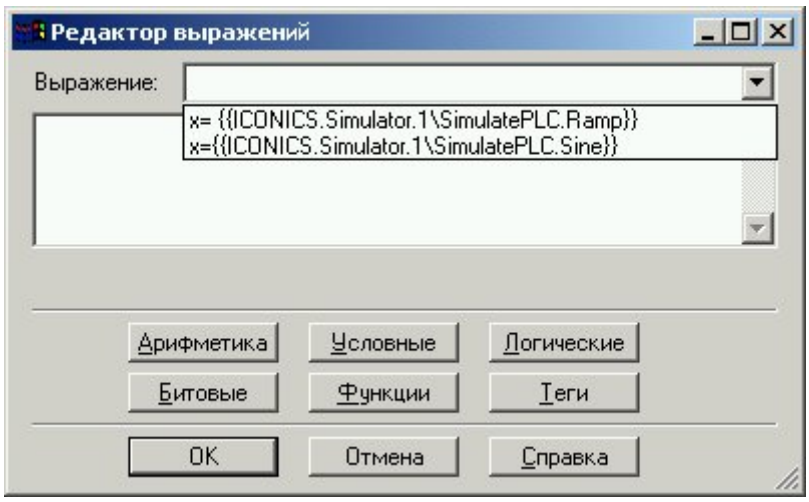
ICONICS
100 Foxborough Blvd.
Foxborough, MA 02035
Tel: 508-543-8600
Fax: 508-543-1503
E-Mail: support@iconics.com
Web: www.iconics.com

Содержание

Создание выражений с использованием Редактора выражений.

Редактор выражений, показанный на рисунке, позволяет создавать выражения для приложений GENESIS32.
Благодаря возможности изменения размера окна **Редактора выражений**, можно выбрать наиболее удобный для пользователя вид просмотра выражений. Выпадающий список для поля **Выражение** позволяет выбрать одно из 50 последних введенных выражений. Первым в выпадающем списке будет наиболее часто используемое выражение.



Выражения

Выражения представляют собой строки, начинающиеся с символов “x=”, например:
x={{ICONICS.Simulator.1\SimulatePLC.PumpSpeed}}
Есть возможность ввести выражение сразу в текстовое поле **Выражение** диалогового окна **Редактор выражений**, а также можно воспользоваться символами и функциями, которые помогут сохранить верный синтаксис выражения в целом.
Имеются следующие категории выражений:

- Арифметика
- Сравнения (Условные)
- Логические
- Битовые
- Функции
- Теги

Строки в выражениях

Строковые константы, используемые в выражениях, выделяются двойными кавычками, например: "Hello World"
Также строковые константы могут быть использованы в виде:
$"Hello World"$

Сравнения строк

Когда сравниваемые строки или числовые данные, приходящие с сервера в виде строк, сравниваться в выражениях они будут на основе порядка следования символов.
Например, результат выражения:
x="world" > "hello"
будет «истина», т.к. символ w следует за h (в алфавитном порядке).
Иногда сравнение строк может иметь неодиданный результат, например, результатом выражения:
x="20" > "100" также будет «истина», т.к. символ «2» следует в таблице символов после «1».

Преобразование типа данных

OPC серверы предоставляют различные типы данных, такие как "float," "long," "integer," "string". Если числовые данные приходят от сервера в виде строк, сравниваться в выражениях они будут тоже как строки, на основе порядка символов. Поэтому выражение "20" > "100" будет иметь значение «истина». Естественно, если ожидается сравнение числовых данных, результат будет некорректным. Один из путей решения этой проблемы заключается в прибавлении числа нуль к значению тега. В этом случае логические операции будут выполняться правильно. Пример:
x=({{JC.N1OPC.1.0\HDQTRS\sys2\ad-3.Present Value}}+0) >

({{JC.N1OPC.1.0\HDQTRS\sys2\ad-4.Present Value}}+0)

Другой путь – изменить OPC-сервер так, что он будет посылать строки с фиксированным количеством цифр и ведущими нулями, или использовать регистры DataWorX32 для преобразования строки в число.

Некоторые функции **Редактора выражений** используют числовые параметры. Когда возможно, Редактор выражений автоматически преобразует строку в число. Например: Строка "20" может быть автоматически переведена в число 20.
Если строковая константа содержит символы алфавита, подобное автоматическое преобразование невозможно. Например: Строка "20hello" не может быть преобразована в число. Даже строка типа "123.45.23" не будет преобразована в числовое значение, т.к. содержит две точки. Иногда выражение содержит и строки и числа. В этом случае **Редактор выражений** преобразует строку в число. Например:
Str+число, где Str – строка, которая будет переведена в число.
Если строка не может быть преобразована в числовое значение, результат будет иметь плохое качество**.** Ниже приведен пример правильного выражения:
x=5+"6"

Синтаксис расширения тега

Синтаксис расширения тега позволяет получать дополнительные параметры OPC-тега, такие как quality (качество) и timestamp (метка времени).
Примеры организации запроса с Синтаксисом расширения:
tag:ICONICS.Simulator\SimulatePLC.Ramp#timestamp
tag:ICONICS.Simulator\SimulatePLC.Ramp#quality
tag:\\pc1\ICONICS.Simulator\SimulatePLC.Ramp#timestamp
tag:\\pc1\ICONICS.Simulator\SimulatePLC.Ramp#quality
Для дополнительной информации о качестве OPC-тегов, см.раздел «Качество».
Иногда может быть необходимо устанавливать определенный «Запрашиваемый тип данных», напр. «строка», для отображения информации в качестве параметра.

Арифметические операции

Меню арифметических операций показано на рисунке.

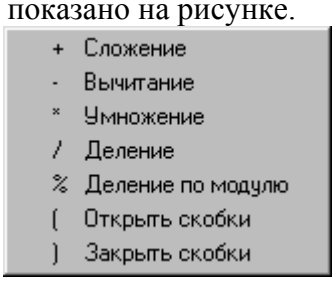


Рис. Арифметические операции

Для вызова меню символов арифметических операций следует нажать кнопку **Арифметика** диалоговой панели **Редактор выражений**.
Используется следующий формат для символов арифметических операций:
Выражение :: Операнд1 **Символ** Операнд2,
где
Операнд1,2 – локальные переменные, псевдонимы, теги OPC, константы или другие выражения
Символ – символ операции: ‘+’, ‘–’, ‘/’, ‘*’, ‘%’
Результат:
Результатом выражения является значение любого типа (действительное, целое и т.п.).
Примеры:

Символ операции	Описание	Пример	Результат
+	Сложение	~~var1~~ + ~~var2~~	9+3=12
–	Вычитание	~~var1~~ – ~~var2~~	9–3=6
*	Умножение	~~var1~~ * ~~var2~~	9*3=27
/	Деление	~~var1~~ / ~~var2~~	9/3=3
%	Деление по модулю	~~var1~~ % ~~var2~~	9%4=1
(..)	Определяет приоритет вычисления для выражения, заключенного в скобки	~~var1~~ /(~~var2~~ + ~~var3~~)	8/(3+2)=1,6

Операции сравнения

Меню операций сравнения показано на рис. 7-6.

