Documentation of the library
WagoAppMath
Release 1.1.1.1
## 1 Description

## 2 10 Documentation

2.1 doc10_general (FB) .................................. 3
2.2 Suffixes for result types ............................. 3
2.3 Angular types ......................................... 3
2.4 Abbreviated argument name ......................... 4
2.5 doc92Specifications (FB) .......................... 4

## 3 20 Program Organization Units

3.1 01 Elementary Math ................................. 5
  3.1.1 01 Commonly Used ................................ 5
    cbrt\_L (FUN) ........................................ 5
    cbrt\_r (FUN) ........................................ 5
    cub\_L (FUN) .......................................... 6
    cub\_r (FUN) .......................................... 6
    exp\_L (FUN) .......................................... 6
    exp\_r (FUN) .......................................... 7
    log10\_L (FUN) ........................................ 7
    log10\_r (FUN) ........................................ 7
    log2\_L (FUN) .......................................... 8
    log2\_r (FUN) .......................................... 8
    logN\_L (FUN) .......................................... 9
    logN\_r (FUN) .......................................... 9
    nthroot\_L (FUN) ...................................... 9
    nthroot\_r (FUN) ...................................... 10
    pot\_L (FUN) .......................................... 10
    pot\_r (FUN) .......................................... 11
    sqr\_L (FUN) .......................................... 11
    sqr\_r (FUN) .......................................... 11
    sqrt\_L (FUN) ......................................... 12
    sqrt\_r (FUN) ......................................... 12
  3.1.2 02 Trignometric ................................. 12
    Types and Conversion ................................ 12
    angleToDegree\_L (FUN) .............................. 12
    angleToDegree\_r (FUN) .............................. 13
    angleToRadiant\_L (FUN) .......................... 13
    angleToRadiant\_r (FUN) ...................... 14
    angle\_t (ALIAS) .................................... 14
    degreeToAngle (FUN) ............................... 14
    normalizeAngle\_Positive (FUN) ................. 15
    normalizeAngle\_Signed (FUN) ................... 15
RadiantToAngle (FUN) .................................................. 15
arcCos (FUN) ................................................................. 16
arcSin (FUN) ................................................................. 16
arcTan (FUN) ................................................................. 16
arcTan2 (FUN) ............................................................... 17
cos_L (FUN) ................................................................. 17
cos_r (FUN) ................................................................. 18
sin_L (FUN) ................................................................. 18
sin_r (FUN) ................................................................. 18
tan_L (FUN) ................................................................. 19
tan_r (FUN) ................................................................. 19
3.1.3 03 Combinatorics .................................................. 19
binomial (FUN) .............................................................. 19
factorial (FUN) ............................................................. 20
greatestCommonDivisor (FUN) ......................................... 20
leastCommonMultiple (FUN) ............................................ 21
3.1.4 04 Miscellaneous ................................................... 21
IsInfinite (FUN) ............................................................. 21
IsNaN (FUN) ................................................................. 22
abs_L (FUN) ................................................................. 22
abs_r (FUN) ................................................................. 22
ceil_L (FUN) ................................................................. 23
ceil_r (FUN) ................................................................. 23
copySign_L (FUN) ......................................................... 24
copySign_r (FUN) ......................................................... 24
div_L (FUN) ................................................................. 25
div_r (FUN) ................................................................. 25
floor_L (FUN) ............................................................... 26
floor_r (FUN) ............................................................... 26
fmod_L (FUN) ............................................................... 26
fmod_r (FUN) ............................................................... 27
fpSplit_L (FUN) ............................................................ 27
fpSplit_r (FUN) ............................................................ 28
isWithinTolerance_L (FUN) .............................................. 28
isWithinTolerance_r (FUN) ............................................. 29
nearest_L (FUN) ........................................................... 29
nearest_r (FUN) ........................................................... 30
sign_L (FUN) ............................................................... 30
sign_r (FUN) ............................................................... 30
4 Constants (GVL) ......................................................... 32
5 LibraryResult (GVL) ...................................................... 33
6 ResultItems (GVL) ......................................................... 34
7 VersionHistory (GVL) .................................................... 35
8 Library Reference ......................................................... 36
This document is automatically generated. Because of this, the chapter 30 Visualization is not shown in this document. If you are interested in getting to know more about visualization, we refer to the library manager of e!Cockpit.

Subject to Changes

WAGO Kontakttechnik GmbH Co. KG reserves the right to provide for any alterations or modifications. WAGO Kontakttechnik GmbH Co. KG owns all rights arising from the granting of patents or from the legal protection of utility patents. Third-party products are always mentioned without any reference to patent rights. Thus, the existence of such rights cannot be excluded.

Personnel Qualification

All tasks that are carried out with libraries made for the e!COCKPIT software must only be performed by qualified electrical specialists instructed in PLC programming according to IEC 61131-3.

