

```
/*
Copyright (C) 1995-2004 by Andrew V. Nesterov. All rights reserved.
Copyright (C) 1995-2004 by Genral Digital Inc. All rights reserved.
Author: Andrew V. Nesterov
Created: 07/27/1995
Modified: 01/04/2004

```

GDD150 TMS320C3X/C4X Double Precision Math Library

If the SOFTWARE is being Licensed by the FEDERAL GOVERNMENT it is provided with Restricted Rights. Use, duplication, or disclosure by the government is subject to restrictions set forth in 48 CFR 52.227-19(c)(1)(2) or DOD FAR supplement 252.227-7013(c)(1)(ii) as amended through the date hereof.

COMMENTS ON THE LIBRARY FUNCTIONING

A double precision number *d* is stored in a structure *DBLREAL* that has two members, single precision float numbers. The first member is the most significant part of the number (*msp*), the second member is the least significant part of the number (*lsp*). The double precision number is constructed as the sum of the *msp* and *lsp*: $d = msp + lsp$.

Both *msp* and *lsp* have 24 bits significand, therefore the precision of a double precision number is 48 bits. This does not necessary mean that a result of a double precision operation will be accurate to all the 48 bits available. The result may have some inaccuracy in the 2-3 LSBs. Error analysis shows that the relative errors of results of arithmetic operations with double precision operands are bounded by

```
relerr(add)  <= 2**(-46)      = 1.42E-14
relerr(sub)  <= 2**(-46)      = 1.42E-14
relerr(mpy)  <= 11*2**(-48)   = 3.91E-14
relerr(div)  <= 21.1*2**(-48) = 7.50E-14
relerr(sqrt) <= 12.7*2**(-48) = 4.51E-14

```

The theoretical bound of number of a number of decimal places of a 48 bits significand is $48 \cdot \lg(2) = 14.4494397\dots$

ARGUMENTS PASSING AND MEMORY MODEL

Library functions use stack argument passing model and support both small and big TI C memory models.

FUNCTION DESCRIPTIONS

```
sdadd(x,y) Add two single precision operands,
            return a double precision result.
sdsb(x,y) Subtract two single precision operands,
            return a double precision result.
sdmpy(x,y) Multiply two single precision operands,
            return a double precision result.
sddiv(x,y) Divide two single precision operands,
            return a double precision result.
sdsqrt(x) Compute square root of a single precision argument,
            return a double precision result.

ddadd(x,y) Add two double precision operands,
            return a double precision result.
ddsb(x,y) Subtract two double precision operands,
            return a double precision result.
ddmpy(x,y) Multiply two double precision operands,
```

return a double precision result.
 dddiv(x,y) Divide two double precision operands,
 return a double precision result.
 ddsqrt(x) Compute square root of a double precision argument,
 return a double precision result.

TIMING (including C call overhead)

FUNCTION	time F (cycles)	40MHz C3x/4x (50ns rate)	float time f (cycles)	upper bound of the ratio F/f
sdadd(x,y)	29	1.45 microseconds	1	30.0
sdsb(x,y)	29	1.45 microseconds	1	30.0
sdmpy(x,y)	65	3.25 microseconds	1	65.0
sddiv(x,y)	121	6.05 microseconds	40	3.1
sdsqrt(x)	158	7.90 microseconds	47	3.4
ddadd(x,y)	54	2.70 microseconds	1	55.0
ddsb(x,y)	54	2.70 microseconds	1	55.0
ddmpy(x,y)	83	4.15 microseconds	1	85.0
ddddiv(x,y)	129	6.45 microseconds	40	3.3
ddsqrt(x)	161	8.05 microseconds	47	3.5

FILES

GDD1503x.lib is the TMS320C3X library module
 GDD1504x.lib is the TMS320C4X library module

REFERENCES

T.J.Dekker, A Floating Point Technique for Extending the Available Precision , Numerical Mathematics, Vol. 18, pp. 224-242, 1971.

S.Linnainmaa, Software for Double-Precision Floating Point Computations, ACM Transactions on Mathematical Software, Vol. 7, No. 3, pp. 272-283, September 1981.

A. Lovrich, Doublelength Floating-Point Arithmetic on the TMS320C30, in Digital Signal Processing Applications with the TMS320 family, vol 3, Texas Instruments, 1990

*****/

```

#ifndef _DBLMT34_H_
#define _DBLMT34_H_

/* DOUBLE PRECISION TYPE */
#ifndef _DOUBLE_PRECISION_TYPE_
#define _DOUBLE_PRECISION_TYPE_
typedef struct
    { float msp; /* most significant part */
      float lsp; /* least significant part */
    } DBLREAL;
#endif

/* FUNCTION DECLARATIONS */

DBLREAL sdadd (float FLT0, float FLT1); /* DBL_RES = FLT0 + FLT1 */
DBLREAL sdsb (float FLT0, float FLT1); /* DBL_RES = FLT0 - FLT1 */
DBLREAL sdmpy (float FLT0, float FLT1); /* DBL_RES = FLT0 * FLT1 */

```

```
DBLREAL sddiv (float FLT0, float FLT1);    /* DBL_RES = FLT0 / FLT1 */
DBLREAL sdsqrt(float FLT0);                /* DBL_RES = SQRT(FLT0) */

DBLREAL ddadd (DBLREAL DBL0, DBLREAL DBL1); /* DBL_RES = DBL0 + DBL1 */
DBLREAL ddsb (DBLREAL DBL0, DBLREAL DBL1); /* DBL_RES = DBL0 - DBL1 */
DBLREAL ddmpy (DBLREAL DBL0, DBLREAL DBL1); /* DBL_RES = DBL0 * DBL1 */
DBLREAL dddiv (DBLREAL DBL0, DBLREAL DBL1); /* DBL_RES = DBL0 / DBL1 */
DBLREAL ddsqrt(DBLREAL DBL0);              /* DBL_RES = SQRT(DBL0) */

#endif /* _DBLMT34_H_ */
```