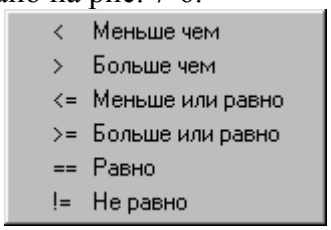


Рис. 7-1. Меню операций сравнения

Для вызова меню символов операций сравнения следует нажать кнопку **Условные** диалоговой панели **Редактор выражений**.
Используется следующий формат для символов операций сравнения:
Выражение :: Операнд1 **Символ** Операнд2,
где
Операнд1,2 – локальные переменные, псевдонимы, теги OPC, константы или другие выражения
Символ – символ операции: ‘<’, ‘>’, ‘<=’, ‘>=’, ‘==’, ‘!=’
Результат:
Результатом выражения является булево значение (0 или 1).
Примеры:

Символ операции	Описание	Пример	Результат
<	Меньше чем	~~var1~~ < ~~var2~~	9<3 = 0
>	Больше чем	~~var1~~ > ~~var2~~	9>3 = 1
<=	Меньше или равно	~~var1~~ <= ~~var2~~	9<=3 = 0
>=	Больше или равно	~~var1~~ >= ~~var2~~	9>=3 = 1
==	Равно	~~var1~~ == ~~var2~~	9==4 = 0
!=	Не равно	~~var1~~ != ~~var2~~	8!=2 = 1

Логические операции

Меню логических операций показано на рис. 7-7.

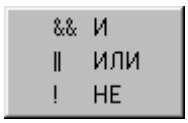


Рис. 7-2. Меню логических операций

Для вызова меню символов логических операций следует нажать кнопку **Логические** диалоговой панели **Редактор выражений**.

Используются следующие форматы для символов логических операций:

Выражение :: Операнд1 **Символ** Операнд2,

где

Операнд1,2 – локальные переменные, псевдонимы, теги OPC, константы или другие выражения

Символ – символ операции: ‘&&’, ‘||’

Для операции НЕ используется следующий формат:

Выражение :: **Символ** Операнд1,

где

Операнд1 – локальная переменная, тег OPC, константа или другие выражения

Символ – символ операции: ‘!’

Результат:

Результатом выражения является булево значение (0 или 1).

Таблица истинности логических операций

~~var1~~	0		1	
~~var2~~	0	1	0	1
~~var1~~ && ~~var2~~	0	0	0	1
~~var1~~ || ~~var2~~	0	1	1	1
! ~~var2~~	1	0	1	0

Примеры:

Символ операции	Описание	Пример	Результат
&&	Логическое И	~~var1~~ && ~~var2~~	9&&3 = 1
||	Логическое ИЛИ	~~var1~~ || ~~var2~~	9||3 = 1
!	Логическое НЕ	! ~~var1~~	!9 = 0

Битовые операции

Меню битовых операций показано на рисунке.

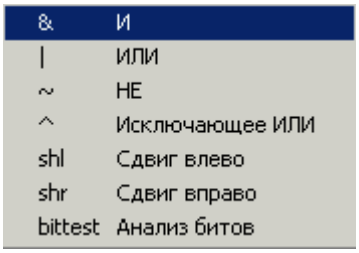


Рис. Меню битовых операций

Для вызова меню символов битовых операций следует нажать кнопку **Битовые** диалоговой панели **Редактор выражений**.

Используются следующие форматы для символов битовых операций:

Выражение :: Операнд1 **Символ** Операнд2,

где

Операнд1,2 – локальные переменные, псевдонимы, теги OPC, константы или другие выражения

Символ – символ операции: ‘&’, ‘|’, ‘^’

Для операции НЕ используется следующий формат:

Выражение :: **Символ** Операнд1,
где
Операнд1 – локальная переменная, тег OPC, константа или другие выражения
Символ – символ операции: ‘~’
Для операции сдвига влево и вправо используется следующий формат:
Выражение :: **Символ**(Операнд1, Сдвиг)
где
Операнд1 – локальная переменная, тег OPC, константа или другое выражение
Символ – символ операции: ‘shl’, ‘shr’
Сдвиг – количество бит, на которое требуется выполнить сдвиг.
Результат:
Результатом выражения является целое значение.
Для операции анализа битов используется следующий формат:
BitTest (Операнд1, Позиция),
где
Операнд1 – локальная переменная, тег OPC, константа или другое выражение
Позиция – номер проверяемого бита. «0»- самый правый бит.
Результат:
Результатом выражения является булево значение (0 или 1). «1» -если проверяемый бит равен 1, в противном случае, результат «0».

Таблица примеров использования битовых операций

~~var1~~	0000 0000 0000 1000 (8)	0000 0000 0110 0000 (96)
~~var2~~	0000 0000 0000 1010 (10)	0000 0000 0000 1000 (8)
~~var1~~ & ~~var2~~	0000 0000 0000 1000 (8)	0000 0000 0000 0000 (0)
~~var1~~ | ~~var2~~	0000 0000 0000 1010 (10)	0000 0000 0110 1000 (104)
~~var1~~ ^ ~~var2~~	0000 0000 0000 0010 (2)	0000 0000 0110 1000 (104)
~ (~~var2~~)	1111 1111 1111 0101 (-11)	1111 1111 1111 0111 (-9)
Shr(~~var1~~,3)	0000 0000 0000 0001 (1)	0000 0000 0000 1100 (12)
Shl(~~var1~~,3)	0000 0000 0100 0000 (64)	0000 0011 0000 0000 (768)
bittest(~~var1~~,3)	0000 0000 0000 0001 (1)	0000 0000 0000 0000 (0)

Примеры:

Символ операции	Описание	Пример	Результат
&	Поразрядное И	~~var1~~ & ~~var2~~	8&3 = 0
|	Поразрядное ИЛИ	~~var1~~ | ~~var2~~	8|3 = 11
^	Исключающее ИЛИ	~~var1~~ ^ ~~var2~~	8^3 = 11
shl	Поразрядный сдвиг влево	Shl(~~var1~~, 3)	Shl(8,3) = 64
shr	Поразрядный сдвиг вправо	Shr(~~var1~~, 3)	Shr(8,3) = 1
~	Поразрядное НЕ	~(~~var1~~)	~ 8 = -9
bittest	Анализ битов	Bittest(5,0)	1

Функциональные операции

Меню функций показано на рисунке.

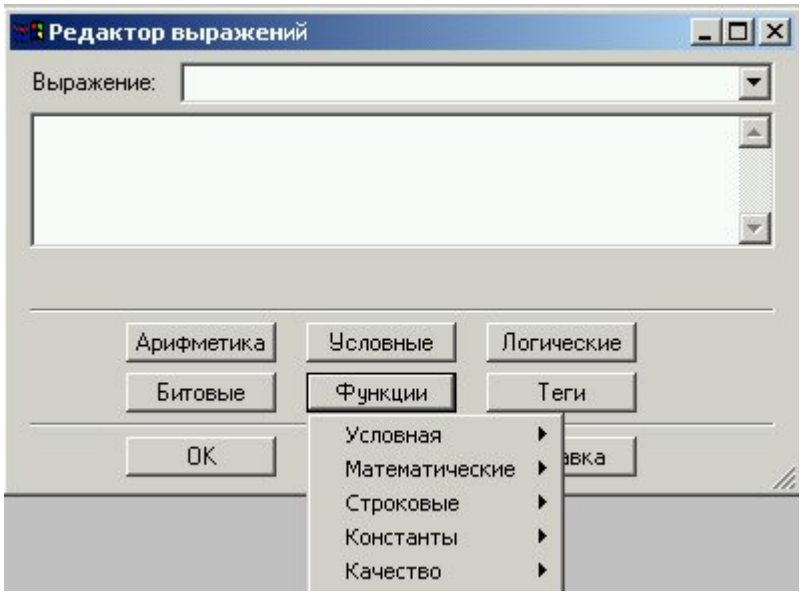


Рис.Меню функций

Условная функция:

Символ ‘if’ - условный оператор, который использует следующий формат
symbol (parameter,parameter,parameter)

