

rexxmathlib

Thomas Richter

COLLABORATORS

	<i>TITLE :</i> rexxmathlib		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Thomas Richter	August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	rexxmathlib	1
1.1	main	1
1.2	The THOR-Software Licence	1
1.3	overview	2
1.4	How to fix the compare bug.	3
1.5	function_index	3
1.6	abs	4
1.7	acos	4
1.8	acosh	4
1.9	asin	5
1.10	asinh	5
1.11	atan	6
1.12	atan2	6
1.13	atanh	7
1.14	ceil	7
1.15	cos	8
1.16	cosec	8
1.17	cosh	8
1.18	cot	9
1.19	e	9
1.20	epsm	10
1.21	emsp	10
1.22	exp	10
1.23	fact	11
1.24	floor	11
1.25	fract	12
1.26	log	12
1.27	log10	13
1.28	nint	13
1.29	pi	14

1.30	pow	14
1.31	root	14
1.32	sec	15
1.33	sin	15
1.34	sinh	16
1.35	sqr	16
1.36	tan	16
1.37	tanh	17

Chapter 1

rexxmathlib

1.1 main

RexxMathLib.Guide - First Aid about RexxMathLib © 1995 THOR-Software
Guide Version 1.01 / Library Version 38.01

Table of Contents

- I. What is it: Overview
- II. Function Index

© THOR-Software
Thomas Richter
Rühmkorffstraße 10A

12209 Berlin

Germany

E-Mail: thor@einstein.math.tu-berlin.de

The rexxmathlib.library is FREeware and copyrighted © 1995 by
Thomas Richter. No commercial use without permission of the
author. Read the licence !

There is a bug in the mathieeedoubbas.library that comes with
Workbench 3.1. Read here to find out how to fix it.

1.2 The THOR-Software Licence

The THOR-Software Licence

This License applies to the computer programs known as "RexxMathLib".
The "Program", below, refers to such program.

The programs and files in this distribution are freely distributable
under the restrictions stated below, but are also Copyright (c)
Thomas Richter.

Distribution of the Program by a commercial organization without written permission from the author to any third party is prohibited if any payment is made in connection with such distribution, whether directly (as in payment for a copy of the Program) or indirectly (as in payment for some service related to the Program, or payment for some product or service that includes a copy of the Program "without charge"; these are only examples, and not an exhaustive enumeration of prohibited activities). However, the following methods of distribution involving payment shall not in and of themselves be a violation of this restriction:

(i) Posting the Program on a public access information storage and retrieval service for which a fee is received for retrieving information (such as an on-line service), provided that the fee is not content-dependent (i.e., the fee would be the same for retrieving the same volume of information consisting of random data).

(ii) Distributing the Program on a CD-ROM, provided that the files containing the Program are reproduced entirely and verbatim on such CD-ROM, and provided further that all information on such CD-ROM be redistributable for non-commercial purposes without charge.

Everything in this distribution must be kept together, in original and unmodified form.

Limitations.

THE PROGRAM IS PROVIDED TO YOU "AS IS," WITHOUT WARRANTY. THERE IS NO WARRANTY FOR THE PROGRAM, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IF YOU DO NOT ACCEPT THIS LICENCE, YOU MUST DELETE ALL FILES CONTAINED IN THIS ARCHIVE.

1.3 overview

This library provides transcendental functions for the use with the ARexx - programming language. It is a complete new revision of Willy Langevelde's rexxmathlib, although no original code has been used. The new release has been completely rewritten in assembly language and is therefore not only faster (approx. 10 times), but provides also a higher precision of 15.9 digits, thanks to cleverer ASCII- to Float conversion routines.

As an extra, some more functions have been added, see the Function Index, and the checking for proper function arguments are now more strictly.

To use this library in AREXX, add the following line to your ARExx-script:

```
call addlib('rexxmathlib.library',0,-30,0)
```

The rexxmathlib.library will use the system math libraries, namely the mathieeedoubbas.library and the mathieeedoubtrans.library and will therefore work fine, regardless of a math-coprocessor.

There is a bug in the V38 mathieeedoubbas.library float-compare routine resulting in a wrong ordering of negative numbers of small absolute value.

However, I included the necessary stuff to fix this bug. Read here to find out how to do this.

