Модуль подсистемы “Сбор данных” <JavaLikeCalc>

Модуль:	JavaLikeCalc
Имя:	Вычислитель на Java-подобном языке.
Тип:	DAQ
Источник:	daq_JavaLikeCalc.so
Версия:	1.3.1
Автор:	Роман Савоченко
Описание:	Предоставляет, основанные на java подобном языке, вычислитель и движок библиотек. Пользователь может создавать и модифицировать функции и их библиотеки.
Лицензия:	GPL

Оглавление

Введение

Модуль контроллера *JavaLikeCalc* предоставляет в систему механизм создания функций и их библиотек на Java-подобном языке. Описание функции на Java-подобном языке сводится к обвязке параметров функции алгоритмом. Кроме этого, модуль наделен функциями непосредственных вычислений, путём создания вычислительных контроллеров.

Непосредственные вычисления обеспечиваются созданием контроллера и связыванием его с функцией этого же модуля. Для связанной функции создаётся кадр значений, над которым и выполняются периодические вычисления.

Параметры функции могут свободно создаваться, удаляться или модифицироваться. Текущая версия модуля поддерживает до 255 параметров функции в сумме с внутренними переменными. Вид редактора функций показан на рис.1.

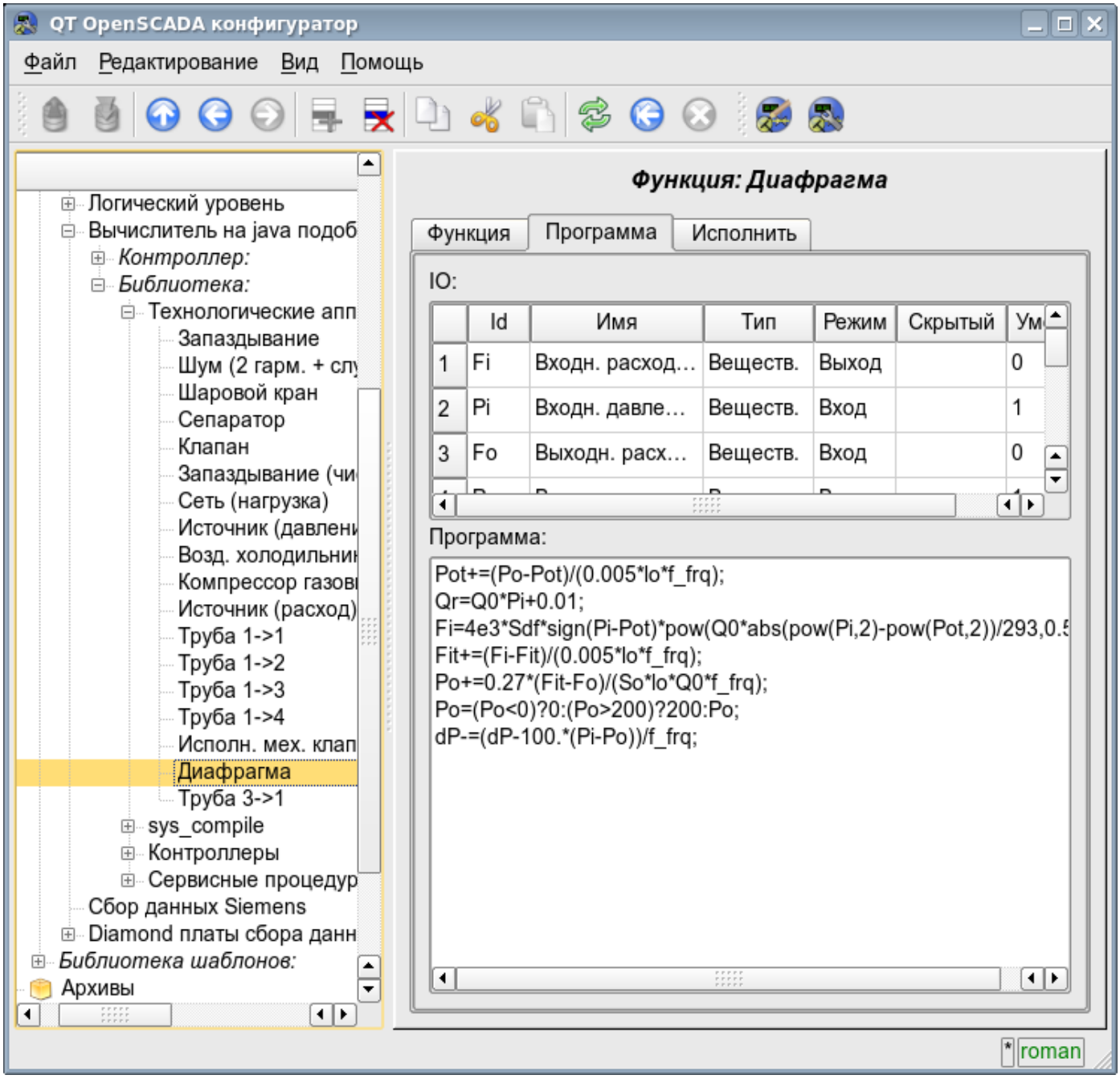

Рис.1. Вид редактора функций.

После любого изменения программы, или конфигурации параметров, выполняется перекомпиляция программы, с упреждением связанных с функцией объектов значений TValCfg. Компилятор языка построен с использованием известного генератора грамматики «Bison», который совместим с не менее известной утилитой Yacc.

Язык использует неявное определение локальных переменных, которое заключается в определении новой переменной в случае присваивания ей значения. Причём тип локальной переменной устанавливается в соответствии с типом присваиваемого значения. Например, выражение <Qr=Q0*Pi+0.01;> определит переменную Qr с типом переменной Q0.

В работе с различными типами данных, язык использует механизм автоматического приведения типов в местах, где подобное приведение является целесообразным.

