Trigon and Trigonometry libraries for PIC18 processors

version 0.05

These libraries were prepared to work with MathDouble replacement of mE's system floating-point library (__Lib_MathDouble.mcl). Please do not use them separately, i.e. with the original MathDouble lib. Moreover, the Trigon library requires presence of other replacement libs, namely the Strings and Conversions libraries, so please use the whole package of mentioned replacement libs.

Both libs described here contain all the routines present in the originals (though differently distributed) plus a set of routines that were formerly contained in an additional lib, called Fpmath and distributed with earlier MathDouble lib replacement versions (up to v 0.07).

The replacement libraries produce smaller code than the originals, while giving more accurate calculation results, and allow to exploit the exception mechanism introduced in MathDouble lib replacement. For the use of the exception mechanism see description of the MathDouble replacement lib.

Detailed differences from the originals:

- exception mechanism for f-p math errors,
- NaN (Not a Number) bit set in FPstatus for negative arguments in Sqrt and in pow(x,y) when x is negative and y is not an integral number,
- added routines: rSgn, rAbs, Round, Trunc, Int, Frac, rFrexp, rMin, rMax, Sqr, conversion routines to/from string & IEEE 754 format,
- corrected errors in atan2 function,
- most functions are noticeably faster, some, like Sqrt - much faster.

Due to introduction of rounding while converting to integer types (in both implicit and explicit conversion), use of the replacement libs may require older code correction, as with the official MathDouble lib conversion was based on truncation (the fractional part of real number was simply omitted). This issue is addressed by introducing a set of useful routines to the Trigon library – one of them, Trunc, may be used whenever rounding is not wanted. There is also Round function which does nothing more than explicit conversion for variables but should be used with constants or there will be a difference between runtime and compile-time calculations.

For clarity and ease of use, strictly trigonometric functions were moved from Trigon to Trigonometry library. Almost all functions contained in the latter are present in official libraries and as such are documented in Help. Added functions, sin_fast and cos_fast, execute in 5 times shorter time than the regular ones while keeping accuracy level sufficient, for example, for graphical calculations.
Functions from Trigon lib that were already present in the original lib do not need explanation as they

are described in Help. Routines added to this library are listed below.

rSgn

rAbs

Round

Trunc

Frac

rFrexp

Int

rMax

rMin

Sqr

L2Bendian

Real2IEEE

IEEE2real

Real2str

str2real

Real2strES

You'll find their descriptions later in text (or click on a name of a routine to jump to it's prototype). There are also version strings declared, called Lib_Trigon_ver and Lib_Trigonometry_ver (constant string [4]) that may be called from user code for verification.

Notes:

- Two routines, rFoor and rCeil, present in old Fpmath lib, were dropped out as they're no longer needed - Floor and Ceil routines in Trigon lib replacement naturally work well with the MathDouble replacement lib.
- The str2float procedure from the Conversions library replacement may still be used, but the new str2real procedure from Trigon lib is more advanced.

Manual libraries' installation:

- find mP PRO installation directory and subdirectory **.../Uses/P18**,
- find original library files, **__Lib_Trigon.mcl** and **__Lib_Trigonometry.mcl**, there and rename them, or move elsewhere,
- unpack the replacement libs and move the files to the **.../Uses/P18** directory.

All the routines, old and new, will be visible in Library Manager.

Have fun,

janni

Trigon library - fp-math routines for PIC18 processors
version 0.05
date 20.01.14
Revision history:
0.01 – added L2Bendian
0.02 – sqrt function optimised
0.03 – compiled with mP PRO v5.00
0.04 – optimised exp function
0.05 – optimised cos, acos; compiled with ver 6.01

Library routines:

function Ceil(x: real): real;

function Floor(x: real): real;

function Sqrt(x:real):real;
Note: NaN is set for negative arguments

function pow(x,y:real):real;
Note: NaN is set for negative x when y is not integral number

function eval_poly(x:real; var dd:array[10] of real; nn:byte):real;

function exp(x:real):real;

function log(x:real):real;

function log10(x:real):real;