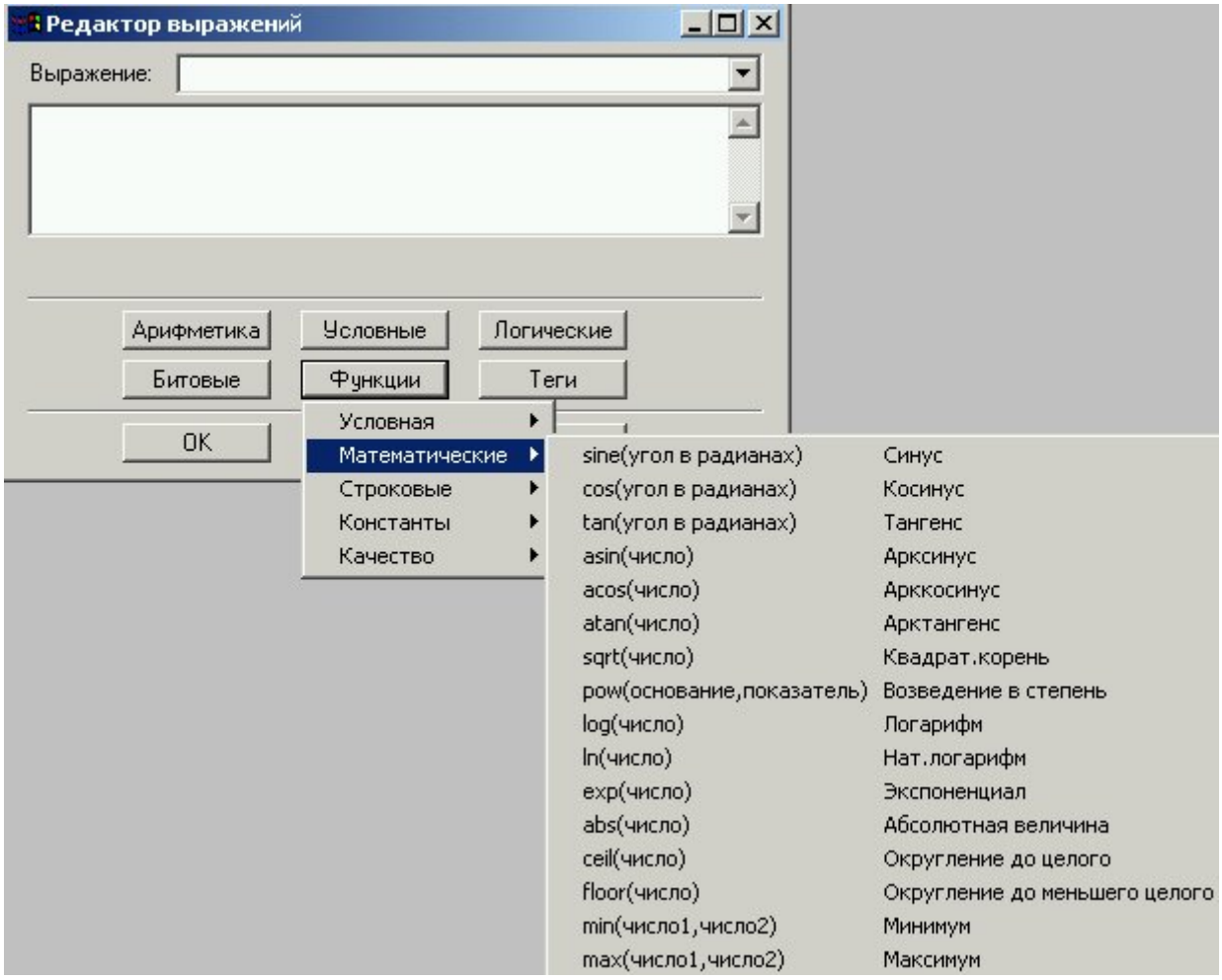
Функция ‘if’ используется для вычисления выражения и определения величины как результата выражения и имеет следующий синтаксис:
Результат=If(Условие, значение1, значение2)
где
Значение 1 – результат, если Условие выполняется («истина»)
Значение 2 – результат, если Условие не выполняется («ложь»)
Условие, Значение 1, Значение 2 могут быть константами, переменными, тегами OPC или выражениями.
Следует отметить, что числовое значение 0 будет восприниматься как «ложь», все другие значения – как «истина».
В таблице ниже приведены примеры использования функции ‘if’:

Функция	Комментарии
If (~~var~~ > 0, 1, 0)	Если локальная переменная ~~var~~ является положительным числом, результат выражения =1. В любом другом случае, результат =0
If (~~var~~<0, 10, 55)	Если локальная переменная ~~var~~ является отрицательным числом, результат выражения =10. В любом другом случае, результат =55
If (~~pressure~~>100, ~~level~~, ~~pumpstatus~~	Если локальная переменная ~~pressure~~ больше 100, результат выражения будет равен ~~level~~. В любом другом случае, результат будет равен ~~pumpstatus~~.

If (~~pump_on~~==1, 1, max (~~var1~~,~~var2~~)	Если величина ~~pump_on~~ равна 1, результат выражения будет равен 1. В любом другом случае, результат будет равен максимальному из двух величин ~~var1~~ и~~var2~~.
If ({{MYTAG}} >=~~min~~ , {{MYTAG}}, ~~min~~	Если значение тега MITAG больше или равно значению локальной переменной ~~min~~, результат выражения будет равен значению самого тега. В любом другом случае, результат будет равен ~~min~~.

Математические функции

Математические функции показаны на рисунке:



Операторы функциональных операций ‘sin’, ‘asin’, ‘cos’, ‘acos’, ‘tan’, ‘atan’, ‘log’, ‘ln’, ‘exp’, ‘sqrt’, ‘abs’, ‘ceil’, ‘floor’ используются в выражениях следующим образом:

Выражение :: **Оператор**(Аргумент)
где
Аргумент – локальная переменная, псевдоним, тег OPC, константа или другие выражения
Операторы функциональных операций ‘pow’, ‘min’, ‘max’ используются в выражениях следующим образом:
Выражение :: **Оператор**(Аргумент1, Аргумент2)
где
Аргумент1,2 – локальная переменная, псевдоним, тег OPC, константа или другие выражения
Примечание: Аргументы должны быть числовыми значениями или строками, преобразованными в числовое значение (см. Преобразование типа данных)

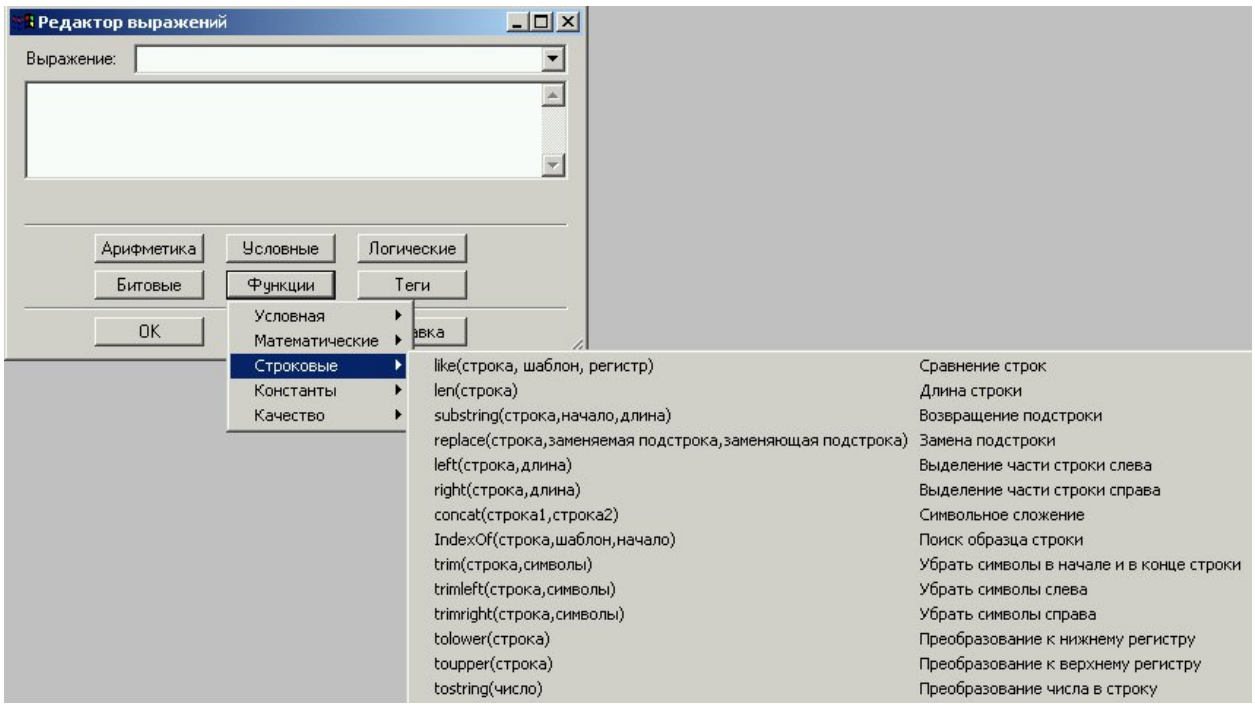
Необходимо соблюдать также допустимую область применения рассматриваемых функций. Например:
функция asin(аргумент) существует только когда –1<= аргумент <= 1.