All tasks that have an effect on the properties or the behavior of automation hardware or software products must only be performed by qualified employees with a thorough knowledge of handling the products concerned.

Intended Use of e!COCKPIT Libraries

Libraries created for the e!COCKPIT software are used to simplify the development of application projects in the IEC 61131-3 programming languages.

For automation tasks, WAGO offers programmable logic controllers in a wide variety of performance classes. In combination with a wide range of I/O modules, the controllers can process standard types of field signals. Controllers can be implemented centrally or in decentralized configurations. The controllers offer interfaces for the most commonly used fieldbuses for use in decentralized configurations. Fieldbus independent I/O modules are then linked via fieldbus couplers. WAGO controllers offer a runtime environment for user programs called e!RUNTIME. Software projects for implementation in e!RUNTIME environments can be created in e!COCKPIT. The programming environment in e!COCKPIT is based on the established CODESYS 3 industrial standard. Users with a previous knowledge of CODESYS 3 will thus find this environment largely familiar. The following programming languages of the IEC 61131-3 standard are available:

- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Sequential Function Chart (SFC)
- Continuous Function Chart (CFC)

The individual programming languages can also be combined as required during the development of the software. A portfolio of prepared libraries can be accessed for many frequently used functions in order to make software development more efficient. This document provides an overview of the WagoAppMath that WAGO offers for e!COCKPIT.
Mathematical standard routines

Further library information are summerized here:

**Company**  WAGO  
**Title**  WagoAppMath  
**Version**  1.1.1.1  
**Categories**  WAGO FunctionalView|Standard Algorithms; WAGO LayerView|App; Application  
**Author**  WAGO / u013972  
**Placeholder**  WagoAppMath
2.1 doc10_general (FB)

This library provides for some elementary mathematical functions and functions blocks which are used ubiqui-
tously in application programs.

Some peculiarities should be noted in advance:

2.2 Suffixes for result types

When using numerical floating point functions, the result may be stored in various variable formats, such as REAL,
LREAD, or also integer formats. Due to this circumstance, many functions or FB carry a suffix when different
versions are provided for different result types.

E.g.:

\[
\begin{align*}
lrResult & := \text{cbrt}_L(x); & \text{// for the cubic-root in LREAL format} \\
rResult & := \text{cbrt}_r(x); & \text{// for the cubic-root in REAL format}
\end{align*}
\]

Note(1): regularly, a small letter suffix is chosen in order not to obscure the main name of the function. For
LREAL, however, a capital ‘L’ is chosen, because a small ‘l’ could be mistaken for a capital ‘I’.

Note(2): This distinguishing takes place only for the result, not for the input. This is because inputs are generally
typed as the largest sensible type for these cases. When a function is fed with an expression of lesser precision, this
could be losslessly converted to the higher precision without any need for a warning - and this is done internally
by the compiler.

2.3 Angular types

In order to

1. provide a maximum performance for resource-consuming trigonometric functions and

2. avoid confusion between radians, degrees, and other angular representations,

angular inputs or outputs are represented by an internal type ‘angle_t’ - which is typically the native trigonometric
floating point type for the given PLC.

\[
\begin{align*}
\text{VAR} & \\
\phi & := \text{DegreeToAngle}(45.0); & \text{// the angle} \\
rKathete & := \text{sin}_r(\phi) & \text{// the sine}
\end{align*}
\]

\text{END\_VAR}
Conversion functions are provided to give explicite representations of such an angle in degrees or radians, and to import angles from degree or radians variables.

Note: As trigonometric functions are generally used for transforming between cartesian coordinates and polar coordinates (in any kind), this angular type helps to avoid confusion between both systems.

2.4 Abbreviated argument name

Most elementary mathematic functions take exactly one floating point argument and give one result. Because the brevity of the function names themselves, it seemed to be inadequate to use long input names (such as ‘lrInput-Value’) for the only input. Instead, the input is just named ‘x’ ( e.g.: arcSin(x) ), as it us usual in mathematical contexts.

Note that this abbreviation applies to elementary function only where the argument name is not mentioned in the call. Function blocks, methods, all other not-so-simple units, and especially all units, where the argument name must be stated for the call, still carry long descriptive names.

2.5 doc92_specifications (FB)

This library refers to the following documents and specifications:

1. CAA_Mathematics.library
2. man pages “math.h” and “libm.h” (C99 / Posix)
4. Gnu Scientific Library
6. Siehe Util-Bibliohek von 3S.
3.1 01 Elementary Math

3.1.1 01 Commonly Used

cbrt_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cbrt_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose third root is calculated.</td>
</tr>
</tbody>
</table>

Function

The cubic root of x (LREAL).

Graphical Illustration

---
cbrt_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cbrt_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value whose third root is calculated.</td>
</tr>
</tbody>
</table>

Function*

The cubic root of x (REAL).