1.4 How to fix the compare bug.

I advise you to fix the bug in the mathieeedoubbas library version 38.2, that comes with Workbench 3.1. For copyright reasons, I can not provide the patched library, but a patch file and a patch program. To apply the patch:

- 1) Copy the file LIBS:mathieeedoubbas.library to RAM:
- 2) Copy the file mathieeedoubbas.pch, which comes with this archive, to RAM:
- 3) Copy the program spatch, which is also included in this archive, to ram:
- 4) Change the directory to ram: with
cd ram:
- 5) Apply the patch with
spatch mathieeedoubbas.library
- 6) Copy back the file RAM:mathieeedoubbas.new to
LIBS:mathieeedoubbas.library. It contains the fixed library.

If any problems arise, make shure you use the original (CBM) version of the library!

1.5 function_index

RexxMathLib.library - Function Index

```
ABS ACOS ACOSH ASIN
ASINH ATAN ATAN2 ATANH
CEIL COS COSEC COSH
COT COTAN CSC E
EPSM EPSP EXP FABS
FACT FLOOR FRACT INT
LN LOG LOG10 NINT
PI POL POW POWER
ROOT SEC SIN SINH
```

SQR SQRT TAN TANH
XTOY

1.6 abs

NAME

ABS(x), FABS(x)

calculate absolute value of the argument

ARGUMENT REQUIREMENTS

none

BUGS

-ABS is never called by AREXX, cause it is provided as a AREXX build-in function. However, you SHOULD use FABS if you need the absolute value, cause it provides a higher precision than the build-in ABS

SEE ALSO

1.7 acos

NAME

ACOS(x)

calculate the inverse cosine of the argument
(in radiants)

ARGUMENT REQUIREMENTS

$-1.0 \leq x \leq 1.0$

BUGS

SEE ALSO

COS
SIN
TAN
ASIN
ATAN

1.8 acosh

NAME

ACOSH(x)

calculate the inverse hyperbolic cosine of the argument

ARGUMENT REQUIREMENTS

$x \geq 1.0$

BUGS

This function is implemented by the identity $\text{ACOSH}(x) = \text{LN}(x + \text{SQRT}(x^2 - 1))$ and might cause an overflow if the argument of the logarithm overflows or x^2 is out of range. A second result of this implementation is a non-guaranteed maximum precision.

SEE ALSO

COSH
SINH
TANH
ASINH
ATANH

1.9 asin

NAME

ASIN(x)

calculate the inverse sine of the argument
(in radians)

ARGUMENT REQUIREMENTS

$-1.0 \leq x \leq 1.0$

BUGS

SEE ALSO

COS
SIN
TAN
ACOS
ATAN

1.10 asinh

NAME

ASINH(x)

calculate the inverse hyperbolic sine of the argument

ARGUMENT REQUIREMENTS

none

BUGS

This function is implemented by the identity $\text{ASINH}(x) = \text{LN}(x + \text{SQRT}(x^2 + 1))$ and might cause an overflow if the argument of the logarithm overflows or x^2 is out of range. A second result of this implementation is a non-

garantueed maximum precision.

SEE ALSO

COSH
SINH
TANH
ACOSH
ATANH

1.11 atan

NAME

ATAN(x)

calculate the inverse tangent of the argument
(in radiants)

ARGUMENT REQUIREMENTS

none

BUGS

As a result of finite precision, the inverse tangent of
PI/2 is NOT infinity.

SEE ALSO

COS
SIN
TAN
ACOS
ASIN

1.12 atan2

NAME

ATAN2(y, x), POL(x, y)

calculate the angle between the point (x|y) and the origin
(in radiants)

NOTE THE DIFFERENT ARGUMENT ORDERING OF ATAN2 AND POL !

This function is also known as the argument-function
of the complex number $z=x+iy$

For *many* values of x and y is this argument identical
to ATAN(x/y), but TRIES to provide a higher precision.

ARGUMENT REQUIREMENTS

$x \neq 0$ | $y \neq 0$

x and y must not be zero at the same time, however $x=0$
OR $y=0$ is allowed.

