

Description

These functions compute the beta function of their respective arguments `x` and `y`. A domain error may occur (a) if either `x` or `y` is a negative integer, or (b) if either `x` or `y` is zero.

Returns

The `beta` functions return

$$\mathsf{B}(x,y) = \frac{\Gamma(x)\,\Gamma(y)}{\Gamma(x+y)}\ .$$

2.1.4 (complete) elliptic integral of the first kind [tr.math.sf.ellK]

Synopsis

```
double       comp_ellint_1(double k);
float        comp_ellint_1f(float k);
long double  comp_ellint_1l(long double k);
```

Description

These functions compute the complete elliptic integral of the first kind of their respective arguments `k`. A domain error occurs if the magnitude of `k` is greater than one.

Returns

The `comp_ellint_1` functions return

$$\mathsf{K}(k) = \mathsf{F}(k,\pi/2) = \int_0^{\pi/2} \frac{\mathsf{d}\theta}{\sqrt{1-k^2 sin^2\theta}}\ .$$

2.1.5 (complete) elliptic integral of the second kind [tr.math.sf.ellEx]

Synopsis

```
double       comp_ellint_2(double k);
float        comp_ellint_2f(float k);
long double  comp_ellint_2l(long double k);
```

Description

These functions compute the complete elliptic integral of the second kind of their respective arguments `k`. A domain error occurs if the magnitude of `k` is greater than one.

Returns

The `comp_ellint_2` functions return

$$\mathsf{E}(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2\theta}\, \mathsf{d}\theta \ .$$

2.1.6 (complete) elliptic integral of the third kind [tr.math.sf.ellPx]

Synopsis

```
double       comp_ellint_3(double k, double nu);
float        comp_ellint_3f(float k, float nu);
long double  comp_ellint_3l(long double k, long double nu);
```

Description

These functions compute the complete elliptic integral of the third kind of their respective arguments `k` and `nu`. A domain error occurs if the magnitude of `k` is greater than one.

Returns

The `comp_ellint_3` functions return

$$\mathsf{\Pi}(\nu, k, \pi/2) = \int_0^{\pi/2} \frac{\mathsf{d}\theta}{(1 - \nu\, sin^2\theta)\sqrt{1 - k^2 sin^2\theta}} \ .$$

2.1.7 confluent hypergeometric functions [tr.math.sf.conhyp]

Synopsis

```
double       conf_hyperg(double a, double c, double x) ;
float        conf_hypergf(float a, float c, float x) ;
long double  conf_hypergl(long double a, long double c, long double x) ;
```

Description

These functions compute the confluent hypergeometric functions of their respective arguments `a`, `c`, and `x`. A domain error occurs (a) if `c` is a negative integer, or (b) if `c` is zero.

Returns

The `conf_hyperg` functions return

$$\mathsf{F}(a; c; x) = \frac{\Gamma(c)}{\Gamma(a)} \sum_{n=0}^{\infty} \frac{\Gamma(a+n)}{\Gamma(c+n)} \frac{x^n}{n!} \ .$$

2.1.8 regular modified cylindrical Bessel functions [tr.math.sf.I]

Synopsis

```
double       cyl_bessel_i(double nu, double x);
float        cyl_bessel_if(float nu, float  x);
long double  cyl_bessel_il(long double nu, long double x);
```

Description

These functions compute the regular modified cylindrical Bessel functions of their respective arguments `nu` and `x`. A domain error may occur if `x` is less than zero.

Returns

The `cyl_bessel_i` functions return

$$\mathsf{I}_\nu(x) = \mathrm{i}^{-\nu}\mathsf{J}_\nu(\mathrm{i}x) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k!\,\Gamma(\nu+k+1)} \,.$$

2.1.9 cylindrical Bessel functions (of the first kind) [tr.math.sf.J]

Synopsis

```
double       cyl_bessel_j(double nu, double x);
float        cyl_bessel_jf(float nu, float  x);
long double  cyl_bessel_jl(long double nu, long double x);
```

Description

These functions compute the cylindrical Bessel functions of the first kind of their respective arguments `nu` and `x`. A domain error may occur if `x` is less than zero.

Returns

The `cyl_bessel_j` functions return

$$\mathsf{J}_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k(x/2)^{\nu+2k}}{k!\,\Gamma(\nu+k+1)} \,.$$

2.1.10 irregular modified cylindrical Bessel functions [tr.math.sf.K]

Synopsis

```
double       cyl_bessel_k(double nu, double x);
float        cyl_bessel_kf(float nu, float  x);
long double  cyl_bessel_kl(long double nu, long double x);
```

Description

These functions compute the irregular modified cylindrical Bessel functions of their respective arguments `nu` and `x`. A domain error may occur if `x` is less zero.

