OpenSCADA v. 0.7.0

(http://oscada.org)

12 Октябрь, 2010

Оглавление

Введение....10
Предпосылки....10
Цели проекта....11
Политика разработки. Лицензия.....11
Области применения....11
Архитектура....11
Функциональные характеристики и требования системы OpenSCADA....13
1. Сфера применения системы OpenSCADA....13
1.1. Сервер SCADA системы:....14
1.2. Станция оператора технологического процесса, пульт диспетчера, панель мониторинга и др.:....15
1.3. Среда исполнения контроллеров (PLC):....16
2. Требования OpenSCADA....18
2.1. Исполнение....18
2.2. Сборка....19
Описание программы OpenSCADA....21
1. Функции системы.....22
1.1. Модульность.....22
1.2. Подсистемы.....23
1.3. PLC и другие источники динамических данных. Подсистема "Сбор данных".....23
1.4. Базы данных. Подсистема "Базы данных"....24
1.5. Архивы. Подсистема "Архивы".....24
1.6. Коммуникации. Подсистемы "Транспорты" и "Транспортные протоколы".....26
1.7. Интерфейсы пользователя. Подсистема "Интерфейсы пользователя".....27
1.8. Безопасность системы. Подсистема "Безопасность".....27
1.9. Управление библиотеками модулей и модулями. Подсистема "Управление модулями".....27
1.10. Непредусмотренные возможности. Подсистема "Специальные".....27
1.11. Пользовательские функции. Объектная модель и среда программирования системы.....28
2. SCADA системы и их структура.....29
3. Варианты конфигурирования и использования.....31
3.1. Простое серверное подключение.....31
3.2. Дублированное серверное подключение.....32
3.3. Дублированное серверное подключение на одном сервере.....32
3.4. Клиентский доступ посредством Web-интерфейса. Место руководителя.....33
3.5. Автоматизированное рабочее место (место руководителя/оператора).....33
3.6. АРМ с сервером сбора и архивирования на одной машине (место оператора, модель ...).....34
3.7. Простейшее смешанное подключение (модель, демонстрация, конфигуратор ...).....35
3.8. Устойчивая, распределённая конфигурация.....36
4. Конфигурация и настройка системы.....38
4.1. Подсистема "БД"....44
4.2. Подсистема "Безопасность"....50
4.3. Подсистема "Транспорты"....53
4.4. Подсистема "Транспортные протоколы"....59
4.5. Подсистема "Сбор данных"....60
4.6. Подсистема "Архивы"....72
4.7. Подсистема "Пользовательские интерфейсы"....83
4.8. Подсистема "Специальные"....84

4.9. Подсистема "Управление модулями"....85
4.10. Конфигурационный файл OpenSCADA и параметры командной строки вызова OpenSCADA....86
5. Общесистемное API пользовательского программирования....97
5.1. Общесистемные пользовательские объекты....97
Объект Array....97
Объект XMLNodeObj....98
5.2. Система (SYS)....98
5.3. Любой объект дерева OpenSCADA (SYS.*)....99
5.4. Подсистема "БД" (SYS.BD)....99
5.5. Подсистема "Сбор данных" (SYS.DAQ)....99
5.6. Подсистема "Архивы" (SYS.Archive)....99
5.7. Подсистема "Транспорты" (SYS.Transport)....100
Сбор данных в OpenSCADA....101
1. Методы сбора данных....103
1.1. Простой синхронный механизм сбора....103
1.2. Простой асинхронный механизм сбора....104
1.3. Пакетный механизм сбора....105
1.4. Пассивный механизм сбора....106
2. Виртуальные источники данных....108
3. Логический уровень обработки данных....110
4. Резервирование источников данных....115
Быстрый старт OpenSCADA....119
1. Термины, определения и аббревиатуры....119
2. Установка и запуск....120
2.1. Установка OpenSCADA из готовых пакетов....120
2.2. Установка из исходных текстов....122
3. Первичная конфигурация и запуск....122
4. Работа с источниками данных....127
4.1. Опрос данных аппарата ТП....127
4.2. Обработка полученных данных ТП....135
4.3. Включение архивирования данных ТП....144
5. Формирование визуального представления....147
5.1. Добавление шаблонной страницы в проект и подключение динамики....148
5.2. Создание нового кадра, мнемосхемы....152
5.3. Создание нового комплексного элемента....159
5.3.1. Создание виджета "Воздушный холодильник" на основе примитива "Элементарная фигура"....159
5.3.2. Создание финального скомпонованного виджета "Холодильник" на основе примитива "Группа элементов"....165
5.3.3. Добавление комплексного элемента на мнемосхему....184
6. Рецепты....186
Заключение....186
Модуль подсистемы “Архивы” <FSArch>....187
1. Архиватор сообщений....188
1.1. Формат файлов архива сообщений....190
1.2. Пример файла архива сообщения....191
2. Архиватор значений....192
2.1. Формат файлов архива значений....194
3. Эффективность....196
Модуль подсистемы “Архивы” <DBArch>....197
1. Архиватор сообщений....198
2. Архиватор значений....198
3. Информационная таблица архивных таблиц....199
Модуль подсистемы “БД” <DBF>....200

1. Операции над БД....200
2. Операции над таблицей....200
3. Операции над содержимым таблицы....200
4. Производительность БД....201
Модуль подсистемы “БД” <MySQL>....202
1. Операции над БД....202
2. Операции над таблицей....202
3. Операции над содержимым таблицы....202
4. Права доступа....203
5. Производительность БД....203
Модуль подсистемы “БД” <SQLite>....204
1. Операции над БД....204
2. Операции над таблицей....204
3. Операции над содержимым таблицы....204
4. Права доступа....205
5. Производительность БД....205
Модуль подсистемы “БД” <FireBird>....206
1. Операции над БД....206
2. Операции над таблицей....206
3. Операции над содержимым таблицы....206
4. Права доступа....207
5. Производительность БД....207
Модуль подсистемы “БД” <PostgreSQL>....208
1. Операции над БД....208
2. Операции над таблицей....208
3. Операции над содержимым таблицы....208
4. Права доступа....209
5. Производительность БД....209
Модуль подсистемы “Сбор данных” <DiamondBoards>....210
1. Контроллер данных плат фирмы Diamond....211
2. Параметры контроллера Diamond....213
Модуль подсистемы “Сбор данных” <System>....215
1. Контроллер данных....216
2. Параметры....217
Модуль подсистемы “Сбор данных” <BlockCalc>....219
1. Контроллер модуля....221
2. Блочная схема контроллера....222
3. Параметры контроллера....225
4. Копирование блочных схем....226
Модуль подсистемы “Сбор данных” <JavaLikeCalc>....227
1. Java-подобный язык....229
1.1. Элементы языка....229
1.2. Операции языка....230
1.3. Встроенные функции языка....231
1.4. Операторы языка....231
1.4.1. Условные операторы....231
1.4.2. Циклы....232
1.4.3. Специальные символы строковых переменных....232
1.5. Объект....232
1.6. Примеры программы на языке....234
2. Контроллер и его конфигурация....235
3. Параметр контроллера и его конфигурация....236
4. Библиотеки функций модуля....237
5. Пользовательские функции модуля....237
Модуль подсистемы “Сбор данных” <LogicLev>....238

1. Контроллер данных....239
2. Параметры....240
Модуль подсистемы “Сбор данных” <SNMP>....243
1. SNMP....243
1.1. MIB....243
1.2. Адресация....244
1.3. Взаимодействие....244
1.4. Авторизация....244
2. Модуль....245
2.1. Контроллер данных....245
2.2. Параметры....246
Модуль подсистемы “Сбор данных” <Siemens>....248
1. Коммуникационные контроллеры CIF....249
2. Контроллер источника данных....251
3. Параметры источника данных....252
4. Асинхронный режим записи....255
Модули <ModBus> подсистемы “Сбор данных” и подсистемы «Транспортные протоколы»
....256
1. Общее описание протокола ModBus....257
1.1. Адресация....257
1.2. Стандартные коды функций....257
2. Модуль реализации протокола....258
2.1. API функции исходящих запросов....258
2.2. Обслуживание запросов по протоколу ModBus....259
Режим узла протокола «Данные»....260
Режим узла протокола «Шлюз узла»....264
Режим узла протокола «Шлюз сети»....265
2.3 Отчёт запросов ModBus....266
3. Модуль сбора данных....267
3.1. Контроллер данных....267
3.2. Параметры....269
Модуль подсистемы “Сбор данных” <DCON>....271
1. Общее описание протокола DCON....271
2. Модуль....272
2.1. Контроллер данных....272
2.2. Параметры....273
3. Таблица совместимости модулей ввода-вывода различных производителей....274
Модуль подсистемы “Сбор данных” <ICP_DAS>....275
1. Контроллер данных....276
2. Параметры....277
2.1 Модуль I-8017....279
2.2 Модуль I-8042....280
2.3 Модуль I-87019....280
2.4 Модуль I-87024....281
2.5 Модуль I-87057....281
3. Настройка контроллеров серии LP-8x81....282
Ссылки....282
Модуль подсистемы “Сбор данных” <DAQGate>....283
1. Контроллер данных....284
2. Параметры....285
Модуль подсистемы “Сбор данных” <SoundCard>....286
1. Контроллер данных....287
2. Параметры....288
Модули <OPC_UA> подсистемы “Сбор данных” и подсистемы "Транспортные протоколы"
....290

1. Протокол OPC UA....291
2. Модуль реализации протокола....292
2.1. Обслуживание запросов по протоколу OPC UA....292
3. Модуль сбора данных....294
3.1. Контроллер данных....294
3.2. Параметры....296
4. Замечания....297
Модуль подсистемы “Транспорты” <Sockets>....298
1. Входящие транспорты....298
2. Исходящие транспорты....300
Модуль подсистемы “Транспорты” <SSL>....302
1. Входящие транспорты....303
2. Исходящие транспорты....304
3. Сертификаты и ключи....305
Модуль подсистемы “Транспорты” <Serial>....306
1. Входящие транспорты....307
2. Исходящие транспорты....309
Модуль подсистемы “Протоколы” <HTTP>....312
1. Аутентификация....313
2. Модули пользовательского WEB-интерфейса....314
3. API функции исходящих запросов....315
Модуль подсистемы “Протоколы” <SelfSystem>....317
1. Синтаксис протокола....317
2. Внутренняя структура исходящего протокола....318
Модуль подсистемы “Протоколы” <UserProtocol>....319
1. Часть протокола для входящих запросов....321
2. Часть протокола для исходящих запросов....322
Модуль подсистемы “Специальные” <FLibComplex1>....324
1. Сигнал (alarm) <111>....324
2. Условие '<' (cond_lt) <239>....324
3. Условие '>' (cond_gt) <240>....324
4. Полное условие (cond_full) <513>....325
5. Дискретный блок (digitBlock) <252>....325
6. Деление (div) <526>....325
7. Экспонента (exp) <476>....325
8. Расход (flow) <235>....325
9. Итератор (increment) <181>....325
10. Задержка (lag) <121>....326
11. Простое умножение(mult) <259>....326
12. Умножение+деление(multDiv) <468>....326
13. ПИД регулятор (pid) <745>....326
14. Степень (pow) <564>....327
15. Выбор (select) <156>....327
16. Простой сумматор (sum) <404>....327
17. Сумма с делением (sum_div) <518>....327
18. Сумма с умножением (sum_mult) <483>....327
Модуль подсистемы “Специальные” <FLibMath>....328
1. Функции....328
Модуль подсистемы “Специальные” <FLibSYS>....329
1. Общесистемные функции....329
1.1. Вызов консольных команд и утилит операционной системы (sysCall)....329
1.2. SQL запрос (dbReqSQL)....330
1.3. Узел XML (xmlNode)....330
1.4. Запрос интерфейса управления (xmlCntrReq)....330
1.5. Архив значений (vArh)....331

Объект VArchObj....331
1.6. Буфер архива значений (vArhBuf)....331
2. Функции для работы с астрономическим временем....332
2.1. Строка времени (tmFStr) <3047>....332
2.2. Полная дата (tmDate) <973>....332
2.3. Абсолютное время (tmTime) <220>....332
2.4. Конвертация времени из символьного представлении во время в секундах, от эпохи 1.1.1970 (tmStrPTime) <2600>....333
2.5. Планирование времени в формате Cron (tmCron)....333
3. Функции работы с сообщениями....333
3.1. Запрос сообщений (messGet)....333
3.2. Генерация сообщения (messPut)....334
4. Функции работы с строками....334
4.1. Получение размера строки (strSize) <114>....334
4.2. Получение части строки (strSubstr) <413>....334
4.3. Вставка одной строки в другую (strInsert) <1200>....334
4.4. Замена части строки другой (strReplace) <531>....335
4.5. Разбор строки по разделителю (strParse) <537>....335
4.6. Разбор пути (strParsePath) <300>....335
4.7. Путь в строку с разделителем (strPath2Sep)....335
4.8. Кодирование строки в HTML (strEnc2HTML)....336
4.9. Кодирование текста в бинарный вид (strEnc2Bin)....336
4.10. Декодирование текста из бинарного вида (strDec4Bin)....336
4.11. Преобразование вещественного в строку (real2str)....336
4.12. Преобразование целого в строку (int2str)....336
4.13. Преобразование строки в вещественное (str2real)....337
4.14. Преобразование строки в целое (str2int)....337
5. Функции работы с вещественным....337
5.1. Разделение float на слова (floatSplitWord) <56>....337
5.2. Объединение float из слов (floatMergeWord) <70>....337
Модуль подсистемы “Специальные” <SystemTests>....338
1. Параметр (Param)....339
2. Разбор XML (XML)....339
3. Сообщения (Mess)....339
4. Подключение SO (SOAttach)....340
5. Атрибут параметра (Val)....340
6. Тест БД (DB)....340
7. Транспорт (TrOut)....341
8. Язык управления системой (SysContrLang)....341
9. Буфер значений (ValBuf)....341
10. Архив значений (Archive)....341
11. Base64 кодирование (Base64Code)....341
Модуль подсистемы “Пользовательские интерфейсы” <QTStarter>....342
Модуль подсистемы “Пользовательские интерфейсы” <QTCfg>....344
1. Конфигурация....347
2. Базовые элементы....348
3. Команды....349
4. Списки....350
5. Таблицы....351
6. Изображения....352
Модуль подсистемы “Пользовательские интерфейсы” <WebCfg>....353
1. Базовые элементы....355
2. Команды....355
3. Списки....356
4. Таблицы....356