BUGS

SEE ALSO

1.13 atanh

NAME

ATANH(x)

calculate the inverse hyperbolic tangent of the argument

ARGUMENT REQUIREMENTS

$-1.0 < x < 1.0$

BUGS

This function is implemented by the identity $ASINH(x) = \ln((1+x)/(1-x))/2$ and might cause an overflow if the argument of the logarithm overflows. A second result of this implementation is a non-garantueed maximum precision.

SEE ALSO

COSH
SINH
TANH
ACOSH
ASINH

1.14 ceil

NAME

CEIL(x)

calculate the lowest integer higher than x

ARGUMENT REQUIREMENTS

none

BUGS

Not a bug, but you should note that this function results for negative values of x in a number of lower absolute value.

So

$CEIL(2.5) = 3$

but

$CEIL(-2.5) = -2$

However, this is the CORRECT mathematical implementation of CEIL !

SEE ALSO

FLOOR
FLOOR
FRACT

NINT

1.15 cos

NAME

COS(x)

calculate the cosine of the argument
(in radiants)

ARGUMENT REQUIREMENTS

none

BUGS

As a result of finite precision, the cosine of x
of high absolute value is more or less random.

SEE ALSO

SIN
TAN
ACOS
ASIN
ATAN

1.16 cosec

NAME

COSEC(x), CSC(x)

calculate the cosecans of the argument
(in radiants)

ARGUMENT REQUIREMENTS

$x \neq 0$

BUGS

As a result of finite precision, the cosecans of
integer multiples of PI is not infinity, also for
x of high absolute value the result is more or
less random.

SEE ALSO

SEC
COT
COTAN

1.17 cosh

NAME

COSH(x)

calculate the hyperbolic cosine of the argument

ARGUMENT REQUIREMENTS

 $-700 < x < 700$ (approx.)

BUGS

SEE ALSO

SINH
TANH
ACOSH
ASINH
ATANH

1.18 cot

NAME

COT(x),COTAN(x)

calculate the hyperbolic cotangent of the argument
(in radians)

ARGUMENT REQUIREMENTS

 $x \neq 0$

BUGS

As a result of finite precision, the cotangent of $\text{PI}/2$
is not zero.

SEE ALSO

TAN
SEC
COSEC
CSC

1.19 e

NAME

E(x)

return the value of E, the base of the natural
logarithm. The argument is not used.

ARGUMENT REQUIREMENTS

none

BUGS

The result has a precision of 17 digits, although

rexxmathlib has only a precision 15.9 digits
(and AREXX of 14 digits)

SEE ALSO
PI

1.20 epsm

NAME
EPSM(x)

return the highest floating point number lower than and
distinguishable from x

ARGUMENT REQUIREMENTS
none

BUGS
The result is only useful as input for mathrexxlib, cause
AREXX itself has a limited precision of 14 digits and
so EPSM(x)=x for AREXX.

SEE ALSO
EPSP

1.21 epsp

NAME
EPSP(x)

return the lowest floating point number higher than and
distinguishable from x

ARGUMENT REQUIREMENTS
none

BUGS
The result is only useful as input for mathrexxlib, cause
AREXX itself has a limited precision of 14 digits and
so EPSP(x)=x for AREXX.

SEE ALSO
EPSM

1.22 exp

NAME
EXP(x)
calculate the exponential function

ARGUMENT REQUIREMENTS

$x < 700$ (approx.)

BUGS

SEE ALSO

E
LOG
LN

1.23 fact

NAME

FACT(x)

calculate the factorial function of x

ARGUMENT REQUIREMENTS

$x \geq 0$ & $x \leq 87$ & x integer

BUGS

For x lower or equal than 12, the result is calculated in integers, for higher x floating point numbers are used, so the result might be a non-integer.

This call should really evaluate the Gamma-function for non-integer x, but this is a non-trivial task !

SEE ALSO

1.24 floor

NAME

FLOOR(x), INT(x)

calculate the highest integer lower than x

ARGUMENT REQUIREMENTS

none

BUGS

Not a bug, but you should note that this function results for negative values of x in a number of higher absolute value.

So

INT(2.5)=2

but

INT(-2.5)=-3

However, this is the CORRECT mathematical implementation of INT !

SEE ALSO

CEIL
FRACT
NINT

1.25 fract

NAME

FRACT(x)

calculate the fractional part of x

ARGUMENT REQUIREMENTS

none

BUGS

Not a bug, but you should note that this function results for negative values of x also in a positive number, cause it is implemented as $x - \text{FLOOR}(x)$.

