Модуль подсистемы “Протоколы” <UserProtocol>

Модуль:	UserProtocol
Имя:	Пользовательский протокол
Тип:	Протокол
Источник:	prot_UserProtocol.so
Версия:	0.6.2
Автор:	Роман Савоченко
Описание:	Позволяет создавать собственные пользовательские протоколы на любом OpenSCADA языке.
Лицензия:	GPL

Оглавление

Введение

Модуль транспортного протокола UserProtocol предназначен для предоставление пользователю возможности создания реализаций различных протоколов собственными силами на одном из внутренних языков OpenSCADA, обычно JavaLikeCalc, и не прибегая к низкоуровневому программированию OpenSCADA.

Основная цель модуля - упростить задачу подключения к системе OpenSCADA устройств источников данных, которые имеют незначительное распространение и/или предоставляют доступ к собственным данным по специфическому протоколу, обычно достаточно простому для реализации на внутреннем языке OpenSCADA. Для реализации этого предоставляется механизм формирования протокола исходящего запроса.

Кроме механизма протокола исходящего запроса предоставляется механизм протокола входящего запроса, который позволяет OpenSCADA обслуживать запросы на получение данных по специфическим протоколам, которые достаточно просто могут быть реализованы на внутреннем языке OpenSCADA.

Модуль предоставляет возможность создания реализаций множества различных протоколов в объекте "Пользовательский протокол" (рис.1).