5. Изображения....357
Модуль подсистемы “Пользовательские интерфейсы” <WebCfgD>....358
1. Конфигурация....360
2. Базовые элементы....361
3. Команды....362
4. Списки....362
5. Таблицы....363
6. Изображения....364
7. Ошибки....365
Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>....367
1. Назначение....368
2. Конфигурация и формирование интерфейсов СВУ....369
3. Архитектура....369
3.1. Кадры и элементы отображения(виджеты)....371
3.2. Проект....374
3.3. Стили....378
3.4. События, их обработка и карты событий....379
3.5. Сигнализация....381
3.6. Управление правами....383
3.7. Связывание с динамикой....383
3.8. Примитивы виджетов....389
3.8.1. Элементарные графические фигуры (ElFigure)....392
3.8.2. Элементы формы (FormEl)....394
3.8.3. Элемент текста (Text)....396
3.8.4. Элемент отображения медиа-материалов (Media)....397
3.8.5. Элемент построения диаграмм/трендов (Diagram)....398
3.8.6. Элемент построения протоколов, на основе архивов сообщений (Protocol)....400
3.8.7. Элемент формирования отчётной документации (Document)....401
3.8.8. Контейнер (Box)....403
3.9. Использование БД для хранения библиотек виджетов и проектов....404
3.10 API пользовательского программирования и сервисные интерфейсы OpenSCADA....406
3.10.1. API пользовательского программирования....406
3.10.2. Сервисные интерфейсы OpenSCADA....408
4. Конфигурация модуля посредством интерфейса управления OpenSCADA....410
Модуль подсистемы “Пользовательские интерфейсы” <Vision>....418
1. Назначение....419
2. Инструмент графического формирования интерфейса СВУ....421
2.1. Стили....431
2.2. Связывание с динамикой....432
3. Исполнение интерфейсов СВУ....434
4. Представление базовых элементов (примитивов)....436
4.1. Примитив элементарная фигура (ElFigure)....437
4.2. Примитив текста (Text)....438
4.3. Примитив элементов формы (FormEl)....439
4.4. Примитив отображения медиа-материалов (Media)....440
4.5. Примитив построения диаграм/графиков (Diagram)....441
4.6. Примитив формирования протокола (Protocol)....441
4.7. Примитив формирования отчётной документации (Document)....442
4.8. Примитив контейнера (Box)....443
5. Общая конфигурация модуля....444
Модуль подсистемы “Пользовательские интерфейсы” <WebVision>....445
1. Назначение....446
2. Исполнение интерфейсов СВУ....447
3. Представление базовых элементов (примитивов)....450

3.1. Примитив элементарная фигура (ElFigure)....451
3.2. Примитив текста (Text)....452
3.3. Примитив элементов формы (FormEl)....453
3.4. Примитив отображения медиа-материалов (Media)....454
3.5. Примитив построения диаграмм/графиков (Diagram)....454
3.6. Примитив формирования протокола (Protocol)....455
3.7. Примитив формирования отчётной документации (Document)....455
3.8. Примитив контейнера (Box)....456
4. Общая конфигурация модуля....457
Заключение....457
Модуль подсистемы “Пользовательские интерфейсы” <WebUser>....458
1. WEB - страницы....460

Введение

OpenSCADA представляет собой открытую SCADA систему, построенную по принципам модульности, многоплатформенности и масштабируемости. SCADA (Supervisory Control And Data Acquisition) в переводе: «Системы диспетчерского управления и сбора данных» является термином, который часто употребляется в сфере автоматизации технологических процессов. Система OpenSCADA предназначена для: сбора, архивирования, визуализации информации, выдачи управляющих воздействий, а также других родственных операций, характерных для полнофункциональной SCADA системы.

Предпосылки

На рынке программ для АСУ-ТП (Автоматизированные системы управления технологическим процессом) сложилась ситуация, когда отсутствуют не только свободные SCADA системы, но и SCADA системы для платформ отличных от MS Windows+x86. Да, единицы есть, но про них почти ничего не слышно и выглядят они крайне блекло на фоне общей массы.

В тоже время фирмы, занимающиеся внедрением АСУ-ТП, заинтересованы в полном контроле над SCADA системой, внедряемой на объектах заказчика. Эта заинтересованность связана со спецификой работы с Заказчиком. Заказчик является последним звеном в цепочке создания программы. От качества взаимодействия Заказчика и разработчиков прямо пропорционально зависит удовлетворённость Заказчика, а также скорость совершенствования программы. Вытекая из вышесказанного, пропорционально зависит и имидж фирмы, внедряющей АСУ-ТП. Применение законченных коммерческих SCADA систем, как правило, приводит к ухудшению или же полному отсутствию взаимодействия Заказчика с разработчиками.

Реально существуют следующие варианты решения сложившейся проблемы:

- Первым вариантом может быть налаживание тесных связей с производителем SCADA систем, т.е интеграция (явная или неявная) в подразделение по внедрению. Что, в свою очередь, равносильно полной или частичной потере самостоятельности.
- Вторым вариантом является создание собственной коммерческой SCADA системы. Этим достигается полный контроль, однако для разработки системы, на высоком уровне, нужен огромный штат специалистов высокой квалификации. Что, в большинстве случаев, не по карману фирмам внедряющим АСУ-ТП. Как результат – квалификация специалистов довольно низка. И даже если фирма решается на подобный подвиг, то SCADA система не дотягивает до уровня самой посредственной сторонней коммерческой системы, и очень часто имеет место низкое качество кода и(или) посредственная функциональность системы.
- Третьим вариантом является использование открытых SCADA систем, то есть – совместная разработка. Это позволяет совместить преимущества вышеперечисленных вариантов:
    - полный контроль над SCADA системой;
    - не требуется огромного штата высококвалифицированных специалистов для развития системы, достаточно специалистов для дистрибьюции системы;
    - приводит к повышению качества системы, за счёт множественной внешней экспертизы и широты поддерживаемых платформ;
    - позволяет выбирать платформу в зависимости от её преимуществ, а не по причине того, что только на ней работает SCADA система;
    - ну и прочие психологические, экономические, моральные и юридические преимущества открытых систем.

Для совместной реализации третьего варианта и был основан проект OpenSCADA!

Цели проекта

Основными целями, которые преследует проект, являются:

- открытость;
- надежность;
- гибкость;
- масштабируемость;
- многоплатформенность;
- безопасность;
- финансовая доступность;
- предоставление удобного интерфейса управления;

Политика разработки. Лицензия.

В качестве политики реализации данного проекта выбраны «OPEN SOURCE» принципы разработки ПО. Данная политика позволит привлечь к разработке, тестированию, развитию, распространению и использованию продукта значительное количество разработчиков, энтузиастов и других заинтересованных лиц при минимальных финансовых затратах. Программа распространяется на условиях лицензии GPL v2.

Области применения

Система OpenSCADA предназначена для выполнения как обычных функций SCADA систем, так и для использования в смежных областях информационных технологий.

Система OpenSCADA может использоваться:

- на промышленных объектах, в качестве полнофункциональной SCADA системы;
- во встраиваемых системах, в качестве среды исполнения (в том числе и PLC);
- для построения различных моделей (технологических, химических, физических, электрических процессов);
- на персональных компьютерах, серверах и кластерах для сбора, обработки, представления и архивации информации о системе и её окружении.

В качестве базовой (хостовой) операционной системы (ОС) для разработки и использования выбрана ОС Linux, которая является стандартной POSIX совместимой ОС. Кроме того, ОС Linux является оптимальным компромиссом в вопросах:

- надёжности;
- гибкости/масштабируемости;
- доступности;
- популярности и распространенности.

Поскольку система OpenSCADA разрабатывается на стандартной POSIX ОС, по принципам кроссплатформенности, то её адаптация на остальные ОС не составит проблемы.

Архитектура

Сердцем системы является модульное ядро.

В зависимости от того, какие модули подключены, система может выполнять как функции различных серверов, так и функции клиентов клиент-серверной архитектуры. Собственно, архитектура системы позволяет реализовывать распределённые клиент-серверные системы любой сложности.

Для достижения высокого быстродействия, за счёт сокращения времени коммуникаций, архитектура позволяет объединять функции распределённых систем в одной программе.

Архитектурно, система OpenSCADA состоит из подсистем:

- *Подсистема безопасности.* Содержит списки пользователей и групп пользователей,

обеспечивает проверку прав на доступ к элементам системы и т.д.

- *Модульная подсистема баз данных.* Обеспечивает доступ к базам данных.
- *Модульная подсистема транспортов.* Обеспечивает коммуникацию с внешней средой, посредством различных коммуникационных интерфейсов.
- *Модульная подсистема коммуникационных протоколов обмена.* Тесно связана с подсистемой транспортов и обеспечивает поддержку различных протоколов обмена с внешними системами.
- *Модульная подсистема сбора данных (DAQ).* Обеспечивает сбор данных от внешних источников: контроллеров, датчиков и т.д. Кроме этого, подсистема может предоставлять среду для написания генераторов данных(модели, регуляторы ...).
- *Модульная подсистема архивов.* Содержит архивы двух типов: архивы сообщений и архивы значений. Способ архивирования определяется алгоритмом, который заложен в модуле архивирования.
- *Модульная подсистема пользовательских интерфейсов.* Содержит функции пользовательских интерфейсов.
- *Подсистема управление модулями.* Обеспечивает контроль над модулями.
- *Модульная подсистема специальных функций.* Содержит функции не вошедшие в остальные подсистемы. В настоящий момент к этим функциям относятся функции тестирования.

Исходя из принципа модульности, указанные выше модульные подсистемы могут расширять свою функциональность путём подключения модулей соответствующего типа.

Модульное ядро системы OpenSCADA выполняется в виде статической и совместно используемой библиотек. Это позволяет встраивать функции системы в существующие программы, а также создавать новые программы на основе модульного ядра системы OpenSCADA.

Однако модульное ядро является самодостаточным и может использоваться посредством простой запускающей программы.

Модули системы OpenSCADA хранятся в динамических библиотеках. Каждая динамическая библиотека может содержать множество модулей различного типа. Наполнение динамических библиотек модулями определяется функциональной связностью самих модулей. Динамические библиотеки допускают горячую замену, что позволяет, в процессе работы, производить обновление модулей. Метод хранения кода модулей в динамических библиотеках является основным для системы OpenSCADA, поскольку поддерживается практически всеми современными ОС. Это не исключает возможности разработки других методов хранения кода модулей.

Функциональные характеристики и требования системы OpenSCADA

1. Сфера применения системы OpenSCADA

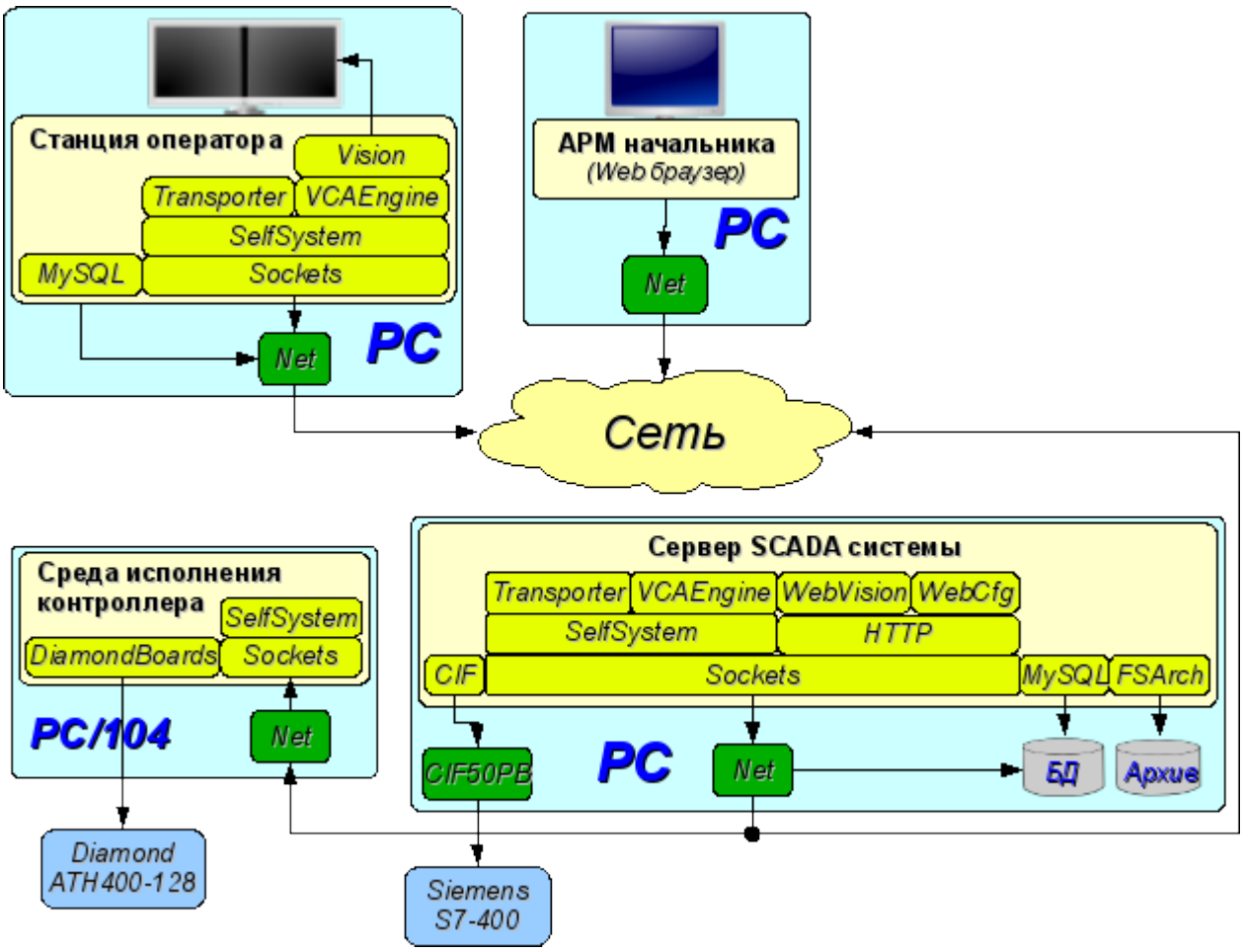


Рис. 1. Роли системы OpenSCADA

1.1. Сервер SCADA системы:

- Визуальный контроль и управление посредством интерфейсов:
  - Удалённый сервер визуализации на основе движка визуализации и управления СВУ VCAEngine. Локальный запуск модуля UI.Vision, подключеный к серверу визуализации.
  - Удалённый WEB интерфейс. Посредством WEB-браузера, модуля визуализации WebVision и модуля ядра среды визуализации и управления VCAEngine.
  - Простые удалённые Web-интерфейсы пользователя. Посредством WEB-браузера и UI-модуля WebUser.
- Сбор данных (DAQ) из источников:
  - Информации о платформе(программно-аппаратной) на которой работает сервер. Посредством DAQ-модуля System.
  - Сбор данных из источников поддерживающих протокол SNMP (Simple Network Management Protocol). Посредством DAQ-модуля SNMP.
  - Сбор данных промышленных контроллеров фирмы Siemens серии S7. Посредством DAQ-модуля Siemens.
  - Сбор данных промышленных контроллеров по протоколу ModBus. Посредством DAQ-модуля ModBus.
  - Сбор данных промышленных контроллеров по протоколу DCON. Посредством DAQ-модуля DCON.
  - Формирование производных структур параметров на основе шаблонов параметров и данных других источников данных. Посредством DAQ-модуля LogicLev.
  - Сбор данных из других серверов и PLC основанных на OpenSCADA, возможно для дублирования. Посредством DAQ-модуля DAQGate.
  - Сбор данных через входные каналы звуковых контроллеров. Посредством DAQ-модуля SoundCard.
  - Сбор данных оборудования фирмы ICP DAS. Посредством DAQ-модуля ICP_DAS.
  - Сбор данных из источников поддерживающих протокол OPC_UA. Посредством DAQ-модуля OPC_UA.
  - Сбор данных из источников различного типа, которые имеют утилиты для доступа к данным или доступны посредством простых специализированных сетевых протоколов. Осуществляется путём написания процедуры получения данных на языке пользовательского программирования DAQ-модуля JavaLikeCalc, а также модуля транспортного протокола User Protocol.
- Предоставление данных системам верхнего уровня:
  - Посредством интерфейсов:
    - Последовательного интерфейса (RS232, RS485, Modem, ...), с помощью модуля транспорта Serial.
    - Сокетов IP-сетей и протоколов сетевого уровня TCP, UDP и Unix, с помощью модуля транспорта Sockets.
    - Безопасного слоя сокетов (SSL), с помощью модуля транспорта SSL.
  - Посредством протоколов:
    - Собственный протокол OpenSCADA, с помощью модуля транспортного протокола SelfSystem.
    - Протоколов семейства ModBUS (TCP, RTU и ASCII), с помощью модуля транспортного протокола ModBUS.
    - Протокола "OPC UA", с помощью модуля транспортного протокола OPC UA.
    - Простых специализированных протоколов, разработаных посредством модуля транспортного протокола User Protocol.
- Выполнение пользовательских вычислений на языках:
  - Язык блочных схем. Посредством DAQ-модуля BlockCalc.
  - На Java-подобном языке высокого уровня. Посредством DAQ-модуля JavaLikeCalc.
- Архивирование сообщений, ведение протоколов по различным категориям и уровням, посредством механизмов:
  - Файлы в XML-формате или плоского текста, с упаковкой устаревших архивов. Посредством модуля архивирования FSArch.

    - В таблицы архивных БД. Посредством модуля архивирования DBArch.
    - В планах. На другой сервер, возможно выделенный сервер архивирования, основанных на OpenSCADA.
- Архивирование значений собранных данных посредством механизмов:
    - Файлы с двойной упаковкой: последовательной и стандартным архиватором gzip. Посредством модуля архивирования FSArch.
    - В таблицы архивных БД. Посредством модуля архивирования DBArch.
- Конфигурация и управление сервером через:
    - WEB-интерфейс. Посредством WEB-браузера и UI-модуля WebCfgD или WebCfg.
    - С удалённой конфигурационной станции. Посредством UI-модуля на конфигурационной станции QTCfg и интерфейса управления OpenSCADA отражённого в протоколе SelfSystem.
- Хранение данных сервера в БД типов:
    - MySQL. Посредством DB-модуля MySQL.
    - SQLite. Посредством DB-модуля SQLite.
    - PostgresSQL. Посредством DB-модуля PostgreSQL.
    - DBF. Посредством DB-модуля DBF.
    - FireBird. Посредством DB-модуля FireBird.
    - В планах. БД доступные на другом сервере основанном на OpenSCADA.
    - В планах. LDAP.

1.2. Станция оператора технологического процесса, пульт диспетчера, панель мониторинга и др.:

- Визуальный контроль и управление посредством интерфейсов:
    - Локальный (быстрый) интерфейс основанный на библиотеке QT. Посредством модуля визуализации Vision и модуля ядра среды визуализации и управления VCAEngine с возможностью визуализации из удалённого движка СВУ, сервера визуализации.
    - Удалённый WEB интерфейс. Посредством WEB-браузера, модуля визуализации WebVision и модуля ядра среды визуализации и управления VCAEngine.
    - Простые удалённые Web-интерфейсы пользователя. Посредством WEB-браузера и UI-модуля WebUser.
- Сбор данных (DAQ) из источников:
    - Сбор данных из других серверов и PLC основанных на OpenSCADA, для проброса данных серверов и дублирования. Посредством DAQ-модуля DAQGate.
    - Сбор данных из источников поддерживающих протокол SNMP (Simple Network Management Protocol). Посредством DAQ-модуля SNMP.
    - Сбор данных из источников поддерживающих протокол OPC_UA. Посредством DAQ-модуля OPC_UA.
- Выполнение пользовательских вычислений на языках:
    - Язык блочных схем. Посредством DAQ-модуля BlockCalc.
    - На Java-подобном языке высокого уровня. Посредством DAQ-модуля JavaLikeCalc.
- Архивирование локальных сообщений, ведение протоколов по различным категориям и уровням посредством механизмов:
    - Файлы в XML-формате или плоского текста, с упаковкой устаревших архивов. Посредством модуля архивирования FSArch.
    - В таблицы архивных БД. Посредством модуля архивирования DBArch.
    - В планах. На сервер, возможно выделенный сервер архивирования, основанных на OpenSCADA.
- Конфигурация и управление станцией через:
    - WEB-интерфейс. Посредством WEB-браузера и UI-модуля WebCfgD или WebCfg.
    - QT-интерфейс. Посредством UI-модуля QTCfg.
    - С удалённой конфигурационной станции. Посредством UI-модуля на конфигурационной станции QTCfg и интерфейса управления OpenSCADA отражённого в протоколе SelfSystem.
- Хранение данных станции в БД типов:

- MySQL. Посредством DB-модуля MySQL.
- SQLite. Посредством DB-модуля SQLite.
- PostgresSQL. Посредством DB-модуля PostgreSQL.
- DBF. Посредством DB-модуля DBF.
- FireBird. Посредством DB-модуля FireBird.
- В планах. БД доступные на другом сервере основанном на OpenSCADA.
- В планах. LDAP.

1.3. Среда исполнения контроллеров (PLC):

- Сбор данных (DAQ) из источников:
  - Платы сбора данных фирмы Diamond Systems. Посредством DAQ-модуля DiamondBoards.
  - Информации о платформе(программно-аппаратной) на которой работает сервер. Посредством DAQ-модуля System.
  - Сбор данных из источников поддерживающих протокол SNMP (Simple Network Management Protocol). Посредством DAQ-модуля SNMP.
  - Сбор данных промышленных контроллеров по протоколу ModBus. Посредством DAQ-модуля ModBus.
  - Сбор данных промышленных контроллеров по протоколу DCON. Посредством DAQ-модуля DCON.
  - Формирование производных структур параметров на основе шаблонов параметров и данных других источников данных. Посредством DAQ-модуля LogicLev.
  - Сбор данных из других серверов и PLC основанных на OpenSCADA, возможно для дублирования. Посредством DAQ-модуля DAQGate.
  - Сбор данных через входные каналы звуковых контроллеров. Посредством DAQ-модуля SoundCard.
  - Сбор данных оборудования фирмы ICP DAS. Посредством DAQ-модуля ICP_DAS.
  - Сбор данных из источников поддерживающих протокол OPC_UA. Посредством DAQ-модуля OPC_UA.
  - Сбор данных из источников различного типа, которые имеют утилиты для доступа к данным или доступны посредством простых специализированных сетевых протоколов. Осуществляется путём написания процедуры получения данных на языке пользовательского программирования DAQ-модуля JavaLikeCalc, а также модуля транспортного протокола User Protocol.
- Предоставление данных системам верхнего уровня:
  - Посредством интерфейсов:
    - Последовательного интерфейса (RS232, RS485, Modem, ...), с помощью модуля транспорта Serial.
    - Сокетов IP-сетей и протоколов сетевого уровня TCP, UDP и Unix, с помощью модуля транспорта Sockets.
    - Безопасного слоя сокетов (SSL), с помощью модуля транспорта SSL.
  - Посредством протоколов:
    - Собственный протокол OpenSCADA, с помощью модуля транспортного протокола SelfSystem.
    - Протоколов семейства ModBUS (TCP, RTU и ASCII), с помощью модуля транспортного протокола ModBUS.
    - Протокола "OPC UA", с помощью модуля транспортного протокола OPC UA.
    - Простых специализированных протоколов, разработаных посредством модуля транспортного протокола User Protocol.
- Управление, регулирование и выполнение других пользовательских вычислений на языках:
  - Язык блочных схем. Посредством DAQ-модуля BlockCalc.
  - На Java-подобном языке высокого уровня. Посредством DAQ-модуля JavaLikeCalc.
- Архивирование сообщений, ведение протоколов по различным категориям и уровням посредством механизмов:
  - Файлы в XML-формате или плоского текста, с упаковкой устаревших архивов. Посредством

модуля архивирования FSArch.
    - В таблицы архивных БД. Посредством модуля архивирования DBArch.
    - В планах. На другой сервер, возможно выделенный сервер архивирования, основанных на OpenSCADA.
- Архивирование значений собранных данных посредством механизмов:
    - Буфера в памяти предопределённой глубины. Посредством встроенного механизма архивирования значений ядра OpenSCADA.
    - Файлы с двойной упаковкой: последовательной и стандартным архиватором gzip. Посредством модуля архивирования FSArch.
    - В таблицы архивных БД. Посредством модуля архивирования DBArch.
- Конфигурация и управление PLC через:
    - WEB-интерфейс. Посредством WEB-браузера и UI-модуля WebCfgD или WebCfg.
    - С удалённой конфигурационной станции. Посредством UI-модуля на конфигурационной станции QTCfg и интерфейса управления OpenSCADA отражённого в протоколе SelfSystem.
- Хранение данных PLC в БД типов:
    - Все данные в конфигурационном файле (фиксированно).
    - MySQL. Посредством DB-модуля MySQL.
    - SQLite. Посредством DB-модуля SQLite.
    - PostgresSQL. Посредством DB-модуля PostgreSQL.
    - DBF. Посредством DB-модуля DBF.
    - FireBird. Посредством DB-модуля FireBird.
    - В планах. БД доступные на другом сервере основанном на OpenSCADA.
    - В планах. LDAP.

2. Требования OpenSCADA

2.1. Исполнение

Аппаратные требования системы OpenSCADA для её исполнения в различных ролях приведены в таблице 1. Программные требования для исполнения системы OpenSCADA и её модулей представлены в таблице 2.

Таблица 1. Аппаратные требования системы OpenSCADA и её модулей.

Роль	Требование
Сервер SCADA системы	**CPU:** x86_32 (более i586) или x86_64, частотой более 500 МГц **MEM:** 128 МБ **HDD:** 10 ГБ включая ОС и место для архивов
Станция оператора технологического процесса, пульт диспетчера, панель мониторинга и др.	**CPU:** x86_32 (более i586) или x86_64, частотой более 1 ГГц **MEM:** 512 МБ **HDD:** 4 ГБ включая ОС и без архивов
Среда исполнения контроллеров (PLC)	**CPU:** x86_32 (более i586) или x86_64, частотой более 133 МГц **MEM:** 32 МБ **HDD:** 32 МБ включая ОС и без архивов.

Таблица 2. Программные требования системы OpenSCADA и её модулей.