So

$\text{FLOOR}(2.4) = 0.4$

but

$\text{INT}(-2.4) = 0.6$

However, this is the CORRECT mathematical implementation of FRACT !

SEE ALSO

CEIL
FRACT
NINT

1.26 log

NAME

LN(x), LOG(x)

calculate the natural logarithm of x

ARGUMENT REQUIREMENTS

$x > 0$

BUGS

SEE ALSO

E
EXP
LOG10

1.27 log10

NAME

LOG10(x)

calculate the decadic logarithm of x

ARGUMENT REQUIREMENTS

x>0

BUGS

SEE ALSO

LOG
LN

1.28 nint

NAME

NINT(x)

calculate the nearest integer to x

ARGUMENT REQUIREMENTS

none

BUGS

Not a bug, but you should note that this function results for negative values of x with a fractional part of 0.5 in a different integer than for positive x. So

NINT(2.5)=3

but

NINT(-2.5)=-2

However, this is the CORRECT mathematical implementation of NINT, but differs from the behavior of the old version of rexxmathlib.

SEE ALSO

FLOOR
INT
CEIL
FRACT

1.29 pi

NAME

PI(x)

return the value of PI. The argument is not used.

ARGUMENT REQUIREMENTS

none

BUGS

The result has a precision of 17 digits, although rexxmathlib has only a precision 15.9 digits (and AREXX of 14 digits)

SEE ALSO

E

1.30 pow

NAME

POW(x, y), POWER(x, y), XTOY(x, y)

return x to the power of y

ARGUMENT REQUIREMENTS

messy...

For non-integer y x must be positive or zero.

For integer y x can be both positive or negative, however x and y must not be both zero.

A second requirement is that both x and y must not be "too large".

BUGS

0 to the power of 0 is not allowed, although the old version of rexxmathlib can handle this. However, 0^0 is mathematically not well defined and can be both, zero or one.

SEE ALSO

ROOT

1.31 root

NAME

ROOT(x, y)

return the yth root of x

ARGUMENT REQUIREMENTS

messy...

For non-integer y x must be positive or zero.

For integer and odd y x can be both positive or negative, y must be non-zero.
A second requirement is that x must not be "to large" and y not "to small".

BUGS

For y being 1 x is returned immediatly and for y being 2 the square-root function is used. All other arguments are passed to POW, except for the extra sign handling of odd roots. This is a real mess...

SEE ALSO

POW
POWER
XTOY

1.32 sec

NAME

SEC(x)

calculate the secans of the argument
(in radiants)

ARGUMENT REQUIREMENTS

none

BUGS

As a result of finite precision, the secans of integer odd multiples of $\text{PI}/2$ is not infinity, also for x of high absolute value the result is more or less random.

SEE ALSO

COSEC
CSC
COT
COTAN

1.33 sin

NAME

SIN(x)

calculate the sine of the argument
(in radiants)

ARGUMENT REQUIREMENTS

none

BUGS

As a result of finite precision, the sine of x of high absolute value is more or less random.

SEE ALSO

COS
TAN
ACOS
ASIN
ATAN

1.34 sinh

NAME

SINH(x)

calculate the hyperbolic sine of the argument

ARGUMENT REQUIREMENTS

$-700 < x < 700$ (approx.)

BUGS

SEE ALSO

COSH
TANH
ACOSH
ASINH
ATANH

1.35 sqr

NAME

SQR(x), SQRT(x)

calculate the square root of x

ARGUMENT REQUIREMENTS

$x \geq 0$

BUGS

SEE ALSO

ROOT
POW
POWER
XTOY

1.36 tan

NAME

TAN(x)

calculate the tangent of the argument
(in radians)

ARGUMENT REQUIREMENTS

none

BUGS

As a result of finite precision, the tangent of $\text{PI}/2$ is not infinity, also for x of high absolute value is more or less random.

SEE ALSO

COS
SIN
ACOS
ASIN
ATAN

1.37 tanh

NAME

TANH(x)

calculate the hyperbolic tangent of the argument
(in radians)

ARGUMENT REQUIREMENTS

none

BUGS

SEE ALSO

COSH
SINH
ACOSH
ASINH
ATANH