Примеры:

Символ операции	Описание	Пример	Результат
Sin	Синус угла в радианах	Sin(~~var1~~)	Sin(0.785)=0.71
Cos	Косинус угла в радианах	Cos(~~var1~~)	Cos(0.785)=0.71
Tan	Тангенс угла в радианах	tan(~~var1~~)	tan(0.785)=1.00
Asin	Арксинус величины с возвратом результата в радианах	asin(~~var1~~)	asin(0.5)=0.52
Acos	Арккосинус величины с возвратом результата в радианах	acos(~~var1~~)	acos(0.5)=1.05
atan	Арктангенс величины с возвратом результата в радианах	atan(~~var1~~)	atan(1)=0.785
Sqrt	Квадратный корень	sqrt(~~var1~~)	Sqrt(100)=10
Pow	Возведение первого аргумента в степень, значение которой равно второму аргументу	pow(~~var1~~, ~~var2~~)	Pow(100,1.5)=1000
Log	Десятичный логарифм аргумента	log(~~var1~~)	Log(100)=2
Ln	Натуральный логарифм аргумента	ln(~~var1~~)	Ln(7.389)=2
Exp	Экспонента с показателем, значение которого равно аргументу	exp(~~var1~~)	Exp(2)=7.389
Abs	Абсолютное значение аргумента	abs(~~var1~~)	Abs(-1)=1
Ceil	Округление до большего целого	ceil(~~var1~~)	Ceil(7.39)=8
Floor	Округление до меньшего целого	floor(~~var1~~)	Floor(7.39)=7
min	Минимальное значение из двух аргументов	min(~~var1~~, ~~var2~~)	Min(10,5)=5
max	Максимальное значение из двух аргументов	max(~~var1~~, ~~var2~~)	Max(10,5)=10

Строковые функции

Строковые функции показаны на рисунке:



Следует отметить, что в строковых функциях, перечисленных ниже в таблице, позиция первого символа – нуль. Соответственно, если в функция возвращает позицию, или позиция передаётся как параметр, номер позиции отсчитывается с нуля.

Строковая функция	**Описание**	**Пример**	**Результат**
Like (строка,шаблон,регистр)	Сравнение строк.	См. ниже	См.ниже
Len (строка)	Длина строки	x=len("Hello World")	11
Substring (строка,начало, длина)	Возвращение подстроки - строка символов, начинающаяся с заданной позиции (*начало*), заданной длины (*длина*)	x=substring("Hello World",0,5)	Hello
Replace (строка,заменяемая подстрока,заменяющая подстрока)	Замена одной подстроки (*заменяемая подстрока*) другой (*заменяющая подстрока*)	x=replace("Status busy","busy","ready")	Status ready
Left (строка,длина)	Выделение части строки заданной длины слева	x=left("Hello World",5)	Hello
Right (строка,длина)	Выделение части строки заданной длины справа	x=right("Hello World",5)	World
Concat (строка1,строка2)	Символьное сложение строки1 и строки2	x=concat("Hello","World")	Hello World
IndexOf (строка,шаблон,*начало*)	Поиск образца строки. Эта функция возвращает целое число - позицию найденного шаблона строки, или возвращает значение 1, если образец в строке не найден. Начало поиска – позиция *начало*.	x=indexof("Hello World","World",0)	6
Trim (строка,символы)	Убрать указанные символы в начале и в конце строки.	x=trim("Hello World","Hdl")	ello Wor
trimleft(строка,символы)	Убрать символы в начале строки	x=trimleft("Hello World","Hdl")	ello World
trimright(строка,символы)	Убрать символы в конце строки	x=trimright("Hello World","Hdl")	Hello Wor

tolower(строка)	Преобразование строки к нижнему регистру	x=tolower("Hello World","Hdl")	hello world
toupper(строка)	Преобразование строки к верхнему регистру	x= toupper ("Hello World","Hdl")	HELLO WORLD
tostring(число)	Преобразование числа в строку	См. ниже	См. ниже

Примечание:
Когда строковая величина, которая может быть переведена в числовое значение, в строковой функции обозначена как число, будет произведено автоматическое преобразование. Например: x=Left("Hello World","4") –правильное выражение.
Если автоматическое преобразование невозможно, результат выражения будет неверным.
Для более подробной информации см. Преобразование типа данных

Примечание:
Возможно также преобразовывать числовое значение в строку, например:
x=len(35) вернёт значение 2.

Примечание:
Имейте в виду, что строковая функция вернёт плохое качество, если заданы некорректные параметры (например, отрицательная длина или отрицательная стартовая позиция).

Функция '**like**'.
Используется для сравнения строк. Производит поиск в строке заданного шаблона и возвращает значение в зависимости от того, содержится ли шаблон в строке.

Выражение: **like**(строка, шаблон, регистр)
где
строка – строка, состав которой анализируется на наличие строки, задаваемой шаблоном. Строка может быть переменной типа String, тегом OPC (типа String), массивом строк (тег OPC типа массив строк), строковой константой, числовым значением. Пример строковой константы: “This is a string”.
Шаблон – строка, тег OPC (типа String) или псевдоним, поиск которого выполняется в анализируемой строке. Могут быть использованы следующие символы ввода шаблона поиска: “*”, “?”, “#”, [*список*], [[!*список*]. Символ “*” применяется вместо группы символов в анализируемой строке; символ “?” используется вместо одного символа; символ “#” – вместо цифры (0-9), [*список*] - вместо любого символа из списка, [!*список*] - вместо любого символа не из списка.
Примечание: Символы ([), (?), (#), (*) могут быть использованы как символы левой скобки, знака вопроса , номера и звездочки соответственно, если они заключены в скобки. Символ правой скобки (]) не может быть использован в группе шаблона как символ правой скобки, но может использоваться вне группы как отдельный символ. Можно использовать символ дефис (-) для разделения верхнего и нижнего уровня диапазона символов в *списке*, например: диапазон поиска [A-Z] включает в себя все символы верхнего регистра от «A» до «Z». Составные диапазоны используются внутри скобок без ограничителей.
Регистр – значение «1» инициирует поиск совпадений в анализируемой строке с учетом регистра; нулевое значение указывает на то, что регистр не будет учитываться при поиске совпадений.
Результат - При обнаружении совпадений возвращается 1 (истина), и 0 (ложь) – в противном случае.
Перечислим несколько важных правил, которые нужно соблюдать в форме шаблона:

- Восклицательный знак (!) в начале *списка* означает, что совпадение строк обнаружено, если любой символ, кроме входящих в *список*, найден в строке. Если этот символ используется вне скобок, он воспринимается как символ восклицательного знака.
- Дефис (-) может использоваться как в начале (например, после восклицательного знака) или в конце *списка* (обозначая символ дефис). В любом другом положении, этот символ обозначает диапазон символов.
- Когда определяется диапазон символов, порядок их должен быть по-возрастающей (снизу вверх). Например, диапазон [Z-A] задан неверно. Верное обозначение-[A-Z].