Graphical Illustration
cub_L (FUN)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cub_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose is raised by the power of three.</td>
</tr>
</tbody>
</table>

*Function*

The cube (third power) of x (LREAL).

**Graphical Illustration**

![Graphical Illustration of cub_L](image)

cub_r (FUN)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cub_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value whose is raised by the power of three.</td>
</tr>
</tbody>
</table>

*Function*

The cube (third power) of x (REAL).

**Graphical Illustration**

![Graphical Illustration of cub_r](image)

exp_L (FUN)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>exp_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value of the exponent.</td>
</tr>
</tbody>
</table>

*Function*

The function value of the exponential function of x (LREAL).

If the value of the result exceeds the range of the return type, ‘cINFTY’ is returned.

**Graphical Illustration**

![Graphical Illustration of exp_L](image)
exp_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>exp_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value of the exponent.</td>
</tr>
</tbody>
</table>

*Function*

The function value of the exponential function of \( x \) (REAL).
If the value of the result exceeds the range of the return type, ‘cINFTY’ is returned.

Graphical Illustration

log10_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>log10_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
</tbody>
</table>

*Function*

The function value of the common logarithm (base-10) of \( x \) (LREAL).
If \( x \) is negative, the constant ‘cNAN’ is returned. If \( x \) is zero, ‘cNINFTY’ is returned.

Graphical Illustration

log10_r (FUN)

Interface variables

3.1. 01 Elementary Math
Function*

The function value of the common logarithm (base-10) of \( x \) (REAL).

If \( x \) is negative, the constant ‘cNAN’ is returned. If \( x \) is zero, ‘cNINFTY’ is returned.

**Graphical Illustration**

\[
\text{Function} \\
\text{log10}_r \\
x \quad \text{REAL} \\
\text{log10}_r
\]

---

**log2_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>log2_L</td>
<td>LREAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
<tr>
<td>Input</td>
<td>( x )</td>
<td>LREAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
</tbody>
</table>

Function*

The function value of the binary logarithm (base-2) of \( x \) (LREAL).

If \( x \) is negative, the constant ‘cNAN’ is returned. If \( x \) is zero, ‘cNINFTY’ is returned.

**Graphical Illustration**

\[
\text{Function} \\
\text{log2}_L \\
x \quad \text{LREAL} \\
\text{log2}_L
\]

---

**log2_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>log2_r</td>
<td>REAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
<tr>
<td>Input</td>
<td>( x )</td>
<td>REAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
</tbody>
</table>

Function*

The function value of the binary logarithm (base-2) of \( x \) (REAL).

If \( x \) is negative, the constant ‘cNAN’ is returned. If \( x \) is zero, ‘cNINFTY’ is returned.

**Graphical Illustration**

\[
\text{Function} \\
\text{log2}_r \\
x \quad \text{REAL} \\
\text{log2}_r
\]
logN_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>logN_L</td>
<td>LREAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
</tbody>
</table>

Function*

The function value of the natural logarithm of x (LREAL).

If x is negative, the constant ‘cNAN’ is returned. If x is zero, ‘cNINFTY’ is returned.

Graphical Illustration

![Function](image)

logN_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>logN_r</td>
<td>REAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value whose logarithm is calculated.</td>
</tr>
</tbody>
</table>

Function*

The function value of the natural logarithm of x (REAL).

If x is negative, the constant ‘cNAN’ is returned. If x is zero, ‘cNINFTY’ is returned.

Graphical Illustration

![Function](image)

nthroot_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>nthroot_L</td>
<td>LREAL</td>
<td>Radicand</td>
</tr>
<tr>
<td>Input</td>
<td>lrRadicand</td>
<td>LREAL</td>
<td>Radicand</td>
</tr>
<tr>
<td></td>
<td>rDegree</td>
<td>REAL</td>
<td>Degree of the root.</td>
</tr>
</tbody>
</table>

Function*

The nth-root of the radicand (LREAL).

If ‘lrRadicand’ is negative, the constant ‘cNAN’ is returned.
Graphical Illustration

nthroot_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>nthroot_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rRadicand</td>
<td>REAL</td>
<td>Radicand</td>
</tr>
<tr>
<td></td>
<td>rDegree</td>
<td>REAL</td>
<td>Degree of the root.</td>
</tr>
</tbody>
</table>

*Function*

The nth-root of the radicand (REAL).
If ‘rRadicand’ is negative, the constant ‘cNAN’ is returned.

Graphical Illustration

pot_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>pot_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrExponent</td>
<td>LREAL</td>
<td>Exponent value</td>
</tr>
</tbody>
</table>

*Function*

Returns ‘lrExponent’ raised to the power of ten (LREAL).