Returns

The `cyl_bessel_k` functions return

$$\mathsf{K}_\nu(x) = (\pi/2)\mathrm{i}^{\nu+1}(\mathsf{J}_\nu(\mathrm{i}x) + \mathrm{i}\mathsf{N}_\nu(\mathrm{i}x)) = \begin{cases} \dfrac{\pi}{2}\dfrac{\mathsf{I}_{-\nu}(x) - \mathsf{I}_\nu(x)}{\sin\nu\pi} & \text{for non-integral } \nu \\ \dfrac{\pi}{2}\lim\limits_{\mu\to\nu}\dfrac{\mathsf{I}_{-\mu}(x) - \mathsf{I}_\mu(x)}{\sin\mu\pi} & \text{for integral } \nu \end{cases} .$$

2.1.11 cylindrical Neumann functions [tr.math.sf.N]

Synopsis

```
double       cyl_neumann(double nu, double x);
float        cyl_neumannf(float nu, float  x);
long double  cyl_neumannl(long double nu, long double x);
```

Description

These functions compute the cylindrical Neumann functions, also known as the cylindrical Bessel functions of the second kind, of their respective arguments `nu` and `x`. A domain error may occur if `x` is less than zero.

Returns

The `cyl_neumann` functions return

$$\mathsf{N}_\nu(x) = \begin{cases} \dfrac{\mathsf{J}_\nu(x)\cos\nu\pi - \mathsf{J}_{-\nu}(x)}{\sin\nu\pi} & \text{for non-integral } \nu \\ \lim\limits_{\mu\to\nu}\dfrac{\mathsf{J}_\mu(x)\cos\mu\pi - \mathsf{J}_{-\mu}(x)}{\sin\mu\pi} & \text{for integral } \nu \end{cases} .$$

2.1.12 (incomplete) elliptic integral of the first kind [tr.math.sf.ellF]

Synopsis

```
double       ellint_1(double k, double phi);
float        ellint_1f(float k, float phi);
long double  ellint_1l(long double k, long double phi);
```

Description

These functions compute the incomplete elliptic integral of the first kind of their respective arguments `k` and `phi`. A domain error may occur if the magnitude of `k` is greater than one.

Returns

The `ellint_1` functions return

$$\mathsf{F}(k,\phi) = \int_0^\phi \frac{\mathsf{d}\theta}{\sqrt{1 - k^2 sin^2\theta}} .$$

2.1.13 (incomplete) elliptic integral of the second kind [tr.math.sf.ellE]

Synopsis

```
double       ellint_2(double k, double phi);
float        ellint_2f(float k, float phi);
long double  ellint_2l(long double k, long double phi);
```

Description

These functions compute the incomplete elliptic integral of the second kind of their respective arguments `k` and `phi`. A domain error may occur if the magnitude of `k` is greater than one.

Returns

The `ellint_2` functions return

$$\mathsf{E}(k,\phi) = \int_0^\phi \sqrt{1 - k^2 \sin^2\theta}\,\mathsf{d}\theta \ .$$

2.1.14 (incomplete) elliptic integral of the third kind [tr.math.sf.ellP]

Synopsis

```
double       ellint_3(double k, double nu, double phi);
float        ellint_3f(float k, float nu, float phi);
long double  ellint_3l(long double k, long double nu, long double phi);
```

Description

These functions compute the incomplete elliptic integral of the third kind of their respective arguments `k`, `nu`, and `phi`. A domain error may occur if the magnitude of `k` is greater than one.

Returns

The `ellint_3` functions return

$$\Pi(\nu,k,\phi) = \int_0^\phi \frac{\mathsf{d}\theta}{(1 - \nu\, sin^2\theta)\sqrt{1 - k^2 sin^2\theta}} \ .$$

2.1.15 exponential integral [tr.math.sf.ei]

Synopsis

```
double       expint(double x);
float        expintf(float x);
long double  expintl(long double x);
```

Description

These functions compute the exponential integral of their respective arguments `x`.

Returns

The `expint` functions return

$$\mathsf{Ei}(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t}\,\mathsf{d}t\ .$$

2.1.16 Hermite polynomials [tr.math.sf.Hn]

Synopsis

```
double       hermite(unsigned n, double x);
float        hermitef(unsigned n, float x);
long double  hermitel(unsigned n, long double x);
```

Description

These functions compute the Hermite polynomials of their respective arguments `n` and `x`.

Returns

The `hermite` functions return

$$\mathsf{H}_n(x) = (-1)^n e^{x^2}\,\frac{\mathsf{d}^n}{\mathsf{d}x^n}\,e^{-x^2}\ .$$

2.1.17 hypergeometric functions [tr.math.sf.hyper]

Synopsis

```
double       hyperg(double a, double b, double c, double x);
float        hypergf(float a, float b, float c, float x);
long double  hypergl(long double a, long double b, long double c,
                     long double x);
```

Description

These functions compute the hypergeometric functions of their respective arguments `a`, `b`, `c`, and `x`. A domain error may occur if the magnitude of `x` is greater than or equal to one.