Additional routines:

function rSgn(x:real):short;
gets sign of type real variable, result equals 1 or -1

function rAbs(x:real): real;
calculates the absolute value of number of type real

function Round(x: real): longint;
rounds the floating-point value to nearest integer (same result may be achieved by simple typecasting, longint(x))

function Trunc(x: real): longint;
truncates the floating-point value to integer (cuts-out the fractional part)

function Frac(x: real): real;
gives the signed fractional part of x

procedure rFrexp(x: real; var Mantissa: real; var Exponent: real);
returns the mantissa of x as Mantissa and the exponent as Exponent

function Int(x: real): real;
returns the integer part of x; that is, x rounded toward zero

function rMax(A,B: real): real;
returns the greater of two values of type real

function rMin(A,B: real): real;
returns the smaller of two values of type real

function Sqr(x: real): real;
returns the square of the argument

procedure L2Bendian(ptr4:^byte);

converts 4-byte number (dword, longint or real) from little to big endian or reverse;
example: LEn2BEn(^byte(@real_var))

function Real2IEEE(var x: real): boolean;

converts x to IEEE 754 format; numbers bigger in magnitude than +/- 3.4E38 are represented as +/- infinity - result is true, if this was not necessary;
NOTE: IEEE 754 number is stored inmemory as little endian, i.e. least significant byte first

function IEEE2real(var x: real): boolean;

converts a number in IEEE 754 format (little endian) to real; denormalised (small) numbers are converted to (signed) zero; infinities and NaNs to maximal float values; Not A Number (NaN) flag is set in FPstatus if appropriate; result is true if there were no compatibility issues, false otherwise

procedure Real2str(x:real; var outst:string[17]; inum,dpoint:byte);

converts the floating-point value to string with basic formatting:
inum - minimal number of chars before decimal point [0..11],
dpoint - number of decimal places [0..5];
if inum is larger than the size of the integer part, leading spaces will be added. It may be smaller - the number will be still correctly converted.

procedure Real2strES(x:real; var outst:string[14]; dpoint:byte; eng:boolean);

converts the floating-point value to string in engineering or scientific format;
dpoint - number of decimal places [0..5];
eng - engineering format (-nnn.dddddE+ee, nnn=1...999), if true,
scientific (-n.dddddE+ee, n=1..9), if false;
for given dpoint value resulting string length is constant for any x:
engineering format: length equals 8 if dpoint=0, otherwise it is dpoint+9
scientific format: length equals 6 if dpoint=0, otherwise it is dpoint+7

function str2real(var inst: string[17]): real;

converts inst to floating-point value; inst must consist of an optional sign (+ or -), a string of digits with an optional decimal point, and an optional exponent. The exponent consists of 'E' or 'e' followed by an optional sign (+ or -) and a whole number. Leading and trailing blanks are ignored; FP_DOM (domain error) bit in FPstatus (which may be read with Get_FPstatus function) is set in case of conversion error or, if the number coded in inst, though formally correct, doesn't belong to Microchip format range (other FPstatus bits may also be set);
NOTE: FPstatus is cleared in str2real, so it's previous state is not carried through the function.
As a side effect, inst is formatted - spaces are removed, 'e' changes to 'E'.

Trigonometry library - trigonometric functions for PIC18 processors

version 0.04

date 31.01.12

Revision history:

- 0.01 - compiled with v4.80 beta
- 0.02 - atan2 corrected
- 0.03 – compiled with mP PRO v5.00
- 0.04 – added sin_fast & cos_fast functions, optimised sin & atan functions

---

Library routines:

function sin(arg:real):real;

function cos(arg:real):real;

function tan(arg:real):real;

function asin(x:real):real;

function acos(x:real):real;

function atan(arg:real):real;

function atan2(y:real; x:real):real;

function sinh(x:real):real;

function cosh(x:real):real;

function tanh(x:real):real;

function sinE3(angle_deg:word):integer;

function cosE3(angle_deg:word):integer;

Additional routines:

function sin_fast(arg:real):real;

arg=-pi..pi, absolute error < 0.001

function cos_fast(arg:real):real;

arg=-pi..pi, absolute error < 0.001