Graphical Illustration
pot_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>pot_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rExponent</td>
<td>REAL</td>
<td>Exponent value</td>
</tr>
</tbody>
</table>

*Function*

Returns ‘rExponent’ raised to the power of ten (REAL).

Graphical Illustration

sqr_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sqr_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose is squared.</td>
</tr>
</tbody>
</table>

*Function*

The square of x (LREAL).

Graphical Illustration

sqr_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sqr_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value whose is squared.</td>
</tr>
</tbody>
</table>

*Function*

The square of x (REAL).

Graphical Illustration
sqrt_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sqrt_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value whose square root is computed.</td>
</tr>
</tbody>
</table>

Function*

The square root of x (LREAL).

If x is negative, the constant ‘cNAN’ is returned.

Graphical Illustration

sqrt_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sqrt_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value whose square root is computed.</td>
</tr>
</tbody>
</table>

Function*

The square root of x (REAL).

If x is negative, the constant ‘cNAN’ is returned.

Graphical Illustration

3.1.2 02 Trignometric

Types and Conversion

Types which has been declared for proper handling of trigonometric functions.

angleToDegree_L (FUN)

Interface variables
WagoAppMath, Release 1.1.1.1

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>angleToDegree_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

*Function*

Converts angle_t to degrees (LREAL).

**Graphical Illustration**

angleToDegree_r (FUN)

*Interface variables*

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>angleToDegree_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

*Function*

Converts angle_t to degrees (REAL).

**Graphical Illustration**

angleToRadiant_L (FUN)

*Interface variables*

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>angleToRadiant_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

*Function*

Converts angle_t to radiant (LREAL).

**Graphical Illustration**
angleToRadiant_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>angleToRadiant_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

Function*

Converts angle_t to radiant (REAL).

angle_t (ALIAS)

Represents an angle for internal calculations.

All trigonometric functions use this alias type for their angle representation. This internal type happens to be 'radians in LREAL', but this may be a subject to changes.

For interfacing with general applications, this library provides conversion functions, which convert the generic angle_t to and from commonly used types, such as:

- degrees (REAL, LREAL, INT)
- radians (REAL and LREAL)

degreeToAngle (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>degreeToAngle</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrDegrees</td>
<td>LREAL</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

Function*

Converts degrees to angle_t.

Graphical Illustration
normalizeAngle_Positive (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>normalizeAngle_Positive</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

Function*
Maps the input angle to the range [0 .. FullCircle].

Graphical Illustration

![Function normalizeAngle_Positive](image1)

normalizeAngle_Signed (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>normalizeAngle_Signed</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

Function*
Maps the input angle to the range [-HalfCircle .. + HalfCircle].

Graphical Illustration

![Function normalizeAngle_Signed](image2)

radiantToAngle (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>radiantToAngle</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrRadiant</td>
<td>LREAL</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

Function*
Converts radiant to angle_t.

Graphical Illustration

![Function radiantToAngle](image3)
arcCos (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>arcCos</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

*Function*

Arc cosine of the input value (LREAL).

*Note* the naming of ‘arcCos’ results out of conflicts with the existing IEC operator ‘ACOS’.

Graphical Illustration

arcSin (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>arcSin</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value to be converted.</td>
</tr>
</tbody>
</table>

*Function*

Arc sine of the input value (LREAL).

*Note* the naming of ‘arcSin’ results out of conflicts with the existing IEC operator ‘ASIN’.

Graphical Illustration

arcTan (FUN)

Interface variables
**arcTan**

Input

- **x**  
  Type: LREAL  
  Value to be converted.

**Function**

Arc tangent of the input value (REAL).

**Note** the naming of ‘arcTan’ results out of conflicts with the existing IEC operator ‘ATAN’.

**Graphical Illustration**

![Function Illustration](arcTan)

**arcTan2**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>arcTan2</td>
<td>angle_t</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>y</td>
<td>LREAL</td>
<td>Value representing the proportion of the x-coordinate.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>LREAL</td>
<td>Value representing the proportion of the y-coordinate.</td>
</tr>
</tbody>
</table>

**Function**

Arc tangent of x and y (LREAL).

**Graphical Illustration**

![Function Illustration](arcTan2)

**Function description**

To compute the value, the function takes into account the sign of both arguments in order to determine the quadrant. If an argument is not a number ‘cNaN’ is returned. If y = x = 0 then ‘cNAN’ is returned.

**cos_L**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cos_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

**Function**

The cosine of angle phi (LREAL).

**Graphical Illustration**

![Function Illustration](cos_L)
cos_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>cos_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

Function*
The cosine of angle phi (REAL).

Graphical Illustration

sin_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sin_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

Function*
The sine of angle phi (LREAL).

Graphical Illustration

sin_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sin_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

Function*
The sine of angle phi (REAL).