Символ последовательности [] рассматривается как пустая строка ("").
Вы также можете использовать символ * (звёздочка) для замены любой группы символов в строке.
Важной особенностью является то, что функция like осуществляет посимвольное сравнение. Если строка сравнения имеет отличную от шаблона длину, результат функции будет «ложь». Единственное исключение этого правила составляют случаи использования * (звёздочки), которая может заменять любую группу символов.

Рассмотрим примеры использования функции “like”:
Like(~~var_user_name~~,"mark",1)
Результат выражения будет «истина», если строковая переменная ~~var_user_name~~ содержит строку "mark". Если же переменная ~~var_user_name~~ содержит строку "mark twain", результат выражения примет значение «ложь», т.к. две строки не совпадают полностью.

Функция	Результат	Комментарии
like("mark","mark",1)	1	Две строки совпадают, поэтому результат «истина»
like("mark","Mark",1)	0	Сравнение задано с учётом регистра, поэтому результат =«ложь»
like("mark","Mark",0)	1	Сравнение задано без учёта регистра, поэтому результат =«истина»
like("mark","ma*",1)	1	Символ «*»автоматически подставляет недостающие символы, поэтому результат = «истина».
like("mark","ma??",1)	1	Символ «?» заменяет любой один символ в строке, результат = «истина».
like("mark","ma?",1)	0	В этом случае, символов «?» недостаточно, чтобы совпала длина строки. Символьный результат выражения ="mar", поэтому результат сравнения =«ложь»
like("mark","m?rk",1)	1	Символ «?» может автоматически заменять символ в любом месте строки. Строка "m?rk" принимается как "mark", поэтому результат сравнения =«истина»
like("mark","m[abc]rk",0)	1	В этом примере в квадратные скобки заключен список символов, результат = «истина»
like("mark","ma[def]k",0)	0	Этот пример похожий на предыдущий, но здесь символ r не входит в список [def], результат выражения =«ложь»
like("mark","[a-z]ark",0)	1	В этом примере вместо первого символа принимается любой - между "a" и "z"
like("mark","[a-z]a*",0)	1	Пример похож на предыдущий, только после первых двух символов принимается любая символьная группа.
like("mark","[a-z]a",0)	0	Результат этого выражения «ложь», т.к. совпадают только первые два символа, длина же шаблона и

		анализируемой строки не совпадают.
like("mark","m[!bc]rk",0)	1	Результат выражения будет «истина», т.к. используется массив символов с восклицательным знаком, а второй символ не входит в указанный массив.
like("mark2","mark#",0)	1	Символ # заменяет любое число, поэтому результат выражения «истина».

Оператор '**tostring**' используется следующим образом:
Функция tostring преобразует значение выражения в скобках в строку.

Функция также может использоваться в выражениях соединения строк:

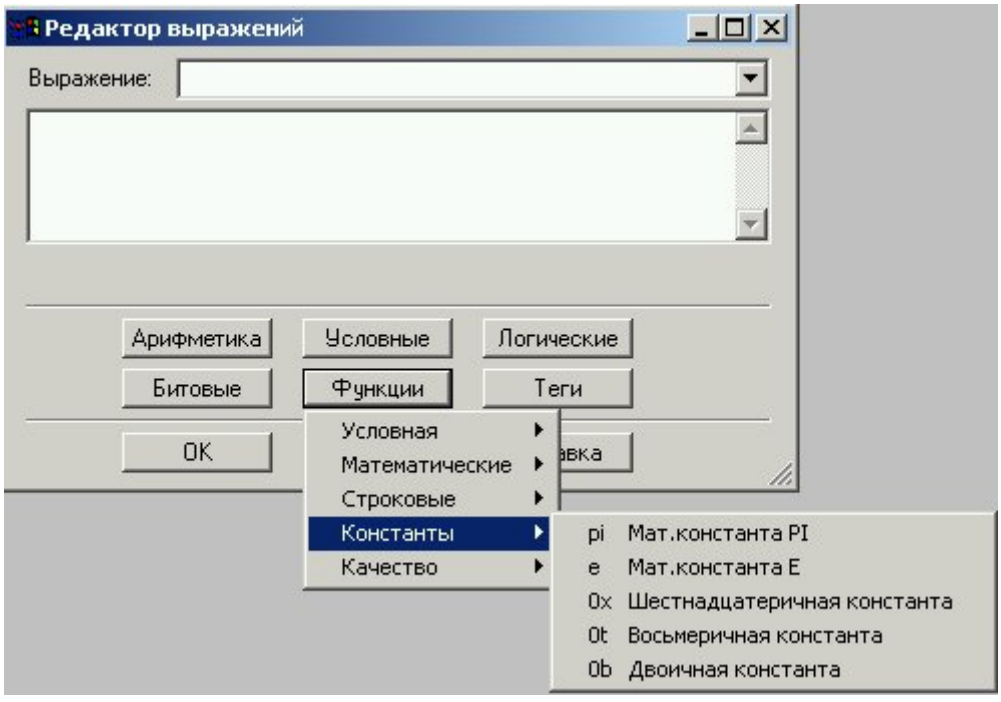
x="Значение = " + tostring(value) + " единицы"

Примечание: в выражении выше слово «единицы» представляет собой текст, заменяемый строкой, обозначающей единицы измерения величины (например, Ватты, метры ит.д.).
Пример.

Выражение	Результат
x="The value is " + tostring({{gfwsim.ramp.float}}) + " Watt"	“The value is 543.2345152 Watt”

Константы.

В **Редакторе выражений** поддерживаются константы, показанные на рисунке:



Примеры:

Символ	Описание	Пример	Результат
pi	Математическая константа pi	x=pi	3.14159265358979323846
e	Математическая константа e	x=e	2.7182818284590452354
0x	Шестнадцатеричная	x=0x11	17

	константа		
0t	Восьмеричная константа	x=0t11	9
0b	Двоичная константа	x=0b11	3

Примечание: Числовые константы "pi" и "e" являются приближенными. Округленные значения этих констант показаны в вышерассмотренной таблице примеров. Использование округленных значений математических констант "pi" и "e" могут привести к неожиданным результатам. Например, использование округленной константы pi в выражении tan(PI*N/2), даст неверный результат.

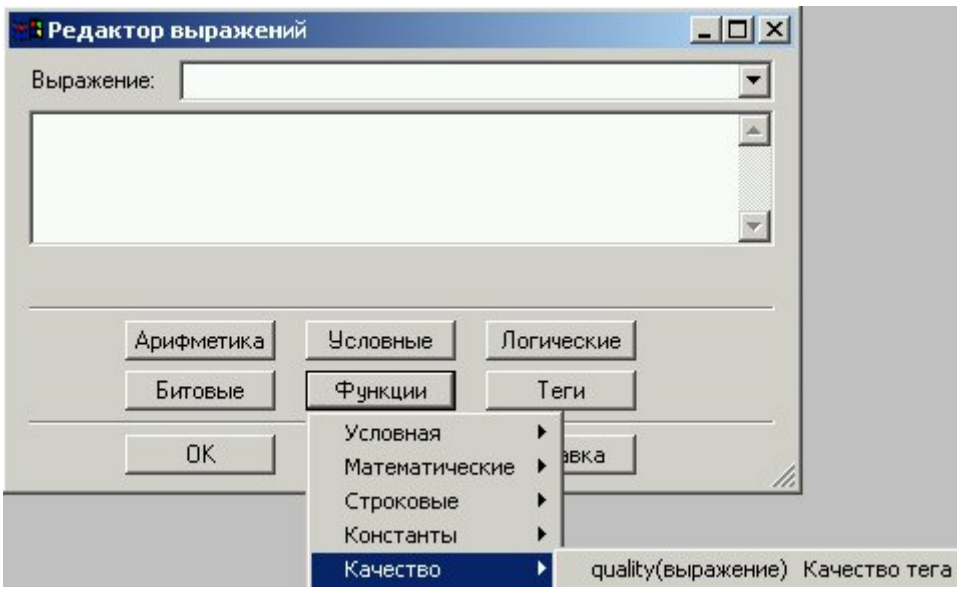
Примеры выражений с использованием констант.
Шестнадцатеричные: 0x20A = 2 * (16^2) + 0 * (16^1) + 10 * (16^0)= 2*256 + 0*16 + 10 * 1