Компонент	Описание
Зависимости ядра системы OpenSCADA	
ОС Linux	Дистрибутив операционной системы Linux (ALTLinux, SuSELinux, Mandriva, ASPLinux, Fedora, Debian, Ubuntu ...)
"Стандартные библиотеки"	Стандартный набор библиотек: linux-gate, libstdc++, libgcc_s, libc, libdl, librt, libcrypt, libm, libpthread. Обычно уже доступны в установленном дистрибутиве. Особое требование это использование нативной библиотеки потоков NPTL, уже используется во всех современных дистрибутивах ОС Linux.
libgd	Графическая библиотека GD версия 2, желательно без поддержки XPM (исключена зависимость на библиотеку X-сервера) и с поддержкой FontConfig.
libexpat	Библиотека XML-парсера.
Модуль DB.MySQL	
libMySQL	Библиотека доступа к СУБД MySQL.
Модуль DB.SQLite	
libsqlite3	Библиотека доступа к встраиваемой БД SQLite версии 3.
Модуль DB.PostgreSQL	
libpq	Библиотека доступа к СУБД PostgreSQL версии более 8.3.0.
Модуль DB.FireBird	
FirebirdSS	СУБД FireBird версии 2. Часто отсутствует в дистрибутивах Linux и требует индивидуальной загрузки с официального сайта (http://www.firebirdsql.org)!
Модуль Transport.SSL	
libssl	Библиотека шифрования OpenSSL.
Модуль DAQ.SNMP	
libsnmp	Библиотека доступа к данным сетевых устройств по протоколу SNMP.
Модуль DAQ.System	
libsensors	Библиотека сенсоров аппаратуры версии 2 или 3.
Модуль DAQ.SoundCard	

Компонент	Описание
libportaudio	Библиотека кросплатформенного доступа к звуковым контроллерам версии 19 и более.
Модуль DAQ.OPC_UA	
libssl	Библиотека шифрования OpenSSL.
Модули: UI.Vision, UI.WebVision, Special.FLibSYS	
libfftw3	Библиотека быстрого разложения сигналов в ряд Фурье.
Модули: UI.QTStarter, UI.QTCfg, UI.Vision	
libQT4(libQtCore, libQtGui)	Библиотеки построения пользовательского графического интерфейса QT версии 4.3 и выше.

2.2. Сборка

Программные требования системы OpenSCADA для сборки ядра OpenSCADA и её модулей приведены в таблице 3.

Таблица 3. Зависимости сборки системы OpenSCADA и её модулей.

Компонент	Описание
Общие требования для сборки OpenSCADA	
ОС Linux	Дистрибутив операционной системы Linux (ALTLinux, SuSELinux, Mandriva, ASPLinux, Fedora, Debian, Ubuntu ...)
g++	Компилятор языка C++ из коллекции компиляторов GCC, включая библиотеку GLibC
autotools(automake, autoconf, libtool)	Инструменты формирования сборочной среды OpenSCADA. Нужны только в случае изменения сборочной среды OpenSCADA, например для добавления нового модуля или изменения фиксированных параметров сборки.
gettext	Группа утилит для подготовки и компиляции переводов интерфейса программ на различные языки в соответствии со стандартом интернационализации I18N.
libgd(devel)	Графическая библиотека GD версия 2, пакет для разработки, желательно без поддержки XPM (исключена зависимость на библиотеку X-сервера) и поддержкой FontConfig. Используется для построения трендов и других изображений в формате PNG, GIF и JPEG.
libexpat(devel)	Библиотека XML-парсера, пакет для разработки. Интерфейс управления OpenSCADA и другие компоненты построены на основе языка XML.
Модуль DB.MySQL	
libMySQL(devel)	Библиотека доступа к СУБД MySQL, пакет для разработки на языке C.
Модуль DB.SQLite	
libsqlite3(devel)	Библиотека доступа к встраиваемой БД SQLite версии 3, пакет для разработки.
Модуль DB.PostgreSQL	
libpq	Библиотека доступа к СУБД PostgreSQL версии более 8.3.0, пакет для разработки.
Модуль DB.FireBird	
FirebirdSS	СУБД FireBird версии 2, пакет для разработки. Часто отсутствует в дистрибутивах Linux и требует индивидуальной загрузки с официального сайта (http://www.firebirdsql.org)!
Модуль Transport.SSL	
libssl(devel)	Библиотека шифрования OpenSSL, пакет для разработки.
Модуль DAQ.JavaLikeCalc	

Компонент	Описание
bison	Программа генерации синтаксических анализаторов на основе грамматики языка.
Модуль DAQ.SNMP	
libsnmp(devel)	Библиотека доступа к данным сетевых устройств по протоколу SNMP, пакет для разработки.
Модуль DAQ.System	
libsensors(devel)	Библиотека сенсоров аппаратуры версий 2 или 3, пакет для разработки.
Модуль DAQ.Siemens	
glibc-kernheaders	Заголовки ядра Linux библиотеки GLibC.
Модуль DAQ.SoundCard	
libportaudio(devel)	Библиотека кросплатформенного доступа к звуковым контроллерам, пакет для разработки версии 19 и более.
Модуль DAQ.OPC_UA	
libssl(devel)	Библиотека шифрования OpenSSL, пакет для разработки.
Модули: UI.Vision, UI.WebVision, Special.FLibSYS	
libfftw3(devel)	Библиотека быстрого разложения сигналов в ряд Фурье, пакет для разработки.
Модули: UI.QTStarter, UI.QTCfg, UI.Vision	
libQT4(devel)	Библиотека построения пользовательского графического интерфейса QT версии 4.3 и выше, пакет для разработки.

Описание программы OpenSCADA

Данный раздел является описанием "open source" проекта системы именуемой "OpenSCADA". OpenSCADA представляет собой открытую SCADA систему, построенную по принципам модульности, многоплатформенности и масштабируемости.

В качестве политики разработки данной системы выбраны "open source" принципы. Выбор данной политики определяется необходимостью создания открытой, надёжной и общедоступной SCADA системы. Данная политика позволяет привлечь к разработке, тестированию, развитию, распространению и использованию продукта значительное количество разработчиков, энтузиастов и других заинтересованных лиц с минимизацией и распределением финансовых затрат.

Система OpenSCADA предназначена для сбора, архивирования, визуализации информации, выдачи управляющих воздействий, а также других родственных операций, характерных для полнофункциональной SCADA системы. Благодаря высокому уровню абстракции и модульности система может использоваться во многих смежных областях.

Система OpenSCADA может применяться:

- на промышленных объектах в качестве полнофункциональной SCADA системы;
- во встраиваемых (embeded) системах в качестве среды исполнения в том числе внутри PLC (программируемых логических контроллерах);
- для построения различных моделей (технологических, химических, физических, электрических процессов);
- на персональных компьютерах, серверах и кластерах для сбора, обработки, представления и архивации информации о системе и её окружении.

В качестве базовой (хостовой) операционной системы для разработки и использования выбрана ОС Linux, которая является оптимальным компромиссом в вопросах:

- надёжности (большая доля серверов и кластеров работает на GNU/Linux);
- гибкости/масштабируемости (ввиду своей открытости и модульности позволяет строить решения под любые требования);
- доступности (благодаря лицензии GPL является полностью свободной системой, а при высокой квалификации пользователя и бесплатной);
- популярности, развитости, поддержке, распространённости (система активно развивается множеством энтузиастов, фирм и государственными учреждениями во всем мире, получает всё большую поддержку на пользовательском и корпоративном рынке, активно внедряется в государственные структуры различных стран).

Поскольку проект разрабатывается и реализуется по принципам многоплатформенности, то не составляет проблемы портировать его на другие ОС, что в дальнейшем и планируется.

Сердцем системы является модульное ядро. И в зависимости от того, какие модули подключены, система может выступать как в роли различных серверов, так и в роли разнообразных клиентов, а также совмещать эти функции в одной программе. Это позволяет реализовывать клиент-серверную архитектуру SCADA системы на базе одних и тех же компонентов/модулей, экономя при этом машинную память, дисковое пространство, а также ценное время программистов.

Серверные конфигурации системы предназначены для сбора, обработки, выдачи воздействий, архивирования, протоколирования информации от различных источников, а также предоставления этой информации клиентам (UI, GUI, TUI ...). Модульная архитектура позволяет расширять функциональность сервера без его перегрузки.

Клиентские конфигурации могут строиться на основе различных графических библиотек (GUI/TUI ToolKits), как используя ядро программы и его модули (путём добавления к нему UI-user interface модуля), так и в качестве самостоятельного приложения, подключая ядро OpenSCADA как библиотеку.

Возможность гибкой конфигурации системы позволяет строить решения под конкретные требования надёжности, функциональности и размеры системы.

1. Функции системы.

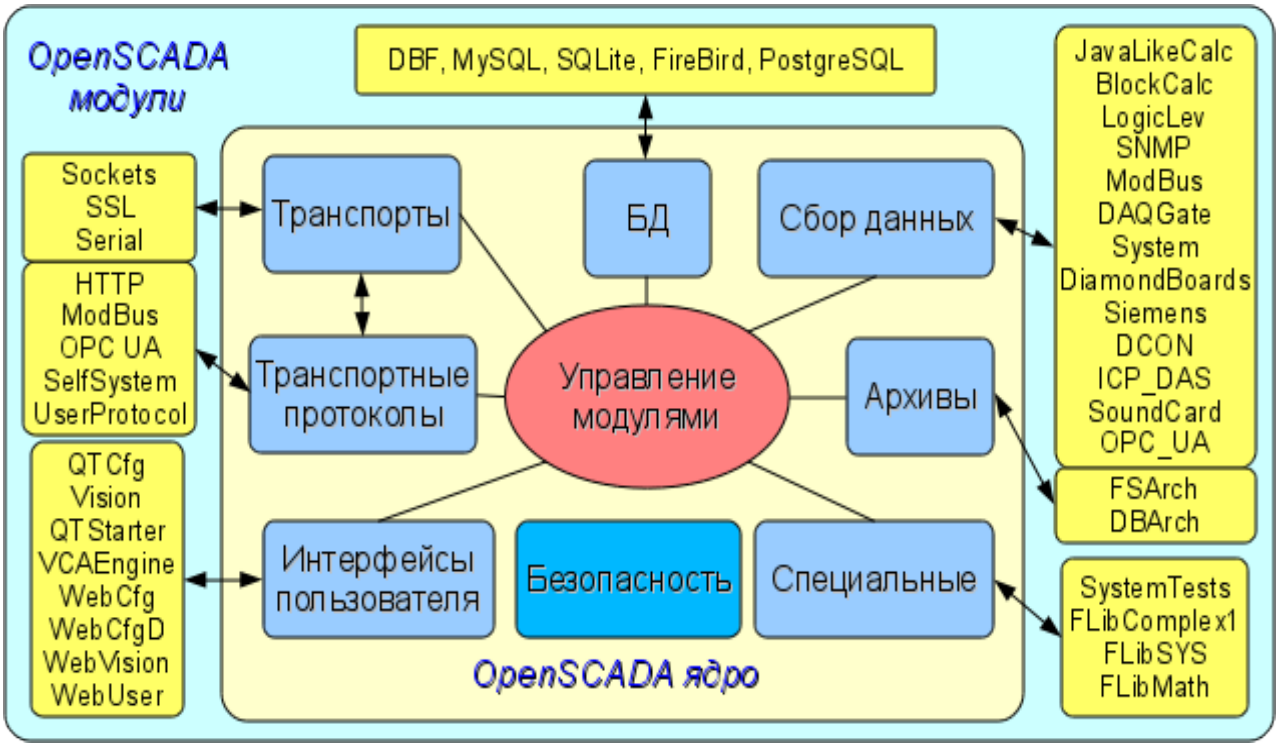


Рис. 1. Блочная схема системы OpenSCADA

1.1. Модульность.

Для придания гибкости и высокой степени масштабируемости система OpenSCADA построена по модульному принципу. Тесная интеграция модулей с ядром улучшает стабильность системы в целом, благодаря повторному использованию отлаженного кода. Однако сам процесс разработки собственного кода модулей OpenSCADA накладывает большую ответственность, возможные ошибки вводят элемент нестабильности в систему. Возможность создания распределённых конфигураций сглаживает эту опасность.

Модули системы OpenSCADA хранятся в динамических библиотеках. Каждая динамическая библиотека может содержать множество модулей различного типа. Наполнение динамических библиотек модулями определяется функциональной связностью самих модулей. Динамические библиотеки допускают горячую замену, что позволяет в процессе функционирования производить обновление отдельных частей системы. Метод хранения кода модулей в динамических библиотеках является основным для системы OpenSCADA, поскольку поддерживается практически всеми современными операционными системами(ОС). Однако это не исключает возможности разработки других методов хранения кода модулей.

На основе модулей реализованы следующие функциональные части системы OpenSCADA:

- базы данных;
- архивы (сообщений и значений);
- протоколы коммуникационных интерфейсов;
- коммуникационные интерфейсы, транспорты;
- источники данных и сбор данных;
- интерфейсы пользователя (GUI, TUI, WebGUI, speach, signal ... .);
- дополнительные модули, специальные.

Управление модулями осуществляется подсистемой «Управление модулями». Функциями подсистемы является: подключение, отключение, обновление модулей, а также другие операции, связанные с модулями и библиотеками модулей.

1.2. Подсистемы.

Архитектурно система OpenSCADA делится на подсистемы. Подсистемы могут быть двух типов: обычные и модульные. Модульные подсистемы обладают свойством расширения посредством модулей. Каждая модульная подсистема может содержать множество модульных объектов. Например, модульная подсистема «Базы данных» содержит модульные объекты типов баз данных. Модульный объект является корнем внутри модуля.

Всего система OpenSCADA содержит 9 подсистем из них 7 подсистем являются модульными. 9 подсистем системы OpenSCADA являются базовыми и присутствуют в любой конфигурации. К списку 9 подсистем могут добавляться новые подсистемы посредством модулей. Подсистемы системы OpenSCADA:

- Архивы (модульная).
- Базы данных (модульная).
- Безопасность.
- Интерфейсы пользователя (модульная).
- Управление модулями.
- Сбор данных (модульная).
- Транспортные протоколы (модульная).
- Специальные (модульная).
- Транспорты (модульная).

1.3. PLC и другие источники динамических данных. Подсистема "Сбор данных".

Для обеспечения поддержки источников динамических данных, будь то PLC-контроллеры, платы УСО, виртуальные источники и т.д., предназначена подсистема «Сбор данных». В функции этой подсистемы входит предоставление полученных данных в структурированном виде и обеспечение управления этими данными, например, модификация данных.