Для комментирования участков кода, в языке предусмотрены символы «//» и «/* ... */». Всё, что идёт после "//", до конца строки, и между «/* ... */» игнорируется компилятором.

В процессе генерации кода компилятор языка производит оптимизацию по константам и приведение типов констант к требуемому типу. Под оптимизацией констант подразумевается выполнение вычислений в процессе построения кода над двумя константами и вставка результата в код. Например, выражение <y=pi*10;> свернётся в простое присваивание <y=31.4159;>. Под приведением типов констант к требуемому типу, подразумевается формирования в коде константы, которая исключает приведение типа в процессе исполнения. Например, выражение <y=x*10>, в случае вещественного типа переменной x, преобразуется в <y=x*10.0>.

Язык поддерживает вызовы внешних и внутренних функций. Имя любой функции, вообще, воспринимается как символ, проверка на принадлежность которого, к той или иной категории, производится в следующем порядке:

- ключевые слова;
- константы;
- встроенные функции;
- внешние функции;
- уже зарегистрированные символы переменных;
- новые атрибуты системных параметров;
- новые параметры функции;
- новая автоматическая переменная.

Вызов внешней функции, как и атрибута системного параметра, записывается как адрес к объекту динамического дерева объектной модели системы OpenSCADA в виде: <DAQ.JavaLikeCalc.lib_techApp.klapNotLin>.

Для предоставления возможности написания пользовательских процедур управления различными компонентами OpenSCADA, модулем предоставляется реализация API прекомпиляции пользовательских процедур отдельных компонентов OpenSCADA на реализации Java-подобного языка. Такими компонентами уже являются: Шаблоны параметров подсистемы «Сбор данных» и Среда визуализации и управления (СВУ).

1. Java-подобный язык

1.1. Элементы языка

Ключевые слова: if, else, while, for, break, continue, return, using, true, false.

Постоянные:

- десятичные: цифры 0–9 (12, 111, 678);
- восьмеричные: цифры 0–7 (012, 011, 076);
- шестнадцатеричные: цифры 0–9, буквы a-f или A-F (0x12, 0XAB);
- вещественные: 345.23, 2.1e5, 3.4E-5, 3e6;
- логические: true, false;
- строковые: «hello».

