18.3 Mathematical functions

Signal value might require some recalculation or signal update prior to being sent. Understandably, existing columns in Excel configuration like `multiply`, `add`, `bit_select` might not be flexible enough. To overcome these limitations, symbolic mathematical expressions can be configured to do calculations automatically on every update of a signal.

> It should be noted that filling mathematical expression disables other mathematical scalar operations for a single value such as `multiply`, `add` or `bit_select`. Other functions (primarily between several signals) are still available such as operation.

Feature list:

- Optimized for speed
  - High parsing performance
  - if-then-else operator with lazy evaluation
- Default implementation with many features
  - 25 predefined functions
  - 18 predefined operators
- Unit support
  - Use postfix operators as unit multipliers (3m -> 0.003)

Mathematical functions

Table. Supported mathematical functions:

Name	Argument count	Explanation
sin	1	sine function (rad)
cos	1	cosine function (rad)
tan	1	tangent function (rad)
asin	1	arcus sine function (rad)
acos	1	arcus cosine function (rad)
atan	1	arcus tangens function (rad)
sinh	1	hyperbolic sine function
cosh	1	hyperbolic cosine
tanh	1	hyperbolic tangens function
asinh	1	hyperbolic arcus sine function
acosh	1	hyperbolic arcus tangens function
atanh	1	hyperbolic arcur tangens function
log2	1	logarithm to the base 2
log10	1	logarithm to the base 10

log	1	logarithm to base e (2.71828...)
ln	1	logarithm to base e (2.71828...)
exp	1	e raised to the power of x
sqrt	1	square root of a value
sign	1	sign function -1 if x<0; 1 if x>0
rint	1	round to nearest integer
abs	1	absolute value
min	variable	min of all arguments
max	variable	max of all arguments
sum	variable	sum of all arguments
avg	variable	mean value of all arguments
floor	1	round down to the nearest integer

It should be noted that trigonometric functions (excluding hiperbolic functions) only support arguments in radians. This means that arguments for this function have to be recalculated if angle is defined in degress.

Value recalculation is only triggered on signal change of the preconfigured signal. That means that using other signals (via TagValue() call) does not trigger value update.

Some mathematical expression cannot be mathematically evaluated in some conditions, for example, square root cannot be found for negative numbers. As complex numbers are not supported, result is then equal to Not a Number (NaN). These results are marked with an invalid (IV) flag.

Binary operations

Table. Supported binary operators:

Operator	Description	Priority
=	assignment	-1
»	right shift	0
«	left shift	0
&	bitwise and	0
|	bitwise or	0
&&	logical and	1
||	logical or	2

<=	less or equal	4
>=	greater or equal	4
!=	not equal	4
==	equal	4
>	greater than	4
<	less than	4
+	addition	5
-	subtraction	5
*	multiplication	6
%	modulo	6
/	division	6
^	raise x to the power of y	7

Ternary operators can be used. This expression can be compared to the operator supported by C/C++ language (Table 39). Condition is written before a question (?) sign. If condition is true, result after question sign is selected. If condition is false, result after colon (:) is selected.

Ternary operations

Table. Supported ternary operators

Operator	Description	Remarks
?:	if then else operator	C++ style syntax

Examples

Users can construct their own equation by using the aforementioned operators and functions. These examples can be seen in Table below.

Table. Example expressions

Expression	Description
value * 0.0001	Multiply the tag by a constant.
value + TagValue(”tag/dev_alias/sig_alias/out”)	Add value of tag/dev_alias/sig_alias/out to the current tag.
sin(value)	Return a predefined sine function value of the tag.
(value>5)? 1: 0	If the value is greater than 5, the result should be equal to 1, otherwise - equal to 0

Variable called value is generated or updated on every signal change and represents the signals being configured. If another value from tag list is intended to be used, one should use `TagValue()` function to retrieve its last value.

The inner argument of `TagValue()` function has to described in a Redis topic structure of WCC Lite. That means that it has to be constructed in a certain way. Quotes should be used to feed the topic name value, otherwise expression evaluation will fail.

Every Redis topic name is constructed as tag/[device_alias]/[signal_alias]/[direction]. Prefix tag/ is always used before the rest of argument. `device_alias` and `signal_alias` represent columns in Excel configuration. direction can have one of four possible values - rout, out, in, rin; all of which depend on the direction data is sent or acquired device-wise. For example, out keyword marks data sent out of WCC Lite device, whereas in direction represents data that WCC Lite is waiting to receive, for example, commands. Additional r before either direction means that data is raw, it was is presented the way it was read by an individual protocol.

Extra functions

Several functions are defined make tag operations possible:

- `TagValue(key)` - returns last known value of tag identified by redis key;
- `TagFlag(key)` - returns 1 if tag flag exists. Name format is: ”key flag”. For example to check if tag is notopical, name would be ”tag/19xxxxxxx/x/x nt”;
- `TagAttribute(key)` - similar to TagFlag, but returns a numeric value of a tag attribute;
- `TagTime(key)` - returns UNIX timestamp in milliseconds of a last know tag value.

Revision #3
★Created 7 October 2022 10:29:16 by Lukas Taroza
Updated 20 December 2022 12:27:32 by Lukas Taroza