Graphical Illustration

```plaintext
Function
sin_r

phi angle_t REAL sin_r
```

tan_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>tan_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

Function*

The tangent of angle phi (LREAL).

Graphical Illustration

```plaintext
Function
tan_L

phi angle_t LREAL tan_L
```

tan_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>tan_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>phi</td>
<td>angle_t</td>
<td>Value represents an angle expressed in radians.</td>
</tr>
</tbody>
</table>

Function*

The tangent of angle phi (REAL).

Graphical Illustration

```plaintext
Function
tan_r

phi angle_t REAL tan_r
```

3.1.3 03 Combinatorics

binomial (FUN)

Interface variables
### Scope | Name | Type | Comment
---|---|---|---
Return | binomial | ULINT | 
Input | uliN | ULINT | ‘n’ part from ‘n over k’
| uliK | ULINT | ‘k’ part from ‘n over k’

**Function**

Binomial coefficient (‘n over k’).

Note: In literature, the input arguments are denoted as ‘N’ and ‘K’ by standard.

Note: Rule: 0 <= ‘K’ <= ‘N’.

**Graphical Illustration**

![Function binomial](image)

#### factorial (FUN)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>factorial</td>
<td>ULINT</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>uliNumber</td>
<td>ULINT</td>
<td>Value to create faculty from.</td>
</tr>
</tbody>
</table>

**Function**

The factorial product (1*2*3*...*n).

**Graphical Illustration**

![Function factorial](image)

**Function description**

For a zero as input, one is returned. For Values which exceeds the range of ULINT, the maximal value of is ULINT returned.

**greatestCommonDivisor (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>greatestCommonDivisor</td>
<td>ULINT</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>uliValueA</td>
<td>ULINT</td>
<td>Value A to process for gcd.</td>
</tr>
<tr>
<td></td>
<td>uliValueB</td>
<td>ULINT</td>
<td>Value B to process for gcd.</td>
</tr>
</tbody>
</table>

**Function**

The greatest common divisor (gcd) of both input arguments (ULINT).
Graphical Illustration

```plaintext
Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>leastCommonMultiple</td>
<td>ULINT</td>
<td>Value A to process lcm.</td>
</tr>
<tr>
<td>uliValueA</td>
<td>ULINT</td>
<td></td>
</tr>
<tr>
<td>uliValueB</td>
<td>ULINT</td>
<td></td>
</tr>
</tbody>
</table>
```

**leastCommonMultiple (FUN)**

**Function**

Find the least common multiple (lcm) of both input arguments (LREAL).

Graphical Illustration

```plaintext
Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsInfinite</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>LREAL</td>
<td></td>
</tr>
</tbody>
</table>
```

**IsInfinite (FUN)**

**Function**

Returns TRUE if the argument is not a finite number but NaN, NINFTY, or INFTY.

Graphical Illustration
IsNaN (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>IsNaN</td>
<td>BOOL</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
</tr>
</tbody>
</table>

Function*

Returns TRUE if the argument is NaN.

Graphical Illustration

abs_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>abs_L</td>
<td>LREAL</td>
<td>Value to obtain magnitude.</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td></td>
</tr>
</tbody>
</table>

Function*

The magnitude value of x (LREAL).

Graphical Illustration

abs_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>abs_r</td>
<td>REAL</td>
<td>Value to obtain magnitude.</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td></td>
</tr>
</tbody>
</table>
The magnitude value of \( x \) (REAL).

**Graphical Illustration**

```
Function

ceil_L

\[
x \text{ REAL} \rightarrow \text{ ceil_L}
\]
```

**ceil_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>ceil_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>( x )</td>
<td>LREAL</td>
<td>Value to round up.</td>
</tr>
</tbody>
</table>

*Function*

Rounds \( x \) upwards to next integral value (LREAL).

**Graphical Illustration**

```
Function

ceil_L

\[
x \text{ LREAL} \rightarrow \text{ ceil_L}
\]
```

**Function description**

This function Rounds \( x \) with a fraction upwards to next integral value (LREAL).

If the value is integral, the current value is returned.

**ceil_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>ceil_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>( x )</td>
<td>REAL</td>
<td>Value to round up.</td>
</tr>
</tbody>
</table>

*Function*

The tangent of angle \( \phi \) (REAL).

**Graphical Illustration**

```
Function

ceil_r

\[
x \text{ REAL} \rightarrow \text{ ceil_r}
\]
```

**Function description**

This function Rounds a value with a fraction upwards to next integral value (REAL).
If the value is integral, the current value is returned.

**copySign_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>copySign_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrMagnitude</td>
<td>LREAL</td>
<td>Value to obtain magnitude for return value.</td>
</tr>
<tr>
<td></td>
<td>lrSign</td>
<td>LREAL</td>
<td>Value to obtain sign for return value.</td>
</tr>
</tbody>
</table>

*Function*

Returns a value with the magnitude and sign of the input arguments (LREAL).