= 512 + 0 + 10 = 522

Восьмеричные: 0t36= 3 * (7^1) + 6 *(7^0) = 3* 7 + 6* 1= 21 + 6 = 27

Двоичные: 0b110 = 1 * (2^2) + 1 * (2^1) + 0 * (2^ 0) = 1 * 4 + 1 * 2 + 0 * 1 = 4+2+0 = 6

Качество.



Функция **Качество** из раздела **Функции Редактора выражений** используется для определения качества тега или выражения.
Синтаксис:
x=quality(выражение)
Примечание: «выражение» может быть простым, состоящим из одного тега.
Результат операции может быть следующим:
192: quality is GOOD
64: quality UNCERTAIN
0: quality BAD
Примечание: по правилам OPC Foundation установлены следующие диапазоны значений для качества:

GOOD: 192-252
UNCERTAIN: 64-191
BAD: 0-63
Дополнительная информация содержится в описании стандарта OPC DA, на сайте международной организации OPC Foundation www.opcfoundation.org/.
Пример выражения:

Выражение	Результат
x=quality({{ICONICS.Simulator.1\SimulatePLC.PumpStatus}})	192 (Quality GOOD)

Качество выражения зависит от каждого тега в выражении. Таким образом, если в выражении присутствует несколько тегов, имеющих различное качество (192 [GOOD], 64 [BAD], 0 [UNCERTAIN]), результата выражения будет иметь качество, соответствующее низшему из уровней качества тегов выражения. Если выражение содержит условные операторы (if, then, or else), на результат выражения будет влиять только качество тегов, находящихся в исполняемой ветви.

Пример.
Если качество Tag1 равно GOOD (т.е. 192), результатом выражения будет Tag1. В противном случае (т.е. если качество UNCERTAIN или BAD), результатом выражения будет значение Tag2.
Результат и качество выражения в зависимости от различных качеств Tag1 и Tag2 приведены в таблице.

Пункт	Качество Tag1	Качество Tag2	Результат	Качество рез.
1	GOOD	GOOD	Tag1	192 (GOOD)
2	GOOD	UNCERTAIN	Tag1	192 (GOOD)
3	GOOD	BAD	Tag1	192 (GOOD)
4	UNCERTAIN	GOOD	Tag2	192 (GOOD)
5	UNCERTAIN	UNCERTAIN	Tag2	64 (UNCERTAIN)
6	UNCERTAIN	BAD	Tag2	0 (BAD)
7	BAD	GOOD	Tag2	192 (GOOD)
8	BAD	UNCERTAIN	Tag2	64 (UNCERTAIN)
9	BAD	BAD	Tag2	0 (BAD)

Примечание. Функция “quality()” возвращает значение, представляющее собой качество выражения в скобках, и всегда имеющее качество GOOD. Например, если Tag1 имеет качество BAD, то выражение “x=quality({{Tag1}})", будет иметь значение 0 с качеством GOOD.

Пример.
Рассмотрим следующее выражение.
x= if ({{TAG_01}}>0,{{TAG_02}},{{TAG_03}})
Выражение будет исполняться следующим образом:
«Если значение TAG_01 больше 0, присвоить результату значение TAG_02. Если значение TAG_01 меньше или равно 0, присвоить результату значение TAG_03."

Предположим, теги имеют следующие значения и качества:

TAG_01=5 качество GOOD

TAG_02=6 качество UNCERTAIN

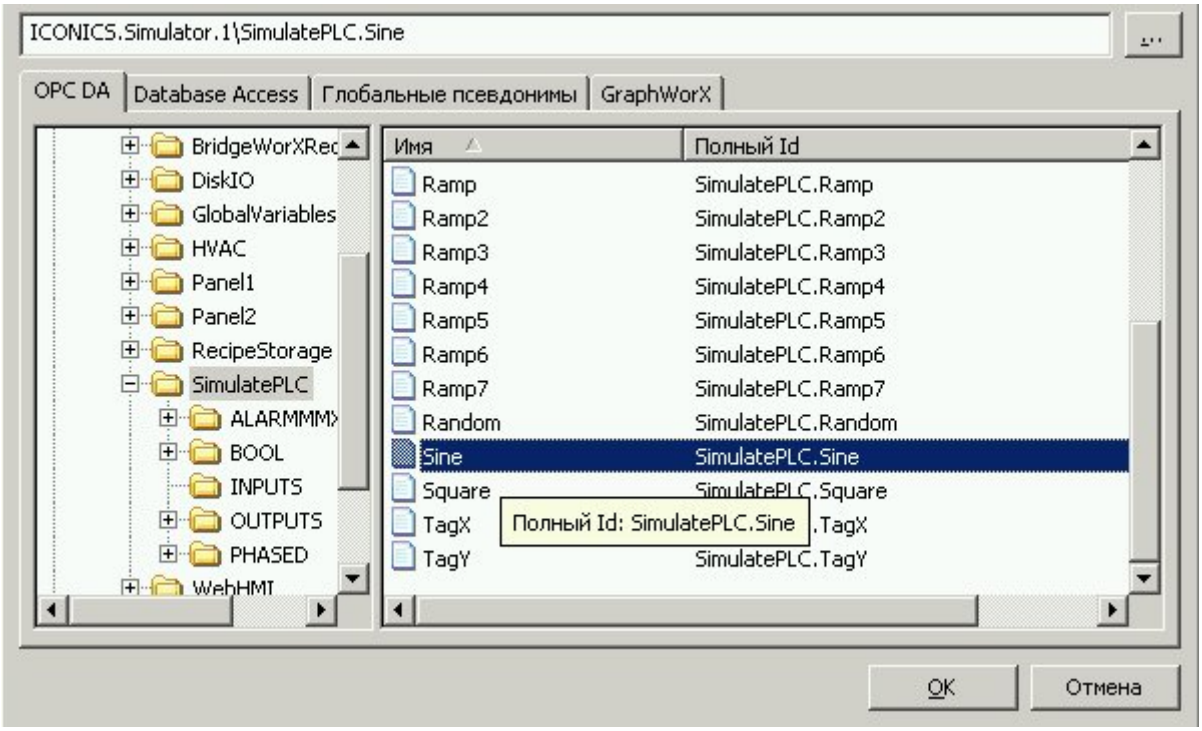
TAG_03=7 качество BAD

Т.к. значение TAG_01=5 (больше нуля), результатом будет значение TAG_02. Таким образом, результат выражения будет равен 6, качество – UNCERTAIN.

Глава 2 Теги.

Теги OPC

Меню тегов доступно в разделе **Теги Редактора выражений**.
OPC-тег – *текстовая строка* (описатель), уникальным образом идентифицирующая структуру данных, которая связывает клиента с OPC-сервером.
OPC-тег может быть использован в выражении при использовании следующего синтаксиса:
{{имя_тега}}
Например: x={{ICONICS.Simulator.1\SimulatePLC.PumpSpeed}}
Вы можете использовать Универсальный навигатор данных, показанный на рисунке, чтобы выбрать OPC AE (тревоги и события), OPC DA (текущие данные), OPC HDA (исторические данные) для включения их в выражениях.