Returns

The `hyperg` functions return

$$\mathsf{F}(a,b;c;x) = \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)}\sum_{n=0}^{\infty}\frac{\Gamma(a+n)\,\Gamma(b+n)}{\Gamma(c+n)}\,\frac{x^n}{n!}\ .$$

2.1.18 Laguerre polynomials [tr.math.sf.Ln]

Synopsis

```
double       laguerre(unsigned n, double x);
float        laguerref(unsigned n, float x);
long double  laguerrel(unsigned n, long double x);
```

Description

These functions compute the Laguerre polynomials of their respective arguments `n` and `x`.

Returns

The `laguerre` functions return

$$\mathsf{L}_n(x) = e^x \frac{\mathsf{d}^n}{\mathsf{d}x^n} \left(x^n e^{-x}\right) .$$

2.1.19 Legendre polynomials [tr.math.sf.Pl]

Synopsis

```
double       legendre(unsigned l, double x)
float        legendref(unsigned l, float x)
long double  legendrel(unsigned l, long double x)
```

Description

These functions compute the Legendre polynomials of their respective arguments `l` and `x`. A domain error may occur if the magnitude of `x` is greater than one.

Returns

The `legendre` functions return

$$\mathsf{P}_l(x) = \frac{1}{2^l\, l!} \frac{\mathsf{d}^l}{\mathsf{d}x^l} \left(x^2 - 1\right)^l .$$

2.1.20 Riemann zeta function [tr.math.sf.riemannzeta]

Synopsis

```
double       riemann_zeta(double x);
float        riemann_zetaf(float x);
long double  riemann_zetal(long double x);
```

Description

These functions compute the Riemann zeta function of their respective arguments `x`. A domain error occurs if `x` is equal to one.

Returns

The `riemann_zeta` functions return

$$\zeta(x) = \begin{cases} \displaystyle\sum_{k=1}^{\infty} k^{-x} & \text{for } x > 1 \\ 2^x \pi^{x-1} \sin(\frac{\pi x}{2})\, \Gamma(1-x)\, \zeta(1-x) & \text{for } x < 1 \end{cases} .$$

2.1.21 spherical Bessel functions (of the first kind) [tr.math.sf.j]

Synopsis

```
double       sph_bessel(unsigned n, double x);
float        sph_besself(unsigned n, float  x);
long double  sph_bessell(unsigned n, long double x);
```

Description

These functions compute the spherical Bessel functions of the first kind of their respective arguments `n` and `x`. A domain error may occur if `x` is less than zero.

Returns

The `sph_bessel` functions return

$$\mathsf{j}_n(x) = (\pi/2x)^{1/2}\mathsf{J}_{n+1/2}(x) \; .$$

2.1.22 spherical associated Legendre functions [tr.math.sf.Ylm]

Synopsis

```
double       sph_legendre(unsigned l, unsigned m, double theta)
float        sph_legendref(unsigned l, unsigned m, float theta)
long double  sph_legendrel(unsigned l, unsigned m, long double theta)
```

Description

These functions compute the spherical associated Legendre functions of their respective arguments `l`, `m`, and `theta`. A domain error occurs if the magnitude of `m` is greater than `l`.

Returns

The `sph_legendre` functions return

$$\mathsf{Y}_l^m(\theta, 0)$$

where

$$\mathsf{Y}_l^m(\theta,\phi) = (-1)^m \left[\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}\right]^{1/2} \mathsf{P}_l^m(\cos\theta)e^{im\phi} \; .$$

[Note: This formulation avoids any need to return non-real numbers. End note.]

2.1.23 spherical Neumann functions [tr.math.sf.n]

Synopsis

```
double       sph_neumann(unsigned n, double x);
float        sph_neumannf(unsigned n, float  x);
long double  sph_neumannl(unsigned n, long double x);
```

Description

These functions compute the spherical Neumann functions, also known as the spherical Bessel functions of the second kind, of their respective arguments `n` and `x`. A domain error may occur if `x` is less than zero.

Returns

The `sph_neumann` functions return

$$\mathsf{n}_n(x) = (\pi/2x)^{1/2}\mathsf{N}_{n+1/2}(x) \ .$$

2.2 Additions to header `<tgmath.h>` synopsis [tr.math.sf.syn]

For each function in the previous section that has one or more parameters of type `double`, the header `<tgmath.h>` shall have a corresponding type-generic macro. These macros shall behave as described in clause 7.22 of the C standard.

Chapter 3

Implementation Notes [tr.num.sfimp]

Need for testing.
Relative errors/ulp vs. absolute errors.
Use of Mathematica, MapleSoft, etc. to generate tests, approximations.
Existing libraries.
Numerical Recipes as a starting point.
Portability issues.