**Graphical Illustration**

![Graphical Illustration of copySign_L](image)

*Function description*

This functions returns a value which assembled out the magnitude of the first argument ‘lrMagnitude’ and the sign of the second argument ‘lrSign’.

**copySign_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>copySign_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rMagnitude</td>
<td>REAL</td>
<td>Value to obtain magnitude for return value.</td>
</tr>
<tr>
<td></td>
<td>rSign</td>
<td>REAL</td>
<td>Value to obtain sign for return value.</td>
</tr>
</tbody>
</table>

*Function*

Returns magnitude and sign of the input arguments (REAL).

**Graphical Illustration**

![Graphical Illustration of copySign_r](image)

*Function description*

This functions returns a value which assembled out the magnitude of the first argument ‘rMagnitude’ and the sign of the second argument ‘rSign’.
div_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>div_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrNominator</td>
<td>LREAL</td>
<td>Dividend - what is to be divided</td>
</tr>
<tr>
<td></td>
<td>lrDenominator</td>
<td>LREAL</td>
<td>Divisor - by what is divided</td>
</tr>
</tbody>
</table>

Function*

Divides without throwing exceptions (REALs).

Graphical Illustration

![Function div_L](div_L_diagram.png)

Function description

Divides the nominator (dividend) by the denominator (divisor).

If the divisor is zero, this function does not throw exceptions, but instead returns Not-A-Number (NaN). If the divisor is non-zero but the result would be too large otherwise, ‘infinity’ or ‘negative infinity’ is returned.

div_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>div_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rNominator</td>
<td>REAL</td>
<td>Dividend - what is to be divided</td>
</tr>
<tr>
<td></td>
<td>rDenominator</td>
<td>REAL</td>
<td>Divisor - by what is divided</td>
</tr>
</tbody>
</table>

Function*

Divides without throwing exceptions (REALs).

Graphical Illustration

![Function div_r](div_r_diagram.png)

Function description

Divides the nominator (dividend) by the denominator (divisor).

If the divisor is zero, this function does not throw exceptions, but instead returns Not-A-Number (NaN). If the divisor is non-zero but the result would be too large otherwise, ‘infinity’ or ‘negative infinity’ is returned.
floor_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>floor_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value to round down.</td>
</tr>
</tbody>
</table>

Function*

Rounds x downwards to next integral value (LREAL).

Graphical Illustration

Function description

This function rounds x with a fraction downwards to next integral value (LREAL).
If the value is integral, the current value is returned.

floor_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>floor_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value to round down.</td>
</tr>
</tbody>
</table>

Function*

Rounds x downwards to next integral value (REAL).

Graphical Illustration

Function description

This function rounds x with a fraction downwards to next integral value (REAL).
If the value is integral, the current value is returned.

fmod_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>fmod_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrNumer</td>
<td>LREAL</td>
<td>Value of the numerator.</td>
</tr>
<tr>
<td></td>
<td>lrDenom</td>
<td>LREAL</td>
<td>Value of the denominator.</td>
</tr>
</tbody>
</table>
Returns the fraction of the modulus division (LREAL).
If the denominator is zero, the constant ‘cNAN’ is returned.

**Function**

Returns the fraction of the modulus division (LREAL).
If the denominator is zero, the constant ‘cNAN’ is returned.

**Graphical Illustration**

![Graphical Illustration of fmod_r function]

**fmod_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>fmod_r</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rNumerator</td>
<td>REAL</td>
<td>Value of the numerator.</td>
</tr>
<tr>
<td></td>
<td>rDenom</td>
<td>REAL</td>
<td>Value of the denominator.</td>
</tr>
</tbody>
</table>

*Function*

Returns the fraction of the modulus division (REAL).
If the denominator is zero, the constant ‘cNAN’ is returned.

**Graphical Illustration**

![Graphical Illustration of fpSplit_L function]

**fpSplit_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td></td>
<td>Value to break into parts.</td>
</tr>
<tr>
<td>Output</td>
<td>lrIntegral</td>
<td>LREAL</td>
<td></td>
<td>Value for integral part.</td>
</tr>
<tr>
<td></td>
<td>lrFraction</td>
<td>LREAL</td>
<td>0</td>
<td>Value for fractional part.</td>
</tr>
</tbody>
</table>

*Function*

Breaks x into an integral and a fractional part (LREAL).

**Graphical Illustration**
Function description

When an argument becomes ‘cINFTY’ or ‘cNAN’. Theses constants are returned for the integral part. The fractional part is set to zero.

**fpSplit_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td>Value to break into parts.</td>
</tr>
<tr>
<td>Output</td>
<td>rIntegral</td>
<td>REAL</td>
<td>Value for integral part.</td>
</tr>
<tr>
<td></td>
<td>rFraction</td>
<td>REAL</td>
<td>Value for fractional part.</td>
</tr>
</tbody>
</table>

*Function*

Breaks x into an integral and a fractional part (REAL).