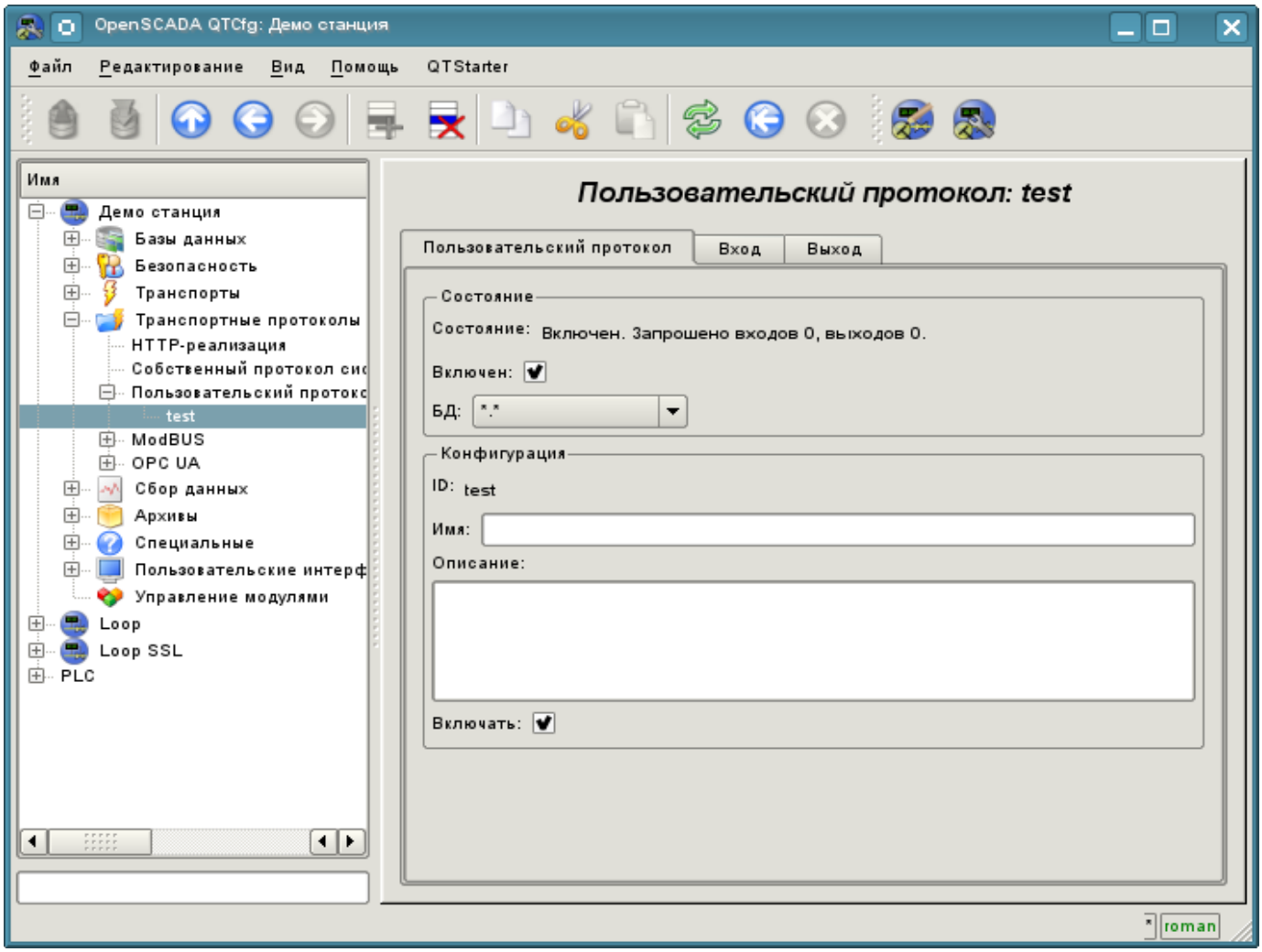

Рис.1. Основная вкладка объекта "Пользовательский протокол".

Главная вкладка содержит основные настройки пользовательского протокола:

- Раздел "Состояние" - содержит свойства, характеризующие состояние протокола:
  - *Состояние* - текущее состояние протокола.
  - *Включен* - состояние протокола "Включен".
  - *БД* - БД в которой хранится конфигурация.

- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе протокола.
  - *Имя* - указывает имя протокола.
  - *Описание* - краткое описание протокола и его назначения.
  - *Включать* - указывает на состояние "Включен", в которое переводить протокол при загрузке.

1. Часть протокола для входящих запросов

Протокол входящих запросов работает в кооперации с входящим транспортом и отдельный объект "Пользовательского протокола" указывается в поле конфигурации протокола транспорта вместе с именем модуля UserProtocol. В дальнейшем все запросы к транспорту будут направляться в процедуру обработки запроса протокола (рис.2).