Подсистема «Сбор данных» является модульной и, как следствие, содержит модульные объекты типов источников динамических данных. Например, на октябрь 2007г, система OpenSCADA поддерживает следующие типы источников данных:

- Платы сбора данных от "Diamond systems".
- Сбор данных операционной системы (ОС).
- Блочный вычислитель.
- Вычислитель на Java-подобном языке.
- Транспортёр данных подсистемы "Сбор данных" от одной OpenSCADA станции к другой.
- Доступ к логическим контроллерам посредством протокола "ModBUS".
- Сбор данных сетевых устройств посредством протокола SNMP.
- Источник данных логического уровня системы OpenSCADA.
- Доступ к высокоинтеллектуальным логическим контроллерам посредством протокола MPI и коммуникационного процессора CIF50PB, фирмы Hilscher GMBH.

Каждый тип источника выполнен в виде отдельного модуля, который может быть подключен/отключен. Каждый тип источника может содержать отдельные источники (контроллеры).

Отдельно взятый контроллер может содержать параметры определённых модулем типов. Например, параметры аналогового типа; основной информацией, которую они предоставляет, является значение целого или вещественного типа. Структурно параметр представляет собой список атрибутов, которые и содержат данные. Атрибуты могут быть четырёх базовых типов: символьная строка(текст), целое, вещественное и логический тип.

Структуры контроллеров, параметров и их типов содержатся в подсистеме "Сбор данных", а объекты модулей выполняют их заполнение в соответствии с собственной спецификой.

Источник динамических данных может быть удалённым, т.е. быть подключен на удалённой системе OpenSCADA. Для связи с такими источниками данных используется транспортный тип

контроллеров (Transporter). Функцией данного типа источника данных является отражение источников данных удалённой OpenSCADA станции на локальную станцию.

1.4. Базы данных. Подсистема "Базы данных"

Для хранения данных системы повсеместно используются базы данных (БД). В целях систематизации доступа и управления базами данных в системе OpenSCADA предусмотрена подсистема "Базы данных". Для обеспечения поддержки различных БД/СУБД подсистема выполнена модульной.

В роли модульных объектов, содержащихся в подсистеме, выступает тип БД/СУБД, т.е. модуль подсистемы «Базы данных» практически содержит реализацию доступа к определённому типу БД. Например, модули: DBF, MySQL, SQLite.

Объект типа БД/СУБД в свою очередь содержит список объектов отдельных БД данного типа, а объект БД содержит список объектов таблиц, которые и содержат данные в табличной форме.

Практически все данные системы OpenSCADA хранятся в той или иной БД. Инструментарий системы позволяет легко переносить данные из одного типа БД в другой, и, как следствие, оптимально подбирать тип БД под конкретную область применения системы OpenSCADA. Перенос информации с одной БД в другую может быть выполнен двумя способами. Первый - это изменение адреса рабочей БД и сохранение всей системы на неё, второй - это прямое копирование информации между БД. Кроме копирования поддерживается и функция прямого редактирования содержимого таблиц БД.

Для организации централизованного доступа распределённой системы к единой БД предусматриваются два способа. Первый это использование сетевых СУБД, например, MySQL. Второй способ это использование транспортного типа БД на локальных системах для доступа к одной центральной БД (Планируется.). Функцией транспортной БД является пересылка запросов к БД на удалённую OpenSCADA систему.

Данные могут храниться также в конфигурационном файле системы. Реализован механизм полного отражения структуры БД на структуру конфигурационного файла. Т.е. стандартную конфигурацию можно размещать в конфигурационном файле. Суть такого механизма в том, что данные системы по умолчанию, например, при старте без БД можно описывать в конфигурационном файле. В дальнейшем эти данные могут переопределяться в БД. Кроме этого для случаев невозможности запуска какой либо БД вообще можно все данные хранить в конфигурационном файле.

Для доступа к базам данных используется механизм регистрации БД. Зарегистрированные в системе БД доступны всем подсистемам системы OpenSCADA и могут использоваться в их работе. Благодаря этому механизму можно обеспечить распределёность хранения данных. Например, различные библиотеки могут храниться и распространяться независимо, а подключение библиотеки будет заключаться в простой регистрации нужной БД.

В дальнейшем планируется реализация дублирования БД путём связывания зарегистрированных БД. Этот механизм позволит значительно повысить надёжность системы OpenSCADA в целом путём резервирования механизма хранения данных. (Планируется.)

1.5. Архивы. Подсистема "Архивы".

Любая SCADA система предоставляет возможность архивирования собранных данных, т.е. формирование истории изменения (динамики) процессов. Архивы условно можно разделить на два типа: архивы сообщений и архивы значений.

Особенностью архивов сообщений является то, что архивируются так называемые события. Характерным признаком события является время возникновения этого события. Архивы сообщений обычно используются для архивирования сообщений в системе, т.е. ведение логов и протоколов. В зависимости от источника сообщения могут классифицироваться по различным критериям. Например, это могут быть протоколы аварийных ситуаций, протоколы действий операторов,

протоколы сбоев связи и др.

Особенностью архивов значений является их периодичность, определяемая промежутком времени между двумя смежными значениями. Архивы значений применяются для архивирования истории непрерывных процессов. Поскольку процесс непрерывный, то и архивировать его можно только путём введения понятия квантования опроса значений, поскольку иначе мы получаем архивы бесконечных размеров, ввиду непрерывности самой природы процесса. Кроме этого практически мы можем получать значения с периодом ограниченным самими источниками данных. Например, довольно качественные источники данных в промышленности редко позволяют получать данные с частотой более 1 кГц. И это без учёта самих датчиков имеющих ещё менее качественные характеристики.

Для решения задач архивирования потоков данных в системе OpenSCADA предусмотрена подсистема "Архивы". Подсистема "Архивы" позволяет вести как архивы сообщений, так и архивы значений. Подсистема "Архивы" является модульной. Модульным объектом, содержащимся в подсистеме "Архивы", выступает тип архиватора. Тип архиватора определяет способ хранения данных, т.е. хранилище (файловая система, СУБД, сеть и т.д.). Каждый модуль подсистемы "Архивы" может реализовывать как архивирование сообщений, так и архивирование значений. Подсистема "Архивы" может содержать множество архивов, обслуживаемых различными модулями подсистемы.

Сообщение в системе OpenSCADA характеризуется датой, уровнем важности, категорией и текстом сообщения. Дата сообщения указывает на время создания сообщения. Уровень важности указывает на степень важности сообщения. Категория определяет адрес или условный идентификатор источника сообщения. Обычно, категория содержит полный путь к источнику сообщения в системе. Текст сообщения, собственно, и несёт смысловую нагрузку сообщения.

В процессе архивирования сообщения пропускаются через фильтр. Фильтр работает по уровню важности и категории сообщения. Уровень сообщения в фильтре указывает, что нужно пропускать сообщения с указанным или более высоким уровнем важности. Для фильтрования по категории применяются шаблоны, которые определяют какие сообщения пропускать. Каждый архиватор содержит собственные настройки фильтра. Следовательно можно легко создавать различные специализированные архиваторы для архива сообщений. Например, архиваторы сообщений можно специализировать на:

- логи, для хранения отладочной информации и другой рабочей информации сервера;
- различные протоколы (протокол действий клиентов, протокол нарушений и исключений, протокол событий ...).

Архив значений в системе OpenSCADA выступает как независимый компонент, который включает буфер, обрабатываемый архиваторами. Основным параметром архива значения является источник данных. В роли источника данных могут выступать атрибуты параметров системы OpenSCADA, а также другие внешние источники данных (пассивный режим). Другими источниками данных могут быть сетевые архиваторы удалённых OpenSCADA систем, среда программирования системы OpenSCADA и др.

Ключевым компонентом архивирования значений непрерывных процессов является буфер значений. Буфер значений предназначен для промежуточного хранения массива значений, полученных с определённой периодичностью (квантом времени). Буфер значений используется как для непосредственного хранения больших массивов значений в архивах значений как перед непосредственным «сбросом» на физические носители, так и для манипуляций с кадрами значений, т.е. в функциях покадрового запроса значений и их помещения в буфера архивов.

Для организации выделенных архиваторов в распределённых системах можно использовать транспортный тип архиватора (Планируется.). Функцией транспортного типа архиватора является отражение удалённого центрального архиватора на локальной системе. Как следствие архиваторы транспортного типа выполняют передачу данных между локальной системой и архиватором удалённой системы, скрывая от подсистем локальной системы реальную природу архиватора.

1.6. Коммуникации. Подсистемы "Транспорты" и "Транспортные протоколы".

Поскольку система OpenSCADA закладывается как высоко-масштабируемая система, то поддержка коммуникаций должна быть достаточно гибкой. Для удовлетворения высокой степени гибкости коммуникации в системе OpenSCADA реализованы в подсистемах "Транспорты" и "Транспортные протоколы", которые являются модульными.

Подсистема «Транспорты» предназначена для обмена неструктурированными данными между системой OpenSCADA и внешними системами. В роли внешних систем могут выступать и удалённые OpenSCADA системы. Под неструктурированными данными понимается массив символов определённой длины. Модульным объектом, содержащимся в подсистеме «Транспорты», выступает тип транспорта. Тип транспорта определяет механизм передачи неструктурированных данных. Например, это могут быть:

- сокеты (TCP/UDP/UNIX);
- каналы;
- разделяемая память.

Подсистема "Транспорты" включает поддержку входящих и исходящих транспортов. Входящий транспорт предназначен для обслуживания внешних запросов и отправки ответов. Исходящий транспорт, наоборот, предназначен для отправки сообщений и ожидания ответа. Следовательно, входящий транспорт содержит конфигурацию данной станции как сервера, а исходящий транспорт содержит конфигурацию удалённого сервера. Модуль подсистемы "Транспорты" реализует поддержку как входящего, так и исходящего транспортов.

Подсистема "Транспортные протоколы" предназначена для структуризации данных, полученных от подсистемы "Транспорты". По сути, подсистема "Транспортные протоколы" является продолжением подсистемы "Транспорты" и выполняет функции проверки структуры и целостности полученных данных. Так, для указания протокола, в связке с которым должен работать транспорт, предусмотрено специальное конфигурационное поле. Модульным объектом, содержащимся в подсистеме "Протоколы", является сам протокол. Например, транспортными протоколами могут быть:

- HTTP (Hyper Text Transfer Protocol);
- SelfSystem (OpenSCADA системный протокол).

Полную цепочку связи можно записать следующим образом:

- сообщение передаётся в транспорт;
- транспорт передаёт сообщение связанному с ним протоколу путём создания нового объекта протокола;
- протокол проверяет целостность данных;
- если пришли все данные, то сообщить транспорту о прекращении ожидания данных и передать ему ответ иначе сообщить, что нужно ожидать ещё;
- транспорт, получив подтверждение, отсылает ответ и удаляет объект протокола;
- если подтверждения нет, то транспорт продолжает ожидание данных, и в случае их поступления передаёт их сохранённому объекту протокола.

Поддерживаются протоколы и для исходящих транспортов. Исходящий протокол берёт на себя функцию общения с транспортом и реализацию особенностей протокола. Внутренняя сторона доступа к протоколу реализуется потоковым образом с собственной структурой для каждого протокольного модуля. Такой механизм позволяет выполнять прозрачный доступ к внешней системе, посредством транспорта, просто указывая имя протокола, с помощью которого обслуживать передачу.

Благодаря стандартному API-доступа к транспортам системы OpenSCADA можно легко менять способ обмена данными, не затрагивая самих обменивающихся систем. Например, в случае локального обмена можно использовать более быстрый транспорт на основе разделяемой памяти, а в случае обмена через интернет и локальную сеть использовать TCP или UDP сокеты.

1.7. Интерфейсы пользователя. Подсистема "Интерфейсы пользователя".

SCADA-системы, как класс, предполагают наличие интерфейсов пользователя. В OpenSCADA для предоставления пользовательских интерфейсов предусмотрена подсистема "Пользовательские интерфейсы". Под пользовательским интерфейсом системы OpenSCADA понимается не только среда визуализации, с которой должен работать конечный пользователь, но и всё, что имеет отношение к пользователю, например:

- среды визуализации;
- конфигураторы;
- сигнализаторы.

Подсистема "Пользовательские интерфейсы" является модульной. Модульным объектом подсистемы выступает собственно конкретный интерфейс пользователя. Модульность подсистемы позволяет создавать различные интерфейсы пользователей на различных GUI/TUI библиотеках и использовать наиболее оптимальное из решений в конкретно взятом случае, например, для сред исполнения программируемых логических контроллеров можно использовать конфигураторы и визуализаторы на основе Web-технологий (WebCfg, WebUI), а в случае стационарных рабочих станций использовать те же конфигураторы и визуализаторы, но на основе библиотек типа QT, GTK.

1.8. Безопасность системы. Подсистема "Безопасность".

Система OpenSCADA является разветвлённой системой, которая состоит из десятка подсистем и может включать множество модулей. Следовательно, предоставление всем неограниченного доступа к этим ресурсам является по крайней мере небезопасным. Поэтому для разграничения доступа в системе OpenSCADA предусмотрена подсистема "Безопасности". Основными функциями подсистемы "Безопасности" является:

- хранение учётных записей пользователей и групп пользователей;
- аутентификация пользователей;
- проверка прав доступа пользователя к тому или иному ресурсу.

1.9. Управление библиотеками модулей и модулями. Подсистема "Управление модулями".

Система OpenSCADA построена по модульному принципу, что подразумевает наличие множества модулей, которыми необходимо управлять. Для выполнения функции управления модулями системы OpenSCADA предусмотрена подсистема "Управление модулями". Все модули на настоящий момент поставляются в систему посредством разделяемых библиотек(контейнеров). Каждый контейнер может содержать множество модулей различного типа.

Подсистема "Управление модулями" реализует контроль за состоянием контейнеров и позволяет выполнять горячее добавление, удаление и обновление контейнеров и содержащихся в них модулей.