**Graphical Illustration**

---

Function description

When an argument becomes ‘cINFTY’ or ‘cNAN’. Theses constants are returned for the integral part. The fractional part is set to zero.

**isWithinTolerance_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>lrArgumentA</td>
<td>LREAL</td>
<td>Value A</td>
</tr>
<tr>
<td></td>
<td>lrArgumentB</td>
<td>LREAL</td>
<td>Value B</td>
</tr>
<tr>
<td></td>
<td>lrTolerance</td>
<td>LREAL</td>
<td>Tolerance to compare with.</td>
</tr>
</tbody>
</table>

*Function*

Compares the difference of A and B with a tolerance and returns a boolean parameter (LREAL).

**Graphical Illustration**
Function description

If the difference of A and B is less or equal as the tolerance, True is returned.

If the difference of A and B is larger as the tolerance or an argument becomes ‘Infinity’ or ‘Not a Number’, False is returned as well.

**isWithinTolerance_r (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>isWithinTolerance_r</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>rArgumentA</td>
<td>REAL</td>
<td>Value A</td>
</tr>
<tr>
<td></td>
<td>rArgumentB</td>
<td>REAL</td>
<td>Value B</td>
</tr>
<tr>
<td></td>
<td>rTolerance</td>
<td>REAL</td>
<td>Tolerance to compare with.</td>
</tr>
</tbody>
</table>

*Function*

Compares the difference of A and B with a tolerance and returns a boolean parameter (REAL).

**Graphical Illustration**

Function description

If the difference of A and B is less or equal as the tolerance, True is returned.

If the difference of A and B is larger as the tolerance or an argument becomes ‘Infinity’ or ‘Not a Number’, False is returned as well.

**nearest_L (FUN)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>nearest_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>LREAL</td>
<td>Value to round either up or down.</td>
</tr>
</tbody>
</table>

*Function*

Rounds x to the closest integral value (LREAL).

**Graphical Illustration**
nearest_r (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>nearest_r</td>
<td>REAL</td>
<td>Value to round either up or down.</td>
</tr>
<tr>
<td>Input</td>
<td>x</td>
<td>REAL</td>
<td></td>
</tr>
</tbody>
</table>

*Function*

Rounds x to the closest integral value (REAL).

Graphical Illustration

sign_L (FUN)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>sign_L</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>lrValue</td>
<td>LREAL</td>
<td>Value to obtain sign from.</td>
</tr>
</tbody>
</table>

*Function*

Returns the sign of the input argument (LREAL).

Graphical Illustration

Function description

Possible results: -1,0,+1

If ‘lrValue’ is undefined, the constant ‘cNAN’ is returned.

sign_r (FUN)

Interface variables
### Function

Returns the sign of the input argument (REAL).

#### Graphical Illustration

![Graphical Illustration](image)

#### Function description

Possible results: -1,0,+1

If ‘rValue’ is undefined, the constant ‘cNAN’ is returned.
### Constants (GVL)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cPi</td>
<td>LREAL</td>
<td>3.1415926535897931</td>
<td>Pi constant</td>
</tr>
<tr>
<td></td>
<td>cEuler</td>
<td>LREAL</td>
<td>2.7182818284590451</td>
<td>Euler constant</td>
</tr>
<tr>
<td></td>
<td>cEpsilon_L</td>
<td>LREAL</td>
<td>2.22E-16</td>
<td>Epsilon (relative error) for floating point</td>
</tr>
<tr>
<td></td>
<td>cEpsilon_r</td>
<td>REAL</td>
<td>1.19E-07</td>
<td>Epsilon (relative error) for floating point</td>
</tr>
<tr>
<td></td>
<td>cFP_Min_L</td>
<td>LREAL</td>
<td>2.2E-308</td>
<td>The smallest floating point LREAL</td>
</tr>
<tr>
<td></td>
<td>cFP_Max_L</td>
<td>LREAL</td>
<td>1.8E+307</td>
<td>The largest floating point LREAL</td>
</tr>
<tr>
<td></td>
<td>cFP_Min_r</td>
<td>REAL</td>
<td>2.2E-45</td>
<td>The smallest floating point REAL</td>
</tr>
<tr>
<td></td>
<td>cFP_Max_r</td>
<td>REAL</td>
<td>1.8E+38</td>
<td>The largest floating point REAL</td>
</tr>
<tr>
<td></td>
<td>cNINFTY</td>
<td>REAL</td>
<td>LREAL_TO_REAL(LOG(0))</td>
<td>Negative infinity</td>
</tr>
<tr>
<td></td>
<td>cINFTY</td>
<td>REAL</td>
<td>LREAL_TO_REAL((INT#0 - LOG(0)))</td>
<td>Positive infinity</td>
</tr>
<tr>
<td></td>
<td>cNAN</td>
<td>REAL</td>
<td>LREAL_TO_REAL(SQR(1))</td>
<td>Not a number</td>
</tr>
</tbody>
</table>
LibraryResult (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>FbResultFactory</td>
</tr>
</tbody>
</table>

Factory for standard result objects

Use this to translate result codes from this library into standard result objects.