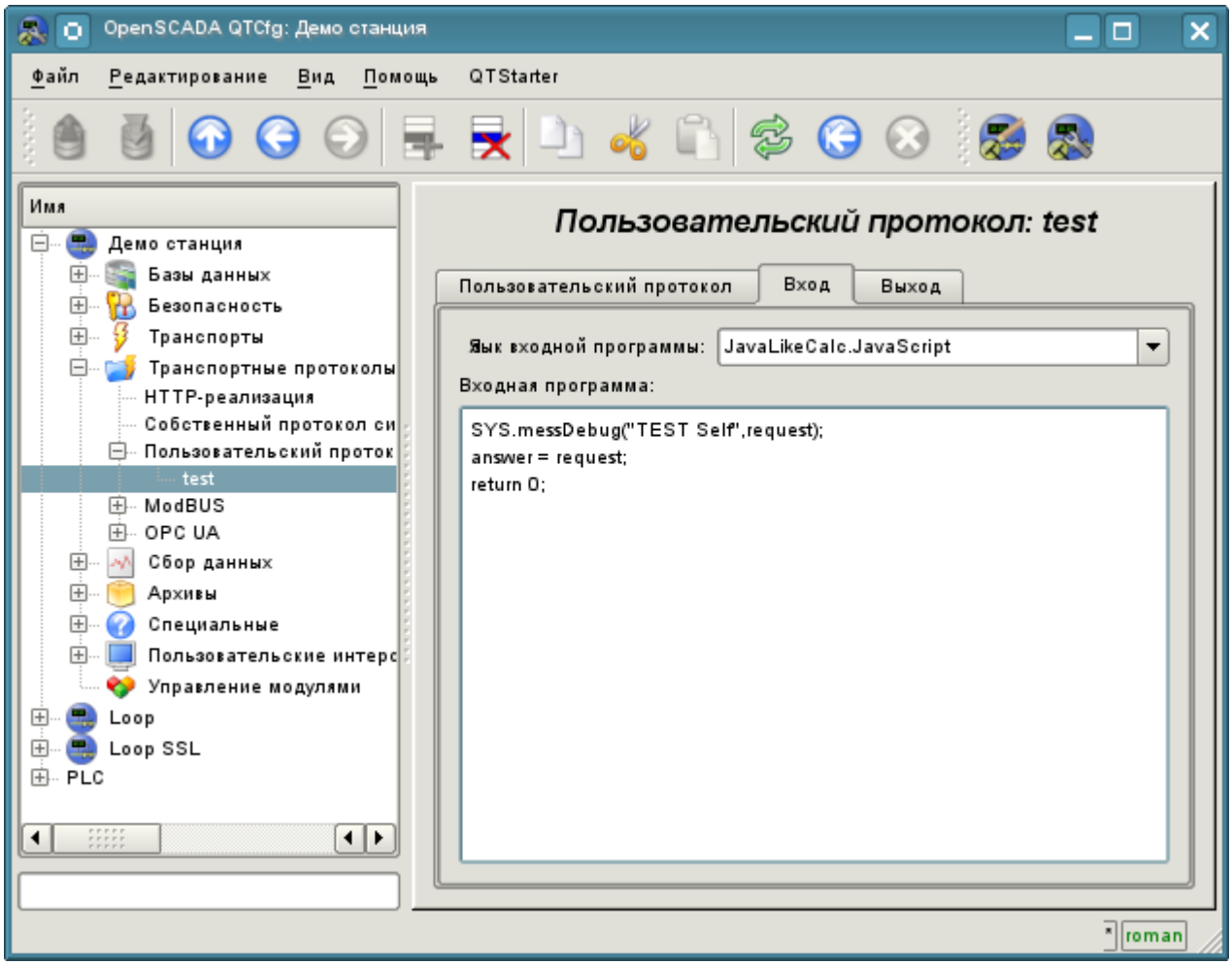

Рис.2. Вкладка процедуры обработки входящих запросов.

Вкладка процедуры обработки входящих запросов содержит поле для выбора внутреннего языка программирования OpenSCADA и поле ввода текста процедуры обработки.

Для процедуры обработки предопределены следующие переменные обмена с входящим транспортом:

- *rez* - результат обработки (false-полный запрос;true-не полный запрос);
- *request* - сообщение запроса;
- *answer* - сообщение ответа;
- *sender* - отправитель запроса.

Общий сценарий обработки входящих запросов:

- Запрос формируется удалённой станцией и через сеть попадает на транспорт OpenSCADA.

- OpenSCADA транспорт передаёт запрос, выбранному в поле протокола, модулю UserProtocol и объекту пользовательского протокола в виде значений переменной "request" - для блока запроса и "sender" - для адреса отправителя запроса.
- Запускается выполнение процедуры протокола входящего запроса, в процессе которой анализируется содержимое переменной "request" и формируется ответ в переменной "answer". По окончанию выполнения процедуры формируется переменная "rez", которая указывает транспорту на факт получения полноценного запроса и формирование корректного ответа (false) или необходимость транспорту ожидать оставшихся данных (true).
- Если в результате процедуры обработки переменная "rez" равна 'false' и ответ в переменной "answer" не нулевой, то транспорт отправляет ответ и обнуляет накопление "request".
- Если в результате процедуры обработки переменная "rez" равна 'true' то транспорт продолжает ожидать данные. При получении следующей порции данных они добавляются к переменной "request" и выполнение процедуры повторяется.

В качестве примера рассмотрим реализацию обработки запросов по протоколу DCON, для некоторых запросов к источнику данных с адресом "10":

```
//SYS.messDebug("TEST REQ: ",request);
//Проверка запроса на полноту
if(request.length < 4 || request[request.length-1] != "\r")
{
      if(request.length > 10) request = "";
      return true;
}
//Проверка запроса на целостность (CRC) и адрес
CRC = 0;
for(i = 0; i < (request.length-3); i++) CRC += request.charCodeAt(i);
if(CRC != request.slice(request.length-3,request.length-1).toInt(16) ||
request.slice(1,3).toInt(16) != 10) return false;
//Анализ запроса и подготовка ответа
if(request.charCodeAt(0) == "#") answer = ">+05.123+04.153+07.234-02.356+10.000-
05.133+02.345+08.234";
else if(request.charCodeAt(0) == "@") answer = ">AB3C";
else answer = "?";
//Завершение ответа
CRC = 0;
for(i=0; i < answer.length; i++) CRC += answer.charCodeAt(i);
answer += (CRC&0xFF).toString(16)+"\r";
//SYS.messDebug("TEST ANSV: "+answer.charCodeAt(0),answer);
return 0;
```