1.10. Непредусмотренные возможности. Подсистема "Специальные".

Разумеется, предусмотреть всех возможных функций невозможно, поэтому в системе OpenSCADA предусмотрена подсистема "Специальные". Подсистема "Специальные" является модульной и предназначена для добавления в систему OpenSCADA непредусмотренных функций путём модульного расширения. Например, с помощью подсистемы «Специальные» могут быть реализованы:

- тесты системы OpenSCADA и её модулей;
- библиотеки функций пользовательского программирования.

1.11. Пользовательские функции. Объектная модель и среда программирования системы.

Любая современная SCADA система должна содержать механизмы, предоставляющие возможность программировать на пользовательском уровне, т.е. содержать среду программирования. Система OpenSCADA содержит такую среду. С помощью среды программирования системы OpenSCADA можно реализовывать:

- Алгоритмы управления технологическими процессами.
- Крупные динамические модели реального времени технологических, химических, физических и других процессов.
- Адаптивные механизмы управления по моделям.
- Пользовательские процедуры управления внутренними функциями системы, её подсистемами и модулями.
- Гибкое формирования структур параметров на уровне пользователя, с целью создания параметров нестандартной структуры и заполнения её по алгоритму пользователя.
- Вспомогательные вычисления.

Среда программирования системы OpenSCADA представляет собой комплекс средств, организующих вычислительное окружение пользователя. В состав комплекса средств входят:

- объектная модель системы OpenSCADA;
- модули библиотек функций;
- вычислительные контроллеры подсистемы «Сбор данных» и другие вычислители.

Модули библиотек функций предоставляют множество функций определённой направленности, расширяющих объектную модель системы. Библиотеки могут реализоваться как набором функций фиксированного типа, так и функциями, допускающими свободную модификацию и дополнение.

Библиотеки функций фиксированного типа могут предоставляться стандартными модулями системы, органично дополняя объектную модель. Функции таких библиотек будут представлять собой интерфейс доступа к средствам модуля на уровне пользователя. Например, «Среда визуального представления данных» может предоставлять функции для выдачи различных сообщений. Используя эти функции, пользователь может реализовывать интерактивные алгоритмы взаимодействия с системой.

Библиотеки функций свободного типа предоставляют среду написания пользовательских функций на одном из языков программирования. В рамках модуля библиотек функций могут предоставляться механизмы создания библиотек функций. Так, можно создавать библиотеки аппаратов технологических процессов, а в последствии использовать их путём связывания. Различные модули библиотек функций могут предоставлять реализации различных языков программирования.

На основе функций, предоставляемых объектной моделью, строятся вычислительные контроллеры. Вычислительные контроллеры выполняют связывание функций с параметрами системы и механизмом вычисления.

2. SCADA системы и их структура.

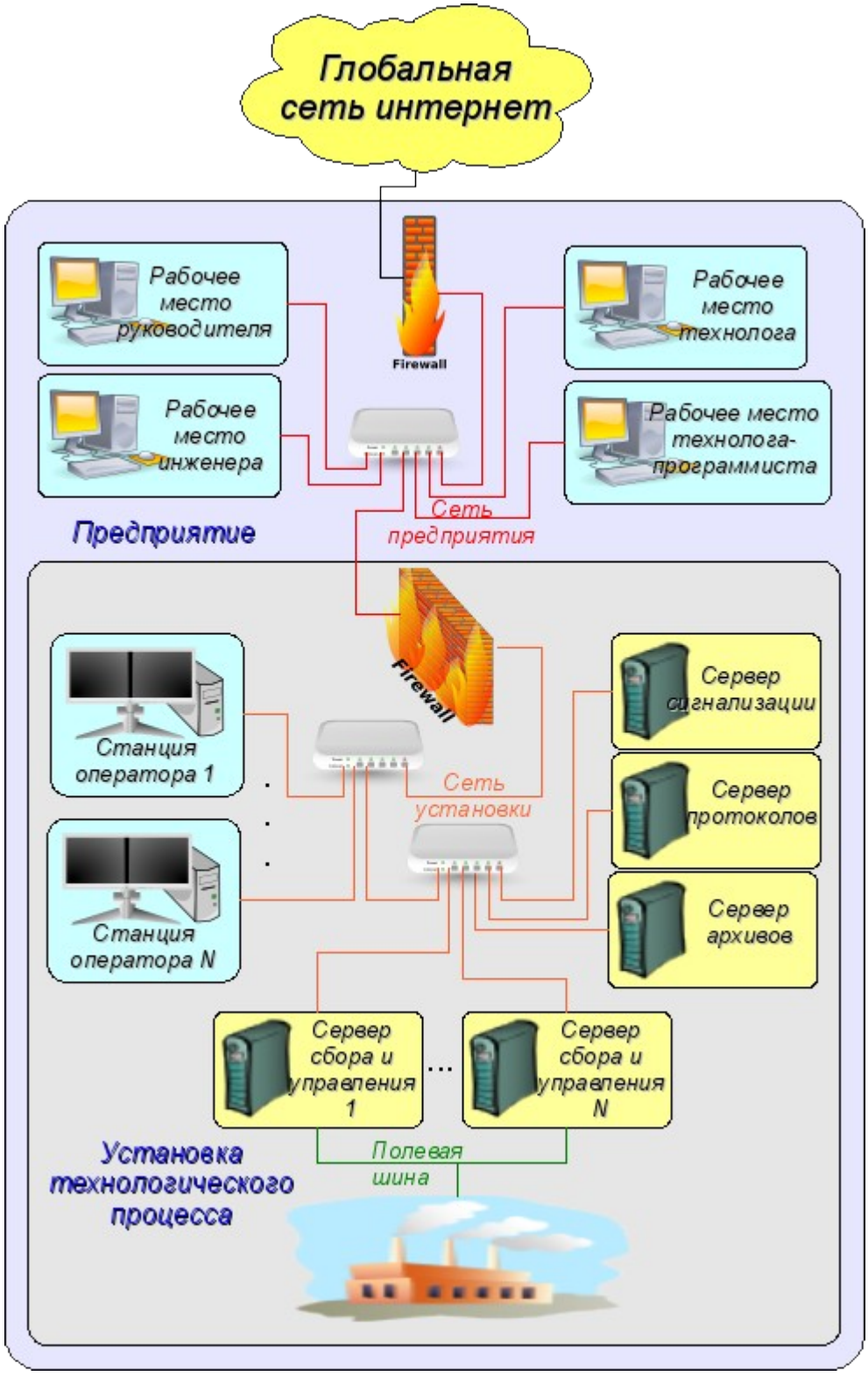


Рис. 2. SCADA-система.

SCADA (Supervisory Control And Data Acquisition), в общем виде, имеют распределённую архитектуру вроде изображённой на рис. 2. Элементы SCADA систем, в смысле программного обеспечения, выполняют следующие функции: **Сервер сбора:** представляет собой задачу или группу задач занимающихся сбором данных из источников данных, или же сами выступают в роли источником данных. В задачи сервера входит:

- получение и/или формирование данных;
- обработка данных;

- обслуживание запросов на доступ к данным;
- обслуживание запросов на модификацию данных.

Сервер архивирования: представляет собой задачу или группу задач занимающихся архивированием данных. В задачи сервера входит:

- архивирование данных SCADA-системы;
- обслуживание запросов на доступ к архивным данным;
- импорт/экспорт архивов.

Сервер протоколирования: представляет собой задачу или группу задач занимающиеся архивированием сообщений. В задачи сервера входит:

- архивирование сообщений узлов SCADA-системы;
- обслуживание запросов на доступ к архивным сообщениям;
- импорт/экспорт архивов.

Сервер сигнализации: представляет собой задачу или группу задач выполняющие функции сервера протоколирования в отношении узкой категории сообщений сигнализации.

Рабочее место оператора: представляет собой постоянно функционирующее GUI(Grafical User Interface) приложение выполненное в одномониторном, многомониторном или панельном режиме и выполняющее функции:

- предоставление пользовательского интерфейса для контроля за состоянием технологического процесса;
- предоставление возможности формирования управляющих воздействий;
- предоставление возможности изучения и анализа истории технологического процесса;
- предоставление инструментария для генерации отчётной документации.

Рабочее место инженера: представляет собой GUI приложение используемое для конфигурации SCADA системы. В задачи приложения входит:

- предоставление инструментария для манипуляции системными функциями системы;
- предоставление инструментария рабочего места оператора;
- предоставление инструментария для манипуляции архитектурой SCADA системы в целом (распределение функций между станциями, создание удаление станций ...).

Рабочее место руководителя: представляет собой GUI приложение, как правило, выполненное в одномониторном режиме и выполняющее функции:

- предоставление пользовательского интерфейса для контроля за состоянием технологического процесса;
- предоставление инструментария для изучения и анализа истории технологического процесса как непосредственно с активного сервера, так и на основе отдельных архивов;
- предоставление инструментария для генерации отчётной документации.

Рабочее место технолога: полностью включает в себя функции рабочего места оператора плюс модель технологического процесса (без непосредственной связи с технологическим процессом).

Рабочее место технолога-программиста: полностью включает в себя функции рабочего места технолога плюс инструментарий для создания моделей технологических процессов.

3. Варианты конфигурирования и использования.

3.1. Простое серверное подключение.

В простейшем случае систему OpenSCADA можно сконфигурировать в серверном режиме (рис. 3.1) для сбора и архивирования данных. Данная конфигурация позволяет выполнять следующие функции:

- опрос контроллеров;
- архивирование значений параметров;
- обслуживание клиентских запросов на получение различных данных сервера;
- предоставление конфигурационного WEB-интерфейса;
- удалённая конфигурация из системы OpenSCADA посредством QT-интерфейса или другого локального интерфейса.
- вторичное регулирование (регулирование в вычислительных контроллерах);
- моделирующие, корректирующие и дополняющие вычисления в вычислительных контроллерах.

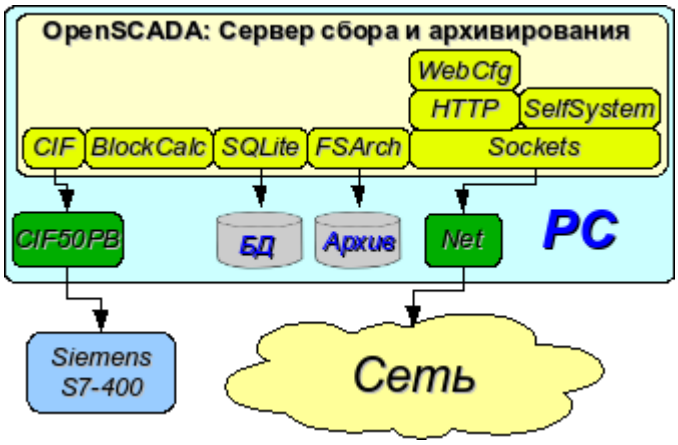


Рис. 3.1. Простое серверное подключение.

3.2. Дублированное серверное подключение.

Для повышения надёжности и производительности система OpenSCADA допускает множественное резервирование (рис. 3.2), при котором контроллеры одного экземпляра отражаются в другом. При использовании подобной конфигурации возможно распределение нагрузки опроса/вычисления на различных станциях. Данная конфигурация позволяет выполнять функции:

- опрос контроллеров;
- архивирование значений параметров;
- обслуживание клиентских запросов на получение различных данных сервера;
- резервирование параметров;
- резервирование архивов;
- распределение нагрузки опроса по серверам;
- предоставление конфигурационного WEB интерфейса;
- вторичное регулирование (регулирование в вычислительных контроллерах);
- моделирующие, корректирующие и дополняющие вычисления в вычислительных контроллерах с возможностью распределения нагрузки по серверам.

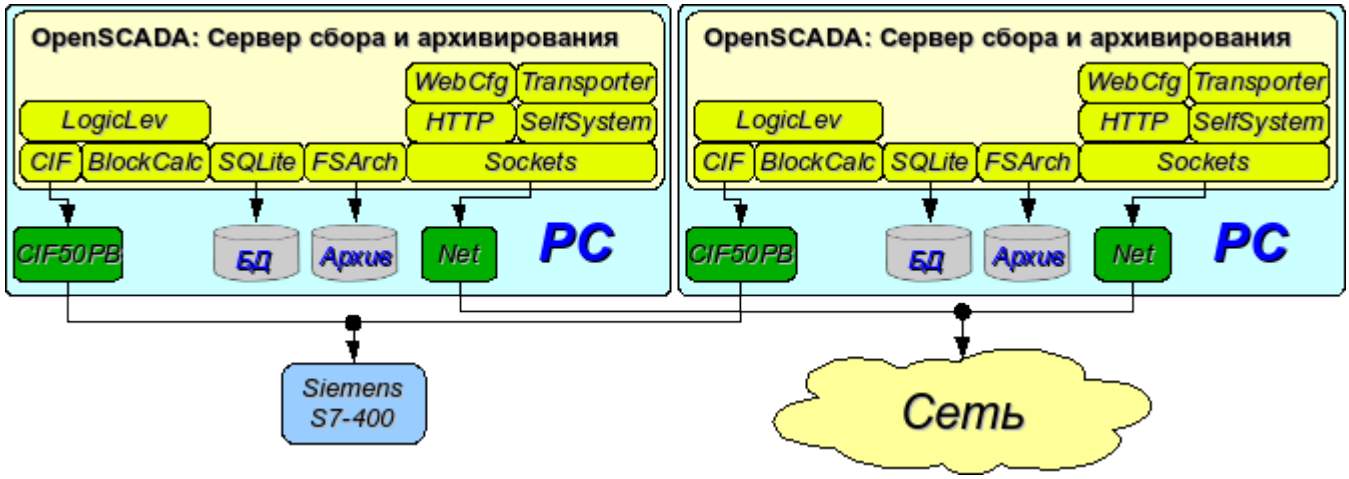


Рис. 3.2. Дублированное серверное подключение.