Типы переменных:

- целое: $-2^{31}...2^{31}$;
- вещественное: $3.4 * 10^{308}$;
- логическое: false, true;
- строка: длина 256 символов и без перехода на другую строку.

Встроенные константы: pi = 3.14159265, e = 2.71828182, EVAL_BOOL(2), EVAL_INT(-2147483647), EVAL_REAL(-3.3E308), EVAL_STR("<EVAL>")

Атрибуты параметров системы OpenSCADA (начиная с подсистемы DAQ, в виде <Тип модуля DAQ>.<Контроллер>.<Параметр>.<Атрибут>).

Функции объектной модели системы OpenSCADA.

1.2. Операции языка

Операции поддерживаемые языком представлены в таблице ниже. Приоритет операций уменьшается с верху вниз. Операции с одинаковым приоритетом входят в одну цветовую группу.

Символ	Описание
()	Вызов функции.
{}	Программные блоки.
++	Инкремент (пост и пре).
--	Декремент (пост и пре).
-	Унарный минус.
!	Логическое отрицание.
~	Побитовое отрицание.
*	Умножение.
/	Деление.
%	Остаток от целочисленного деления.
+	Сложение
-	Вычитание
<<	Поразрядный сдвиг влево
>>	Поразрядный сдвиг вправо
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Неравно
|	Поразрядное «ИЛИ»
&	Поразрядное «И»
^	Поразрядное «Исключающее ИЛИ»
&&	Логический «И»
||	Логический «ИЛИ»
?:	Условная операция (i=(i<0)?0:i;)
=	Присваивание.
+=	Присваивание с сложением.
-=	Присваивание с вычитанием.
*=	Присваивание с умножением.
/=	Присваивание с делением.

1.3. Встроенные функции языка

Для обеспечения высокой скорости работы в математических вычислениях, модуль предоставляет встроенные математические функции, которые вызываются на уровне команд виртуальной машины. Встроенные математические функции:

- sin(x) — синус x;
- cos(x) — косинус x;
- tan(x) — тангенс x;
- sinh(x) — синус гиперболический от x;
- cosh(x) — косинус гиперболический от x;
- tanh(x) — тангенс гиперболический от x;
- asin(x) — арксинус от x;
- acos(x) — арккосинус от x;
- atan(x) — арктангенс от x;
- rand(x) — случайное число от 0 до x;
- lg(x) — десятичный логарифм от x;
- ln(x) — натуральный логарифм от x;
- exp(x) — экспонента от x;
- pow(x,x1) — возведение x в степень x1;
- max(x,x1) — максимальное значение из x и x1;
- min(x,x1) — минимальное значение из x и x1;
- sqrt(x) — корень квадратный от x;
- abs(x) — абсолютное значение от x;
- sign(x) — знак числа x;
- ceil(x) — округление числа x до большего целого;
- floor(x) — округление числа x до меньшего целого.

1.4. Операторы языка

Общий перечень операторов языка:

- *if* — оператор условия «Если»;
- *else* — оператор условия «Иначе»;
- *while* — описание цикла while;
- *for* — описание цикла for;
- *break* — прерывание выполнение цикла;
- *continue* — продолжить выполнение цикла с начала;
- *using* — позволяет установить область видимости функций часто используемой библиотеки (using Special.FLibSYS;) для последующего обращения только по имени функции;
- *return* — прерывание функции и возврат результата, результат копируется в атрибут с флагом возврата (return 123;).

1.4.1. Условные операторы

Языком модуля поддерживаются два типа условий. Первый это операции условия для использования внутри выражения, второй – глобальный, основанный на условных операторах.

Условие внутри выражения строится на операциях «?» и «:». В качестве примера можно записать следующее практическое выражение <st_open=(pos>=100)?true:false;>, что читается как «Если переменная <*pos*> больше или равна 100, то переменной st_open присваивается значение true иначе false.

Глобальное условие строится на основе условных операторов «if» и «else». В качестве примера можно привести тоже выражение, но записанное другим способом <if(pos>100) st_open=true; else st_open=false;>. Как видно, выражение записано по другому но читается также.