2. Часть протокола для исходящих запросов

Протокол исходящих запросов работает в кооперации с исходящим транспортом и отдельным объектом "Пользовательского протокола". Источником запроса через протокол может выступать функция общесистемного API пользовательского программирования исходящего транспорта *int messIO(XMLNodeObj req, string prt);*, в параметрах которой указывается:

- *req* - запрос в виде дерева XML со структурой соответствующей входному формату реализованного протокола;
- *prt* - имя модуля "UserProtocol".

Запрос отправленный вышеуказанным образом направляться в процедуру обработки запроса протокола (рис.3) с идентификатором пользовательского протокола указываемым в атрибуте req.attr("ProtIt").

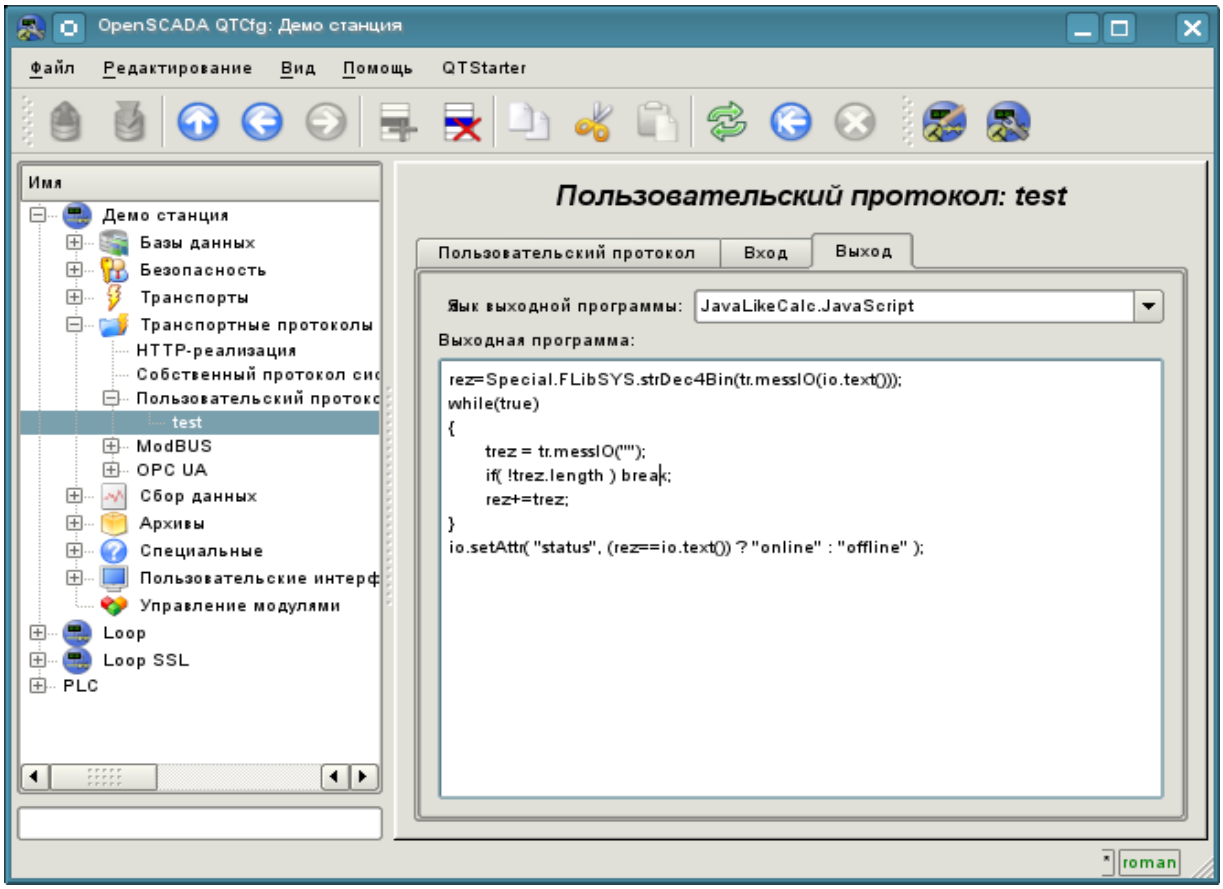

Рис.3. Вкладка процедуры обслуживания исходящих запросов.