3.3. Дублированное серверное подключение на одном сервере.

Частным случаем дублированного соединения является дублированное соединение в рамках одного сервера (рис. 3.3), т. е запуск нескольких станций на одной машине с перекрещиванием параметров. Целью данной конфигурации является повышение надёжности и отказоустойчивости системы путём резервирования ПО.

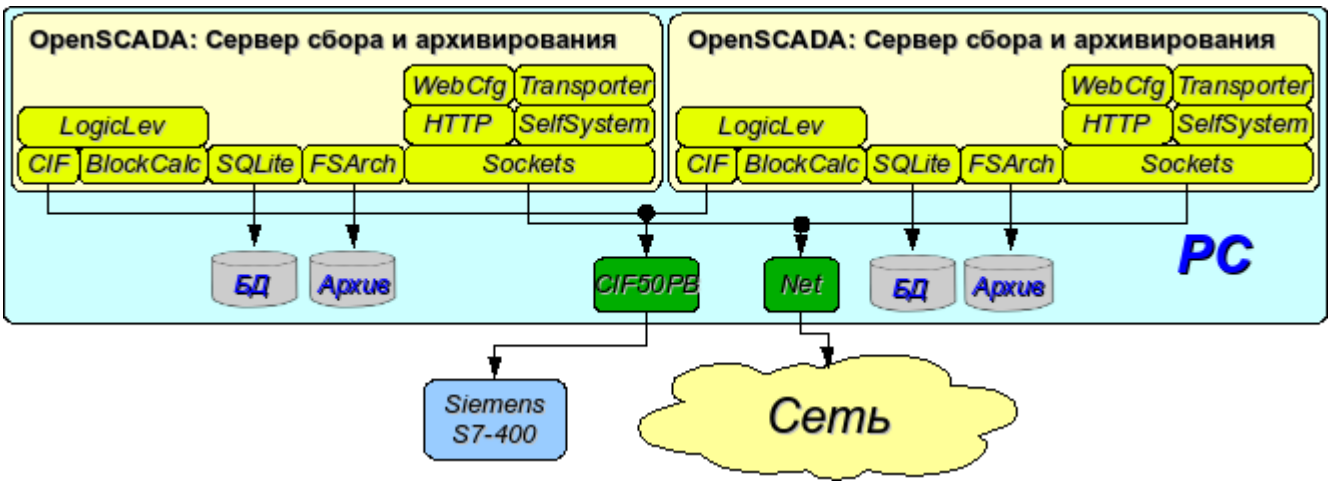


Рис. 3.3. Дублированное серверное подключение на одном сервере.

3.4. Клиентский доступ посредством Web-интерфейса. Место руководителя.

Для визуализации данных, содержащихся на сервере, хорошим решением является использование пользовательского WEB-интерфейса (рис. 3.4). Данное решение позволяет использовать стандартный WEB-браузер у клиента и следовательно является наиболее гибким, поскольку не привязано к одной платформе, т.е. является многоплатформенным. Однако это решение имеет существенные недостатки – это невысокая производительность и надёжность. В связи с этим рекомендуется использовать данный метод для визуализации некритичных данных или данных, имеющих резервный высоконадёжный способ визуализации. Например, хорошим решением будет использование этого метода у начальства промышленных установок, где всегда существует операторская с надёжным способом визуализации. Данная конфигурация позволяет выполнять следующие функции:

- опрос сервера на предмет получения данных визуализации и конфигурации;
- визуализация данных в доступном для понимания виде;
- формирование протоколов, отчётов;
- манипуляция параметрами, допускающими изменение.

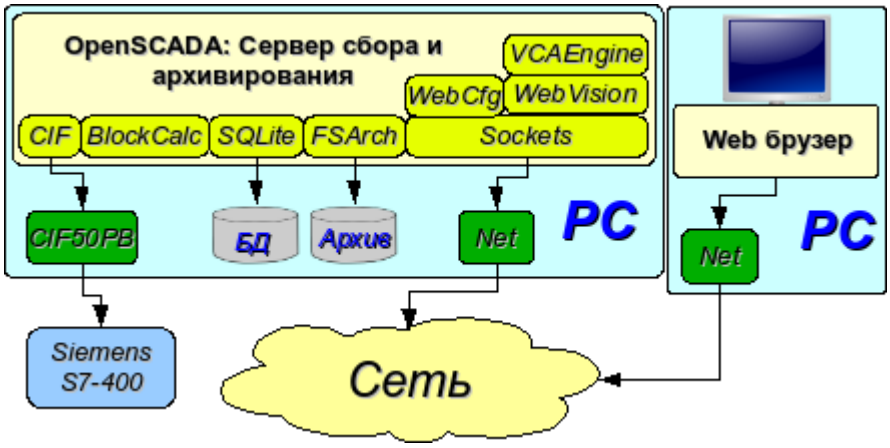


Рис. 3.4. Клиентский доступ посредством Web-интерфейса. Место руководителя.

3.5. Автоматизированное рабочее место (место руководителя/оператора).

Для визуализации критических данных, а также в случае если требуется высокое качество и производительность, можно использовать визуализацию на основе системы OpenSCADA сконфигурированной с GUI модулем (рис. 3.5). Данная конфигурация позволяет выполнять следующие функции:

- опрос сервера на предмет обновления текущих значений;
- визуализация опрошенных данных в доступном для понимания виде;
- формирование протоколов и отчётов;
- манипуляция параметрами, допускающими изменения.

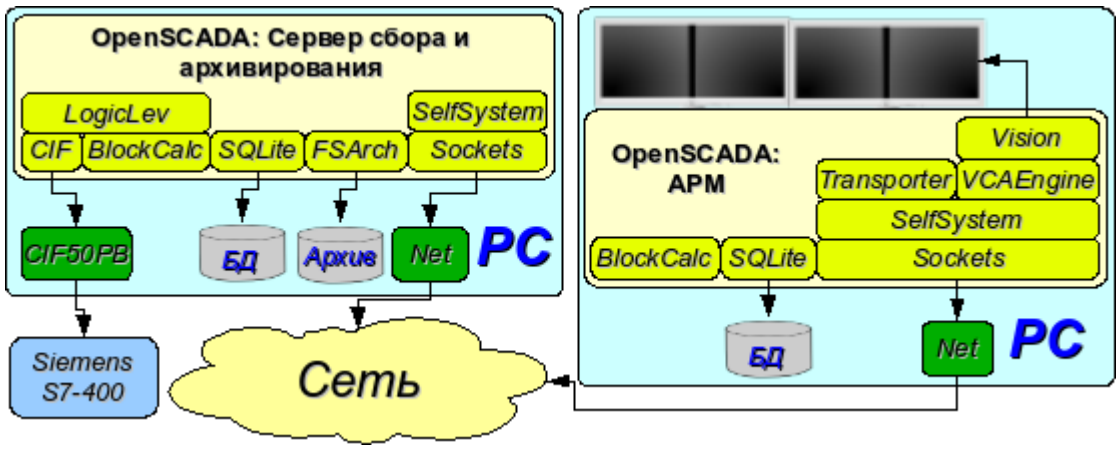


Рис. 3.5. Автоматизированное рабочее место (место руководителя/оператора)

3.6. АРМ с сервером сбора и архивирования на одной машине (место оператора, модель ...).

Полнофункциональная клиент-серверная конфигурация на одной машине (рис. 3.6) может использоваться для повышения надёжности системы в целом путём запуска клиента и сервера в разных процессах. Данная конфигурация позволяет без последствий для сервера останавливать клиент и выполнять с ним различные профилактические работы. Рекомендуется для использования на станциях оператора путём установки двух машин совмещающих в себе станции оператора и резервированный сервер. Данная конфигурация позволяет выполнять следующие функции:

- опрос контроллеров;
- обслуживание клиентских запросов;
- визуализация;
- выдача управляющих воздействий;
- генерация протоколов и отчётов;
- вторичное регулирование;
- моделирующие, корректирующие и дополнительные вычисления в вычислительных контроллерах;
- сбор и визуализация информации о персональном компьютере, сервере ... .

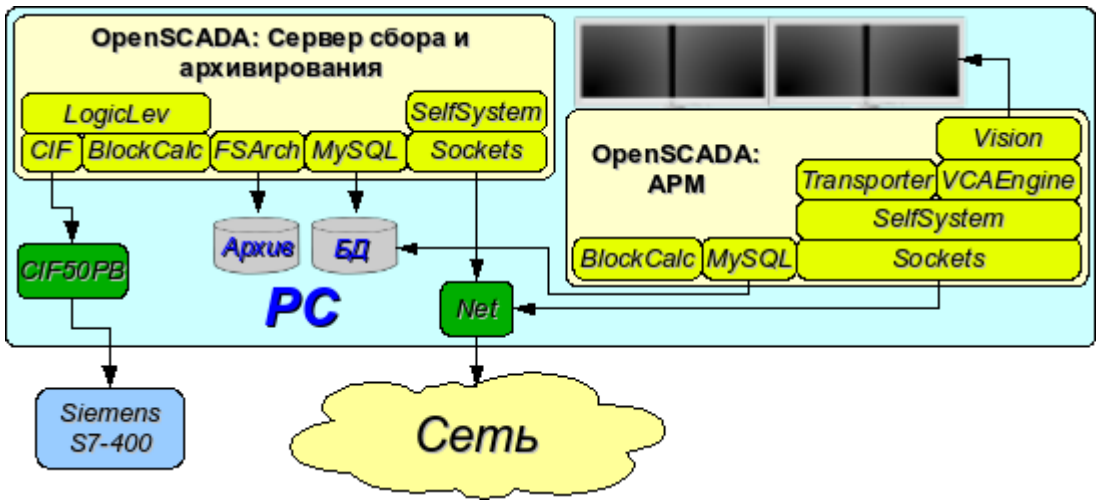


Рис. 3.6. АРМ с сервером сбора и архивирования на одной машине (место оператора, модель ...)

3.7. Простейшее смешанное подключение (модель, демонстрация, конфигуратор ...).

Смешанное подключение совмещает функции сервера и клиента (рис. 3.7). Может использоваться для тестовых, демонстрационных функций, а также для предоставления моделей технологических процессов как единое целое. В этом режиме могут выполняться следующие функции:

- опрос контроллеров;
- обслуживание клиентских запросов;
- визуализация;
- выдача управляющих воздействий;
- генерация протоколов и отчётов;
- вторичное регулирование;
- моделирующие, корректирующие и дополняющие вычисления в вычислительных контроллерах;
- сбор и визуализация текущей информации о персональном компьютере, сервере, модели ... ;
- конфигурация баз данных, подключений и др.

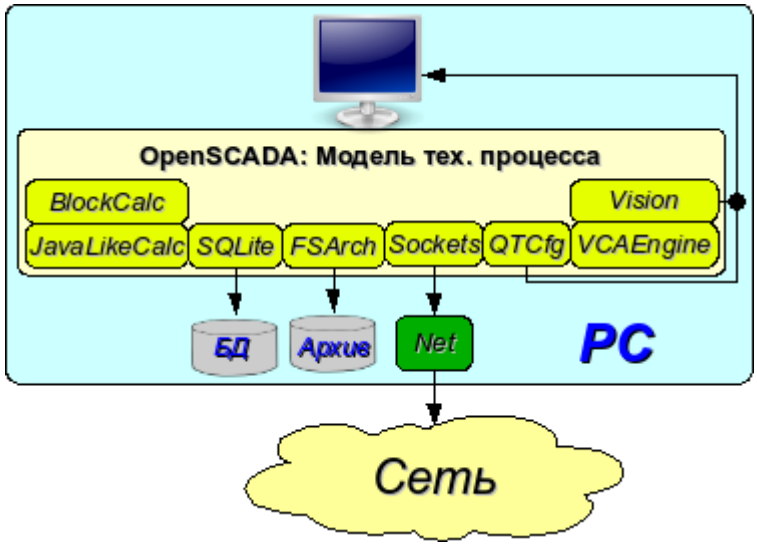


Рис. 3.7. Простейшее смешанное подключение (модель, демонстрация, конфигуратор ...)

3.8. Устойчивая, распределённая конфигурация.

Данная конфигурация является одним из вариантов устойчивого/надёжного соединения (рис. 3.8). Устойчивость достигается путём распределения функций по:

- серверам опроса;
- центральному серверу архивирования и обслуживания клиентских запросов;
- клиентам: АРМы и WEB-клиенты.