1.4.2. Циклы

Поддерживаются два типа циклов: while и for. Синтаксис циклов соответсвует языкам программирования: C++, Java и JavaScript.

Цикл **while**, в общем, записывается следующим образом: *while(<условие>) <тело цикла>;* Цикл **for** записывается следующим образом: *for(<пре-инициализ>;<условие>;<пост-вычисление>) <тело цикла>;* Где:

<*условие*> — выражение определяющее условие;

<*тело цикла*> — тело цикла множественного исполнения;

<*пре-инициализ*> — выражение предварительной инициализации переменных цикла;

<*пост-вычисление*> — выражение модификации параметров цикла, после очередной итерации.

1.4.3. Специальные символы строковых переменных

Языком предусмотрена поддержка следующих специальных символов строковых переменных:

'\n' — перевод строки;

'\t' — символ табуляции;

'\b' — забой;

'\f' — перевод страницы;

'\r' — возврат каретки;

'\\' — сам символ '\'.

1.5. Примеры программы на языке

Приведём несколько примеров программ на Java-подобном языке:

```
//Модель хода исполнительного механизма шарового крана
if( !(st_close && !com) && !(st_open && com) )
{
      tmp_up=(pos>0&&pos<100)?0:(tmp_up>0&&lst_com==com)?tmp_up-1./frq:t_up;
      pos+=(tmp_up>0)?0:(100.*(com?1.:-1.))/(t_full*frq);
      pos=(pos>100)?100:(pos<0)?0:pos;
      st_open=(pos>=100)?true:false;
      st_close=(pos<=0)?true:false;
      lst_com=com;
}
//Модель клапана
Qr=Q0+Q0*Kpr*(Pi-1)+0.01;
Sr=(S_kl1*l_kl1+S_kl2*l_kl2)/100.;
Ftmp=(Pi>2.*Po)?Pi*pow(Q0*0.75/Ti,0.5):(Po>2.*Pi)?
      Po*pow(Q0*0.75/To,0.5):pow(abs(Q0*(pow(Pi,2)-pow(Po,2))/Ti),0.5);
Fi-=(Fi-7260.*Sr*sign(Pi-Po)*Ftmp)/(0.01*lo*frq);
Po+=0.27*(Fi-Fo)/(So*lo*Q0*frq);
Po=(Po<0)?0:(Po>100)?100:Po;
To+=(abs(Fi)*(Ti*pow(Po/Pi,0.02)-To)+(Fwind+1)*(Twind-To)/Riz)/(Ct*So*lo*Qr*frq);
```

2. Контроллер и его конфигурация

Контроллер этого модуля связывается с функциями из библиотек построенных с его помощью для обеспечения непосредственных вычислений. Для предоставления вычисленных данных в систему OpenSCADA, в контроллере могут создаваться параметры. Пример вкладки конфигурации контроллера данного типа изображен на рис.2.