Usage:

```
VAR
  eMyResult : eResultCode; // result code which is to be investigated
  oError    : FbResult;    // result object for use in higher levels.
END_VAR;

eMyResult := myFunction(...);
Namespace.LibraryResult.Factory.SetResult(eMyResult, oError);
```

where ‘Namespace’ denotes the namespace which is used for including the specific library and ‘myFunction()’ is an example for a general function from this library.
ResultItems (GVL)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>ERROR</td>
<td>ARRAY [0..0] OF typResultItem</td>
<td>[STRUCT(ID := OK, Severity := eSeverity.none, Text := ‘OK’)]</td>
</tr>
</tbody>
</table>

Standard result items specific for this library

Note: This is a general mapping of result codes to short standard texts which are appropriate to the usage of these codes in this library.

Typically, each unit (function, method, or function block) in this library uses only a subset of these codes. Please, refer to the documentation of the specific unit for the set of codes which is actually used and for a detailed explanation of the meaning of a result code in the specific context.

**Note():** This library does not yet use an error codes.
## VersionHistory (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoAppMath</td>
<td>ProjectInfo</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.03.2019</td>
<td>1.1.1.1</td>
<td>WAGO / u013972</td>
<td>Resolve libraries as placeholder</td>
</tr>
<tr>
<td>08.01.2019</td>
<td>1.1.1.0</td>
<td>u015842</td>
<td>Properties: free placeholder added</td>
</tr>
<tr>
<td>20.20.2017</td>
<td>1.1.0.0</td>
<td>WAGO / u013972</td>
<td>Remove Controller-Functions</td>
</tr>
<tr>
<td>09.03.2016</td>
<td>1.0.0.0</td>
<td>WAGO / u013972</td>
<td>Release Version</td>
</tr>
</tbody>
</table>
Library Reference

This is a dictionary of all referenced libraries and their name spaces.

**CAA Mathematics**

*Library Identification:*

Placeholder: CAA Mathematics

Default Resolution: CAA Mathematics, * (CAA Technical Workgroup)

Namespace: MATH

*Library Properties:*

- LinkAllContent: False
- QualifiedOnly: True
- SystemLibrary: False
- Optional: False

**FloatingPointUtils**

*Library Identification:*

Placeholder: FloatingPointUtils

Default Resolution: FloatingPointUtils, * (System)

Namespace: FPU

*Library Properties:*

- LinkAllContent: False
- QualifiedOnly: True
- SystemLibrary: False
- Optional: False

**Standard**

*Library Identification:*

Placeholder: Standard

Default Resolution: Standard, * (System)

Namespace: Standard

*Library Properties:*

-
Util

**Library Identification:**
Placeholder: Util
Default Resolution: Util, * (System)
Namespace: Util

**Library Properties:**
- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

**Library Parameter:**
Parameter: IBLOCKSIZE = 22800

WagoSysErrorBase

**Library Identification:**
Placeholder: WagoSysErrorBase
Default Resolution: WagoSysErrorBase, * (WAGO)
Namespace: WagoSysErrorBase

**Library Properties:**
- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

**Library Parameter:**
Parameter: RES_LOG_MAX_FILESIZE = 2000
Parameter: RES_LOG_MAX_FILES = 1
Parameter: RES_LOG_MAX_ENTRIES = 200
Parameter: RES_LOG_NAME = 'WagoAppResultLogger'

WagoSysPlainMem

**Library Identification:**
Placeholder: WagoSysPlainMem
Default Resolution: WagoSysPlainMem, * (WAGO)
Namespace: WagoSysPlainMem
Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

WagoSysVersion

Library Identification:
Name: WagoSysVersion
Version: 1.0.0.0
Company: WAGO
Namespace: WagoSysVersion

Library Properties:

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

WagoTypesCommon

Library Identification:
Placeholder: WagoTypesCommon
Default Resolution: WagoTypesCommon, * (WAGO)
Namespace: WagoTypes

Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

© WAGO Kontakttechnik GmbH & Co. KG, Germany 2018 – All rights reserved. For the avoidance of doubt, this copyright notice does not only apply to the information above but also and primarily to the described library itself. Please note that third-party products are always mentioned without reference to intellectual property rights, including patents, utility models, designs and trademarks, accordingly the existence of such rights cannot be excluded. WAGO is a registered trademark of WAGO Verwaltungsgesellschaft mbH.