Вкладка процедуры обработки исходящих запросов содержит поле для выбора внутреннего языка программирования OpenSCADA и поле ввода текста процедуры обработки.

Для процедуры обработки предопределены следующие переменные обмена:

- *io* - XML узел обмена с клиентом, через который протокол принимает запросы и в который помещает результат с форматом реализуемым в процедуре;
- *tr* - объект транспорта, предназначен для вызова функции транспорта *string messIO(string mess, real timeOut = 1000);* "tr.messIO(req)".

Общий сценарий формирования исходящего запроса:

- Формирование XML-дерева в соответствии со структурой реализуемой протоколом и указание идентификатора пользовательского протокола в атрибуте "ProtIt".
- Отправка запроса к транспорту через протокол *SYS.Transport["Modul"]["OutTransp"].messIO(req,"UserProtocol");*.
- Выбор пользовательского интерфейса в соответствии с req.attr("ProtIt") и инициализация переменных исходящего транспорта io - соответственно к первому аргументу messIO() и tr -

объект "OutTransp".

- Вызов процедуры на исполнение, которая обработав структуру "io" формирует прямой запрос транспорту *tr.messIO(req);*, результат которого обрабатывается и помещается назад в io.

Суть выделения протокольной части кода в процедуру пользовательского протокола заключается в упрощении интерфейса клиентского обмена при многократном использовании и предполагает формирование структуры XML-узла обмена в виде атрибутов адресов удалённых станций, адресов читаемых и записываемых переменных, а также значений самих переменных. При этом весь груз непосредственного кодирования запроса и декодирования ответа возлагается на процедуру пользовательского протокола.

В качестве примера рассмотрим реализацию запросов посредством протокола DCON, к обработчику, реализованному в предыдущем разделе. Начнём с реализации протокольной части:

```
//Формирование конечного запроса
request = io.name().slice(0,1)+io.attr("addr").toInt().toString(16,2)+io.text();
CRC = 0;
for(i=0; i < request.length; i++) CRC += request.charCodeAt(i);
request += (CRC&0xFF).toString(16)+"\r";
//Отправка запроса
resp = tr.messIO(request);
while(resp[resp.length-1] != "\r")
{
      tresp = tr.messIO("");
      if(!tresp.length) break;
      resp += tresp;
}
//Анализ ответа
if(resp.length < 4 || resp[resp.length-1] != "\r") { io.setAttr("err","10:Ошибка
или нет ответа."); return; }
//Проверка ответа на целостность (CRC)
CRC = 0;
for(i = 0; i < (resp.length-3); i++) CRC += resp.charCodeAt(i);
if(CRC != resp.slice(resp.length-3,resp.length-1).toInt(16))
{ io.setAttr("err","11:Ошибка CRC."); return; }
if(resp[0] != ">") { io.setAttr("err","12:"+resp[0]+":Ошибка DCON."); return; }
//Возврат результата
io.setAttr("err","");
io.setText(resp.slice(1,resp.length-3));
```

И процедура непосредственной отправки DCON запроса, через предыдущую процедуру протокола. Эту процедуру необходимо поместить в нужную задачу или промежуточную функцию OpenSCADA, например в процедуру контроллера DAQ.JavaLikeCalc: //Подготовка запроса

```
req = SYS.XMLNode("#").setAttr("ProtIt","DCON").setAttr("addr",10);
//Отправка запроса
SYS.Transport["Serial"]["out_TestDCON"].messIO(req,"UserProtocol");
if(!req.attr("err").length) SYS.messDebug("TEST REQ","RES: "+req.text());
//Подготовка второго запроса
req = SYS.XMLNode("@").setAttr("ProtIt","DCON").setAttr("addr",10);
//Отправка второго запроса
SYS.Transport["Serial"]["out_TestDCON"].messIO(req,"UserProtocol");
if(!req.attr("err").length) SYS.messDebug("TEST REQ","RES: "+req.text());
```