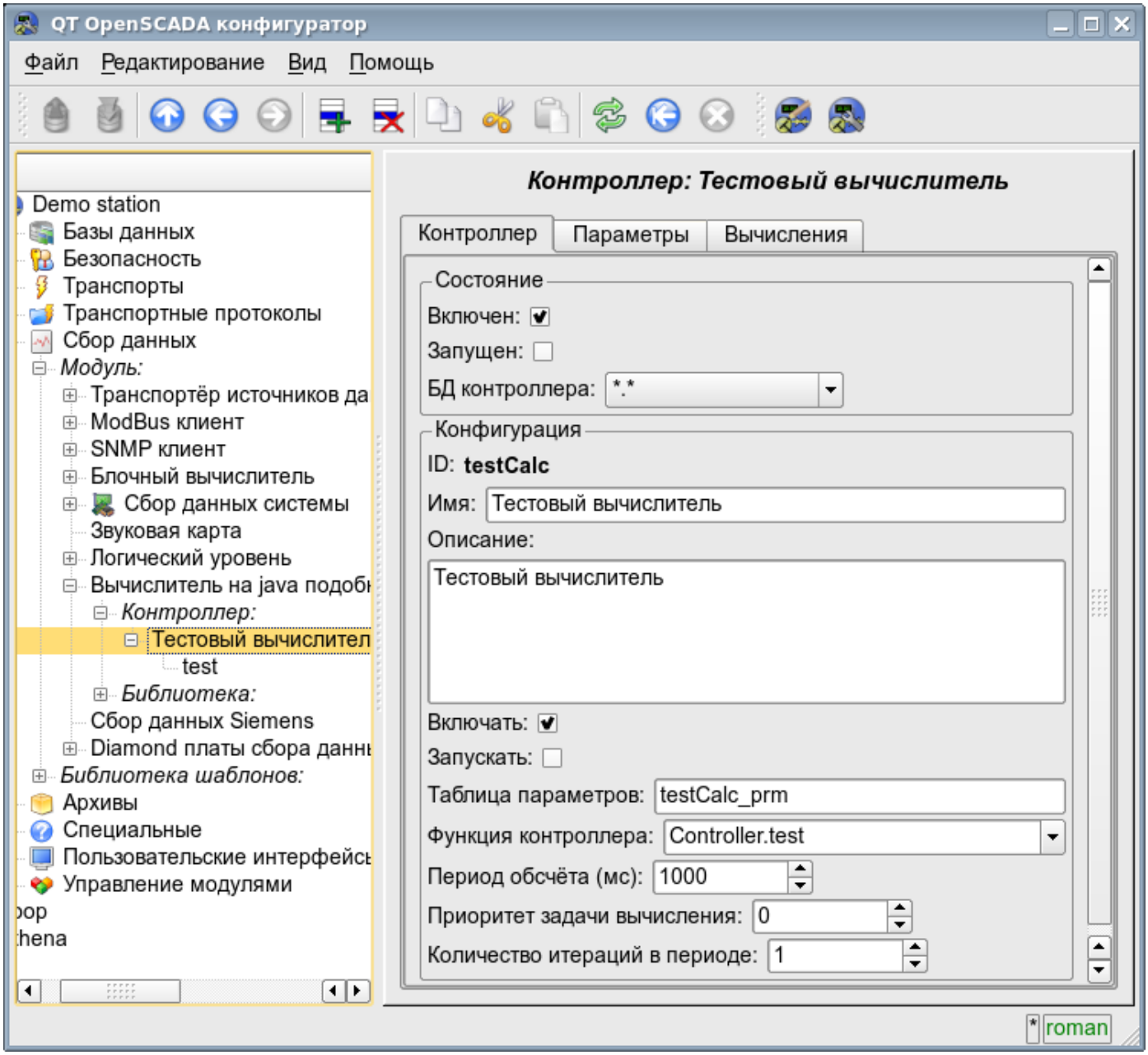

Рис.2. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: «Включен», «Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Имя таблицы для хранения параметров.
- Адрес вычислительной функции.
- Период, приоритет и число итераций в одном цикле задачи вычисления.
- Период автоматической синхронизации блоков с БД.

- Сохранить/загрузить контроллер в БД.

Вкладка «Вычисления», контроллера (Рис. 3), содержит параметры и текст программы, непосредственно выполняемой контроллером. Также, для контроля выполнения, выводится время вычисления программы.