Выбор OPC-тегов из Универсального навигатора данных

Локальные и глобальные псевдонимы

Псевдоним – это строка, представляющая или описывающая объект или тег экранной формы. В выражениях могут быть использованы глобальные и локальные псевдонимы.

Локальные псевдонимы

Для использования локальных псевдонимов в качестве операндов выражений применяется следующий синтаксис:

<<имя_локального_псевдонима>>

Например:

X= <<TankLevel>>

Глобальные псевдонимы

Для использования глобальных псевдонимов в качестве операндов выражений применяется следующий синтаксис:

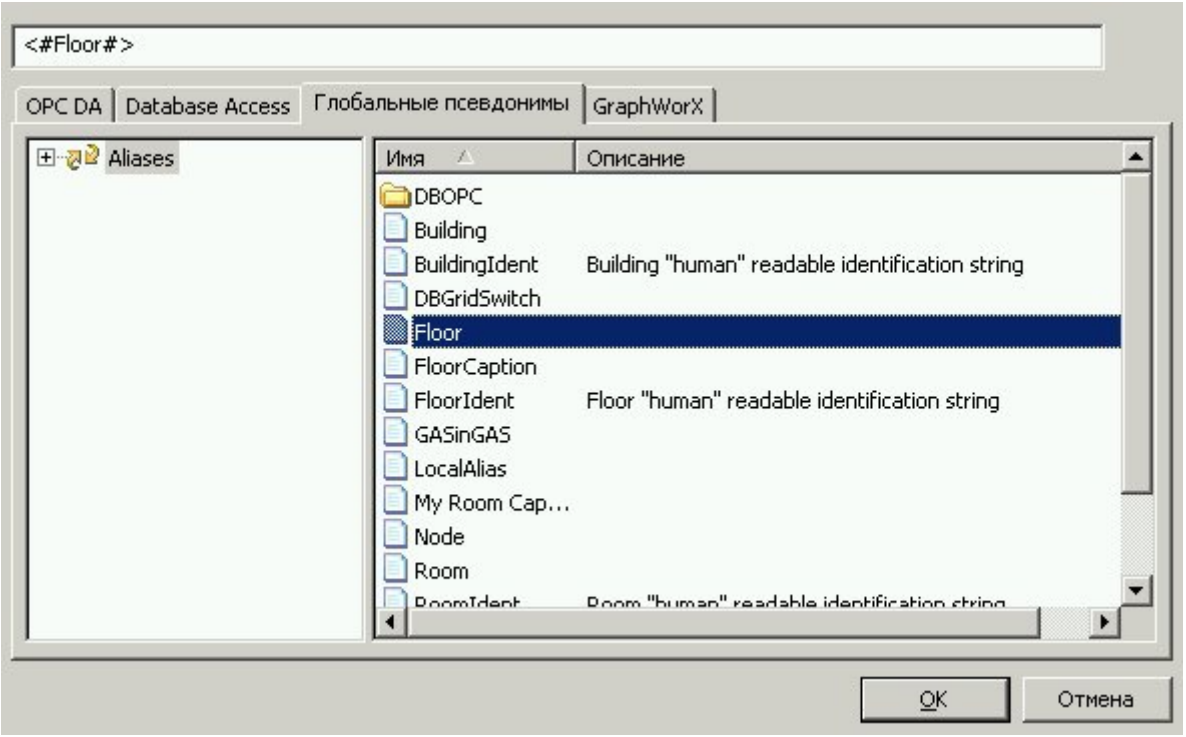
<#имя_глобального_псевдонима#>

Например:

X=<#RoomTemperature#>

Для выбора глобальных псевдонимов Вы можете использовать Универсальный навигатор данных. Это освобождает от необходимости вводить псевдонимы вручную. Все глобальные псевдонимы, прописанные в Конфигураторе глобальных псевдонимов,

доступны в навигаторе. Выбор псевдонима осуществляется двойным щелчком мыши, после чего выбранный псевдоним появляется в верхней строке и дополняется разделителями. Выбрав псевдоним, нажмите кнопку ОК.



Выбор глобального псевдонима из Универсального навигатора данных.

Переменные

Для использования переменных в выражениях, необходимо использовать следующий синтаксис:

Для локальной переменной

~~имя_локальной_переменной~~

Например:

X=~~Setpoint~~

Для переменной моделирования

{{ имя_переменной_для_моделирования}}

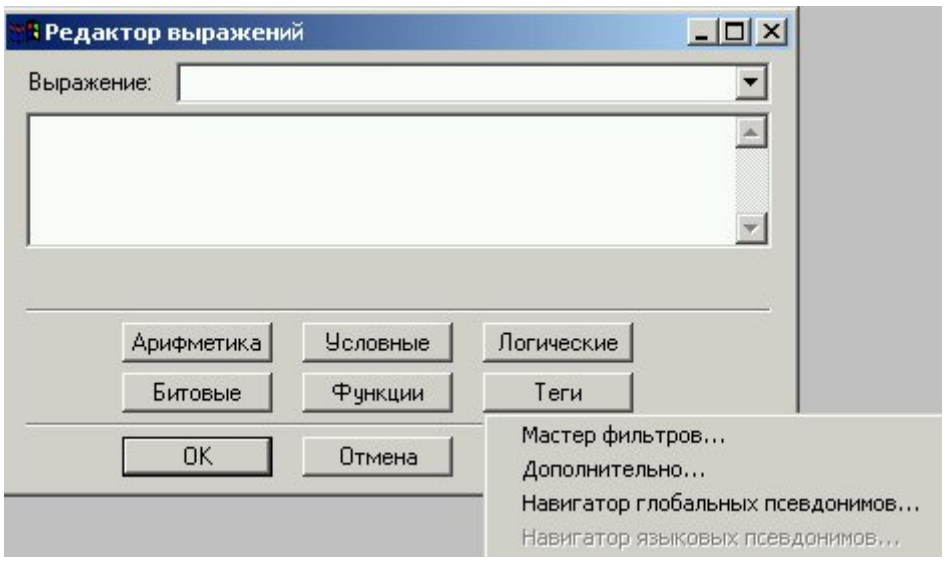
Например:

x={{gfwsim.random.long}}

Фильтры тревог

Редактор выражений, показанный на рисунке, может быть использован для изменения клиентских фильтров тревог (для более подробной информации, обратитесь к справочной документации по AlarmWorX32).

Для простого и удобного создания фильтров тревог, из **Редактора выражений** есть возможность вызова Мастера создания фильтров, а также списка тревог. Используя функции Редактора выражений, есть возможность установить фильтры тревог вручную.



Изменение фильтров тревог с использованием Редактора выражений.

Мастер фильтров

Мастер фильтров, показанный на рисунке, позволяет использовать в выражениях указанные параметры фильтра. Выберите один или более параметров, затем нажмите ОК. Строка с этими параметрами фильтра автоматически вставится в диалоговое окно Редактора выражений.

Типы событий: Событие, Подтвержденное событие, Неподтвержденное событие, Норма, Взаимодействие, Оператор

Подсостояния: LoLo, Lo, Hi, HiHi, Скорость изменения и Дискретное состояние

Важность: величина приоритета тревоги. Уровень важности лежит в диапазоне от 1 (нижний предел) до 1000 (верхний).