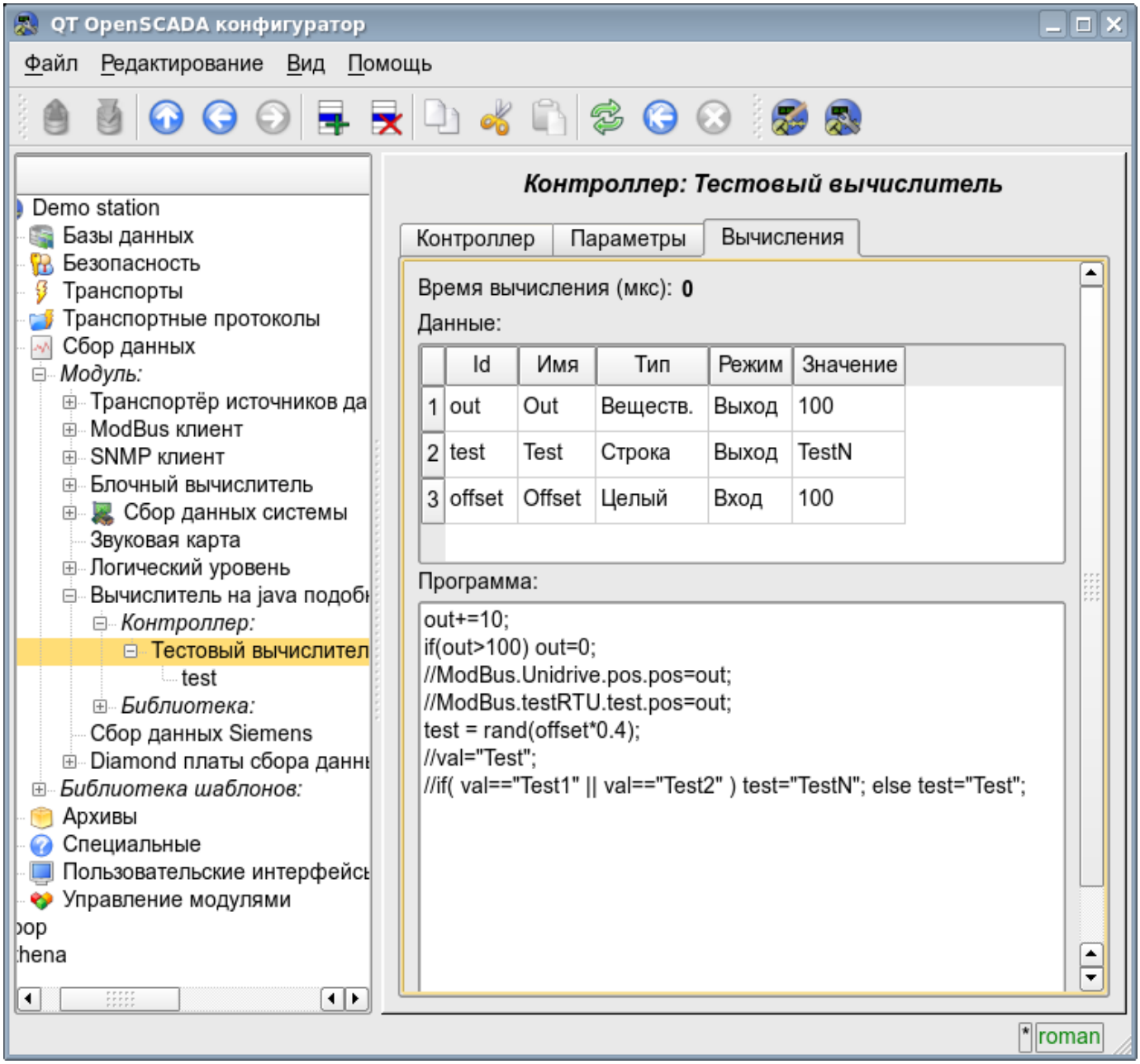

Рис.3. Вкладка «Вычисления» контроллера.

3. Параметр контроллера и его конфигурация

Параметр контроллера данного модуля выполняет функцию предоставления доступа к результатам вычисления контроллера в систему OpenSCADA, посредством атрибутов параметров. Из специфических полей, вкладка конфигурации параметра контроллера содержит только поле перечисления параметров вычисляемой функции, которые необходимо отразить.

4. Библиотеки функций модуля

Модуль предоставляет механизм для создания библиотек пользовательских функций на Java-подобном языке. Пример вкладки конфигурации библиотеки изображен на Рис.4. Вкладка содержит базовые поля: состояния, идентификатор, имя и описание, а также адрес таблицы хранящей библиотеку. Во вкладке «Функции» библиотеки, кроме перечня функций, содержится форма копирования функций.