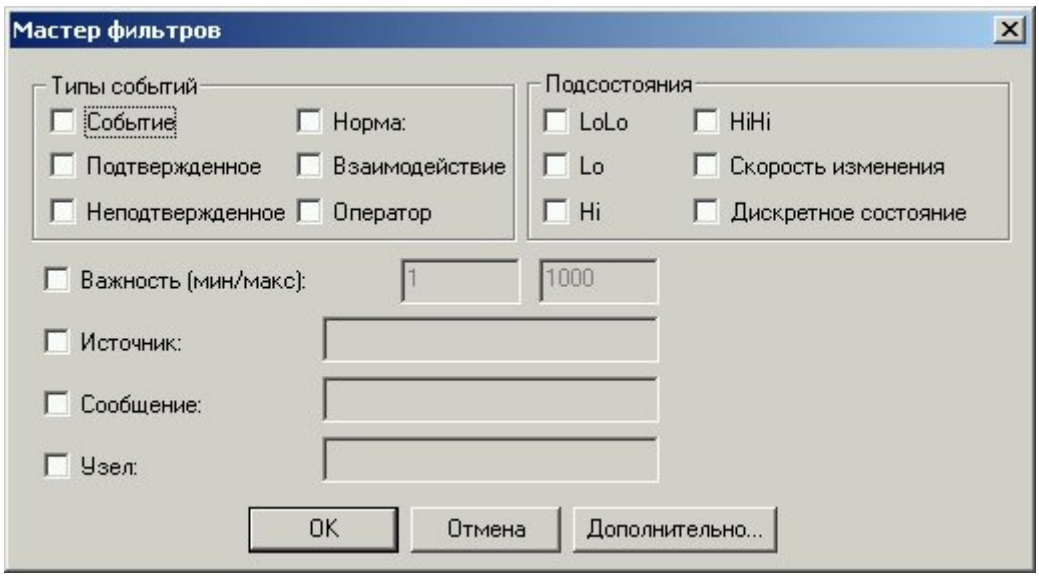
Источник: имя тега

Сообщение: Описание тега

Узел: Имя компьютера, с которого создается тег

Для открытия **Редактора выражений**, нужно нажать на кнопку **Дополнительно**....

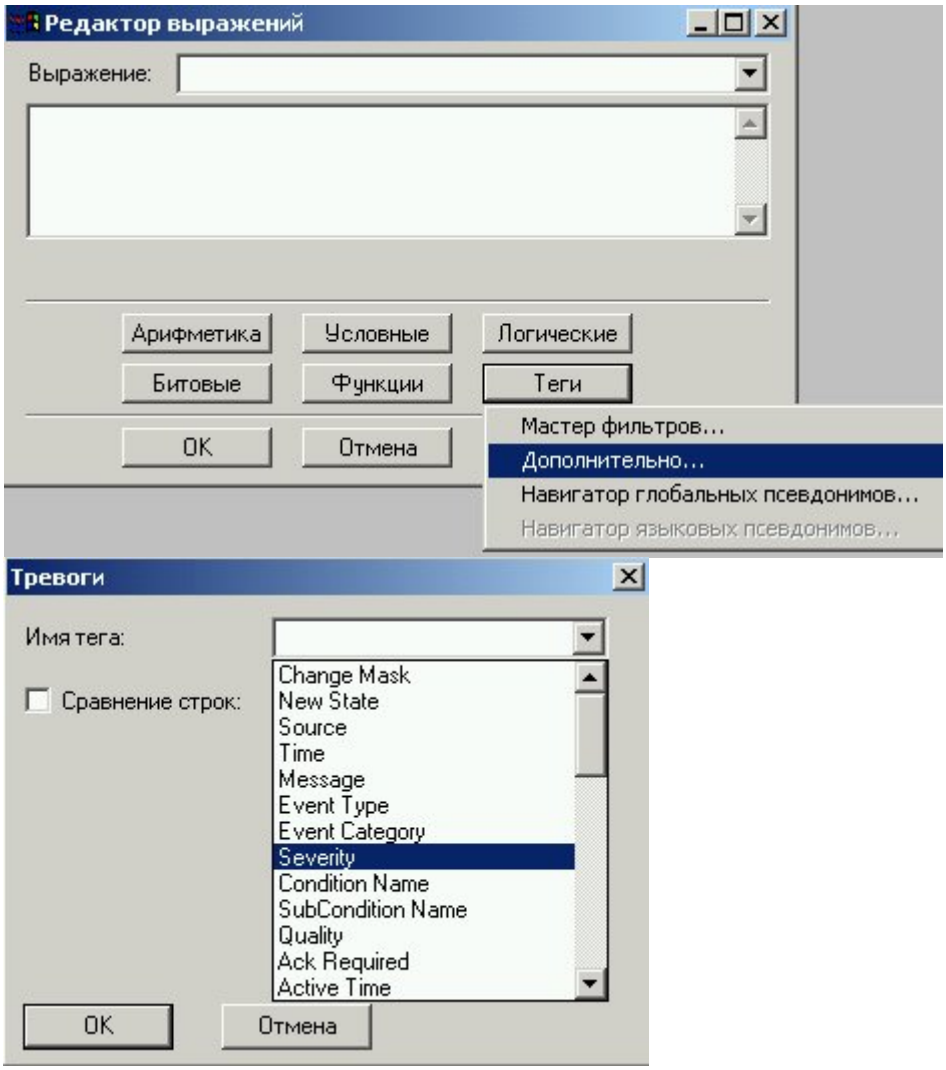
Примечание: Объединение нескольких **Типов событий** между собой в строке выражения осуществляется через логическое «или». Выбор нескольких **Подсостояний** также равнозначно логическому «или». Объединение Типов событий, Подсостояний, Важности, Источника, Сообщения и Узла осуществляется с помощью логического «и».



Мастер фильтров

Выбор параметров тревог

Выбрав пункт **Дополнительно...** в меню **Теги** Редактора выражений**,** открывается диалоговое окно **Тревоги,** позволяющее выбрать параметры фильтра тревог.



Список параметров фильтра тревог

Из выпадающего списка параметров фильтров (поле **Имя тега**) можно выбрать необходимый для включения в выражение параметр, затем нажать ОК.

Список возможных параметров:

Change Mask	Маска изменения
New State	Новое состояние
Source	Источник события
Time	Время события
Message	Сообщение
Event Type	Тип события
Event Category	Категория события
Severity	Важность
Condition Name	Имя состояния
SubCondition Name	Имя подсостояния
Quality	Качество
Ack Required	Требование подтверждения
Active Time	Время активности тревоги
Cookie	Cookie
Number Event Attributes	Количество атрибутов события
Actor ID	Идентификатор субъекта
Attribute 1-20	Атрибут 1-20
Alarm Type	Тип тревоги
Current Time	Текущее время
Server Description	Описание сервера
Server Node	Узел сервера
Server ProgID	ProgID сервера
Subscription	Подписка

Примечание: Диалоговое окно **Тревоги** позволяет выполнять сравнение строк с использованием оператора «like». Для этого нужно отметить галочкой поле **Сравнение строк** и указать в текстовом поле строку поиска.

Пример фильтров тревог:

Выражение	**Результат**
X = {{Severity}} > 500.	Видны будут тревоги, важность которых больше, чем 500
X = Like({{Source}}, "Tag",0)	Будут показаны сообщения от источников, в имени которых используется “Tag”
X = Like({{Message}}, "Boiler Pump not",0)	Результат выражения – сообщения, описание которых начинается с "Boiler Pump not"
X = Like({{Node}}, "Alpha",0)	Пропускаются только сообщения с узла Alpha
X = Like({{Attribute1}}, "Plant Area 1",0) X = {{Attribute2}} == 40.	Понятия *Атрибут 1-20* определяются в подписке на сообщения сервера. Они могут быть строковыми, числовыми значениями или массивами. Наиболее часто используются массивы для **Областей**. Если вы работаете с **Областями**, очень удобно использовать функцию Like
X = 1.	Фильтр показывает все сообщения

X = 0.	Фильтр не пропускает ни одного сообщения

Все фильтры принимают значение «истина» или «ложь». Любое ненулевое значение воспринимается как «истина».

Для более подробной информации, обратитесь к справочной документации по AlarmWorX32).