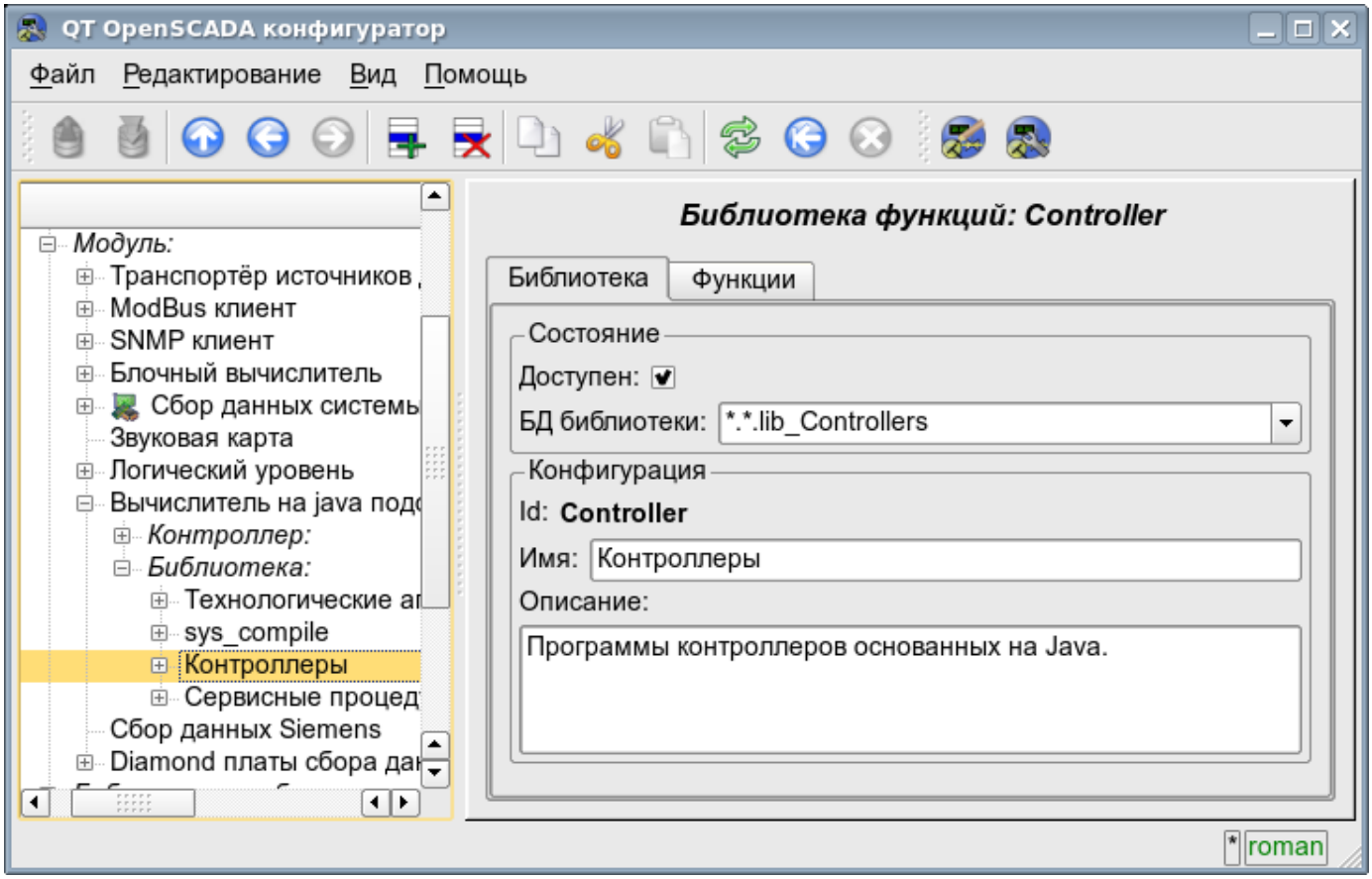

Рис.4. Вкладка конфигурации библиотеки.

5. Пользовательские функции модуля

Функция, также как и библиотека, содержит базовую вкладку конфигурации, вкладку формирования программы и параметров функции (Рис.1), а также вкладку исполнения созданной функции.