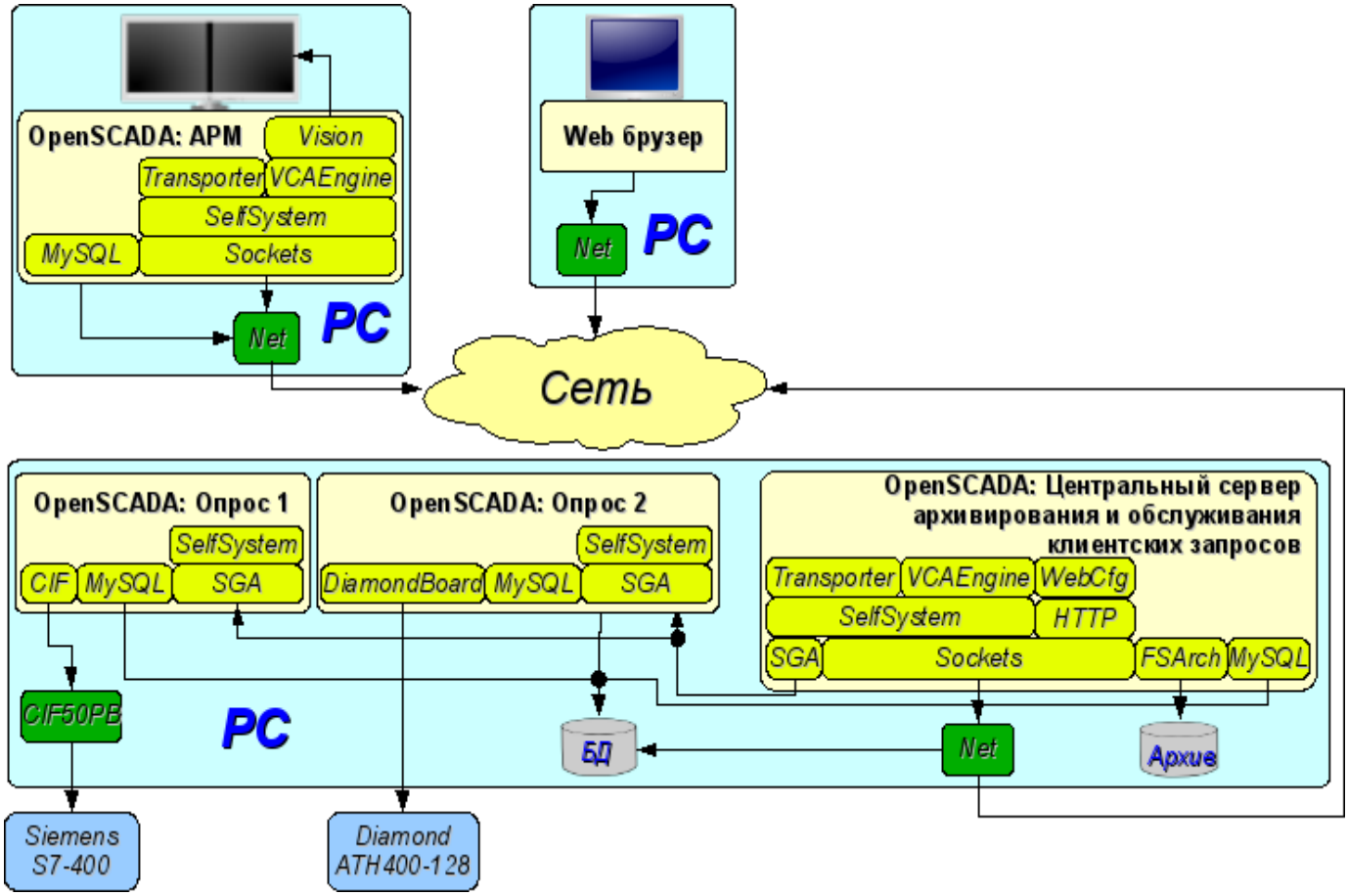


Рис. 3.8. Устойчивая, распределённая конфигурация

Сервер опроса конфигурируется на основе системы OpenSCADA и представляет собой задачу (группу задач), занимающихся опросом контроллера (группы контроллеров одного типа). Полученные значения доступны центральному серверу через любой транспорт, поддержка которого добавляется путём подключения соответствующего модуля транспорта. Для снижения частоты опроса и величины сетевого трафика, сервер опроса может оснащаться небольшим архивом значений. Конфигурация сервера опроса хранится в одной из доступных БД.

Центральный сервер архивирования и обслуживания клиентских запросов выполняет функцию централизованного сбора и обработки параметров серверов опроса и их значений. Доступ к серверам опроса выполняется посредством одного из доступных в OpenSCADA транспортов+протоколов (на примере это SGA). Для предоставления единого интерфейса доступа к параметрам и контроллерам используется модуль Transporter, который отражает данные серверов опроса на структуру локальных параметров.

Для выполнения внутренних вычислений и дополнительного анализа параметров используются вычислительные контроллеры.

Для разностороннего и глубокого архивирования используются различные модули архивов.

Для доступа клиентов к серверу используются доступные для OpenSCADA сетевые транспорты, на примере - это Sockets, и транспортные протоколы, на примере - это протокол OpenSCADA "SelfSystem".

Конфигурация центрального сервера хранится в одной из доступных БД (на примере это сетевая СУБД MySQL).

Для предоставления пользовательского WEB-интерфейса используется модуль WebCfg посредством транспортного протокола "HTTP".

Различные клиенты в их числе АРМ и WEB-клиенты выполняются на отдельных машинах в необходимом количестве. АРМ реализуется на основе системы OpenSCADA. В его функции входит опрос значений параметров из центрального сервера и их визуализация на GUI интерфейсе(ах). Для получения параметров в АРМ также используется модуль отражения удалённых параметров Transporter. Для предоставления доступа к архивам может использоваться модуль архива сетевого типа. Конфигурация АРМ может храниться в одной из доступных БД (в примере это сетевая СУБД MySQL, расположенная на машине центрального сервера архивирования).

4. Конфигурация и настройка системы.

Как можно видеть в разделе выше, OpenSCADA предоставляет возможность конфигурации для исполнения в различных ролях. Поддержка этой возможности обеспечивается развитыми механизмами конфигурации и хранения конфигурационных данных. Данный раздел содержит описание этих механизмов, призванное дать представление о гибкости и разнообразии, позволив тем самым использовать OpenSCADA на 100%.

При описании механизмов конфигурации и способов её хранения в этом разделе будет делаться упор на описание общесистемных механизмов. Особенности конфигурации и использования модулей подсистем OpenSCADA предоставляются в собственной документации модулей.

В OpenSCADA используется формализованный подход к описанию конфигурационных интерфейсов, основанный на языке XML. Фактически особенности конфигурации компонента системы предоставляется самим компонентом, пронизывая тем самым всю систему, как нервная система организма. В терминах OpenSCADA это называется интерфейсом контроля OpenSCADA (Control interface). На основе интерфейса контроля формируются графические интерфейсы конфигурации пользователя посредством модулей OpenSCADA. Такой подход имеет следующие важные преимущества:

- Масштабируемость. Можно подключать только нужные модули конфигурации или вообще использовать только удалённые механизмы.
- Исключение необходимости обновления конфигураторов с добавлением нового модуля/функции, а также исключение "распухания" конфигуратора, обеспечивающего поддержку всей истории уже ненужных и устаревших модулей/функций.
- Простота создания графических интерфейсов конфигурации на различной основе за счёт чёткой формализованости.
- Предоставляется возможность динамической конфигурации, т.е. конфигурацию можно выполнять непосредственно при работе системы как локально, так и удалённо, непосредственно контролируя результат.
- Простая и целевая расширяемость конфигурационного интерфейса, путём добавления полей конфигурации на языке описания интерфейса управления только в компонентах, этого требующих.

В OpenSCADA уже предоставляется три модуля конфигурации на разной основе визуализации. Отметим их и их возможности конфигурации:

- Модуль конфигурации на библиотеке графического интерфейса QT(http://www.trolltech.com/qt) - UI.QTCfg. Предоставляет развитый интерфейс конфигурации, позволяющий управлять как локальной станцией, так и удалёнными станциями в локальной и глобальной сетях, включая безопасное соединение.
- Модуль конфигурации на основе динамических WEB-технологий (DHTML) - UI.WebCfgD. Предоставляет развитый интерфейс конфигурации, позволяющий управлять как локальной станцией сервера, так и удалёнными станциями в локальной и глобальной сетях, включая работу по безопасному соединению. Клиентское подключение осуществляется посредством обычного Web-браузера.
- Модуль конфигурации на основе статических WEB-технологий (XHTML) - UI.WebCfg. Предоставляет достаточный интерфейс конфигурации, позволяющий управлять локальной станцией сервера посредством обычного Web-браузера.

Значения конфигурации, изменённые в конфигураторах, а также большинство данных сохраняются в базах данных (БД). Учитывая модульность подсистемы "БД", ими могут быть различные БД. Причём предоставляется возможность хранения разных частей OpenSCADA как в разных БД одного типа, так и в БД разных типов.

Кроме БД данные о конфигурации могут содержаться в конфигурационном файле OpenSCADA, а также передаваться посредством параметров командной строки при вызове OpenSCADA. Однако сохранение изменений конфигурации в конфигураторах доступно только в БД. Типовым именем конфигурационного файла OpenSCADA является /etc/oscada.xml. Формат конфигурационного файла и параметры командной строки рассмотрим в отдельном разделе.

Дальнейшее рассмотрение конфигурации OpenSCADA будет производиться на основе интерфейса конфигуратора UI.QTCfg, однако принципы работы будут полностью соответствовать и остальным конфигураторам благодаря общности в используемом интерфейсе контроля OpenSCADA.

Рассмотрение начнём с конфигурации системных параметров OpenSCADA, которая размещается в трёх вкладках корневой страницы станции:

- Вкладка "Станция" содержит основные информационные и конфигурационные поля станции, рис.4а. Перечислим предоставляемые поля и прокомментируем их:
  - *ID* - содержит информацию об идентификаторе станции. Указывается параметром командной строки -Station. При загрузке ищется соответствующий идентификатору станции раздел в конфигурационном файле и, если не обнаруживается, то используется первый доступный.
  - *Имя станции* - указывает локализованное имя станции.
  - *Программа* - содержит информацию об имени программы. Обычно это OpenSCADA или имя основанного на OpenSCADA решения.
  - *Версия* - содержит информацию о текущей версии программы.
  - *Имя хоста* - содержит информацию о имени машины, на которой запущена станция.
  - *Системный пользователь* - содержит информацию о пользователе, от имени которого выполняется программа в системе (ОС).
  - *Операционная система* - содержит информацию о имени и версии ОС, ядре ОС, на которой исполняется программа.
  - *Частота (МГц)* - содержит информацию о частоте центрального процессора, которым исполняется программа. Значение частоты проверяется раз в 10 секунд и позволяет отслеживать её изменение, например, механизмами управления питанием.
  - *Разрешение часов реального времени (мс)* - содержит информацию о возможности или разрешении часов реального времени ОС. Позволяет сориентироваться с минимальным интервалом времени периодических задач, например, для задач сбора данных.
  - *Внутренняя кодировка* - содержит информацию о кодировке, в которой хранятся текстовые сообщения внутри программы.
  - *Конфигурационный файл* - содержит информацию о конфигурационном файле, используемом программой. Устанавливается параметром командной строки -Config.
  - *Рабочая директория* - указывает на рабочую директорию станции. Используется в относительной адресации объектов на файловой системе, например, файлов БД. Допускает изменение пользователем для сохранения данных системы в другую БД. При этом значение этого поля не сохраняется в БД, а может быть изменено только в секции "WorkDB" конфигурационного файла.
  - *Директория иконок* - указывает на директорию, содержащую иконки программы. Если в дереве навигации конфигуратора отсутствуют иконки, то Вы неправильно указали значение этого поля.
  - *Директория модулей* - указывает на директорию модулей для OpenSCADA. Если значение этого поля некорректно, то при запуске Вы не увидите никакого графического интерфейса, а только информацию в консоли о корректном запуске ядра OpenSCADA.
  - *Рабочая БД* - указывает на рабочую базу данных (БД), а именно на БД, используемую для хранения основных данных программы. Изменение этого поля отмечает все объекты программы как модифицированные, что позволяет сохранить или загрузить данные станции из указанной основной БД.
  - *Сохранять систему при выходе* - указывает на необходимость сохранения изменённых данных при завершении работы.
  - *Период сохранения системы* - указывает на периодичность в секундах, с которой сохранять изменённые данные станции.
  - *Язык* - указывает на язык сообщений программы. Изменение этого поля допустимо, однако приводит к изменению языка сообщений только для интерфейса и

динамических сообщений!

- *Базовый язык текстовых переменных* - используется для включения режима поддержки многоязыковых текстовых переменных путём указания непустого базового языка. Значение базового языка выбирается из списка двухсимвольных кодов языков, обычно только текущий и базовый языки в списке. Далее для текстовых переменных на небазовом языке в таблицах БД будут создаваться отдельные колонки. Под текстовыми переменными подразумеваются все текстовые поля конфигуратора, которые могут быть переведены на другой язык. Числа и другие символьные значения к их числу не относятся и не переводятся.
- *Сообщения:* - раздел группы параметров, управляющих работой с сообщениями станции:
  - *Наименьший уровень:* - указывает на уровень сообщений, начиная с которого необходимо их обрабатывать. Сообщения ниже этого уровня будут игнорироваться. Необходимо, например, для исключения из обработки отладочных сообщений уровня 0.
  - *В системный логер(syslog)* - указывает на необходимость направления сообщений в системный логер, механизм ОС для работы с сообщениями системы и ПО. При включении этого параметра появляется возможность управлять и контролировать сообщения OpenSCADA механизмами ОС.
  - *На стандартный выход(stdout)* - указывает на использование в качестве вывода сообщения стандартного механизмы вывода в консоль. Выключение этого свойства исключит весь вывод в консоль, если не указан следующий параметр.
  - *На стандартный выход ошибок(stderr)* - указывает на использование в качестве вывода сообщения стандартного механизмы вывода ошибок, обычно тоже направляется в консоль.
  - *В архив* - указывает на необходимость вывода сообщений в архив сообщений OpenSCADA. Этот параметр обычно включен, а его выключение приводит к фактическому отключению архивирования сообщений на станции.
- Вкладка "Подсистемы" содержит список подсистем (рис.4b) и позволяет выполнять прямые переходы к ним с помощью контекстного меню.
- Вкладка "Задачи" содержит таблицу со списком задач открытых различными компонентами OpenSCADA (рис.4c). Из таблицы можно получит различную информацию о задачах, а также назначить процессора для задач, на многопроцессорных системах.
- Вкладка "Помощь" содержит краткую помощь для данной страницы, рис.4d. В данном случае это доступные параметры командной строки и поля конфигурационного файла для данной страницы.

Для модификации полей этой страницы могут потребоваться права привилегированного пользователя. Получить такие права можно, включив вашего пользователя в группу суперпользователя "root", или, войдя на станцию от имени суперпользователя "root".

Нужно отметить ещё один важный момент: поля идентификаторов всех объектов OpenSCADA недопустимы для прямого редактирования, поскольку являются ключом для хранения данных объектов в БД. Однако поменять идентификатор объекта можно с помощью команды переноса и последующей вставки объекта (Cut->Paste) в конфигураторе.

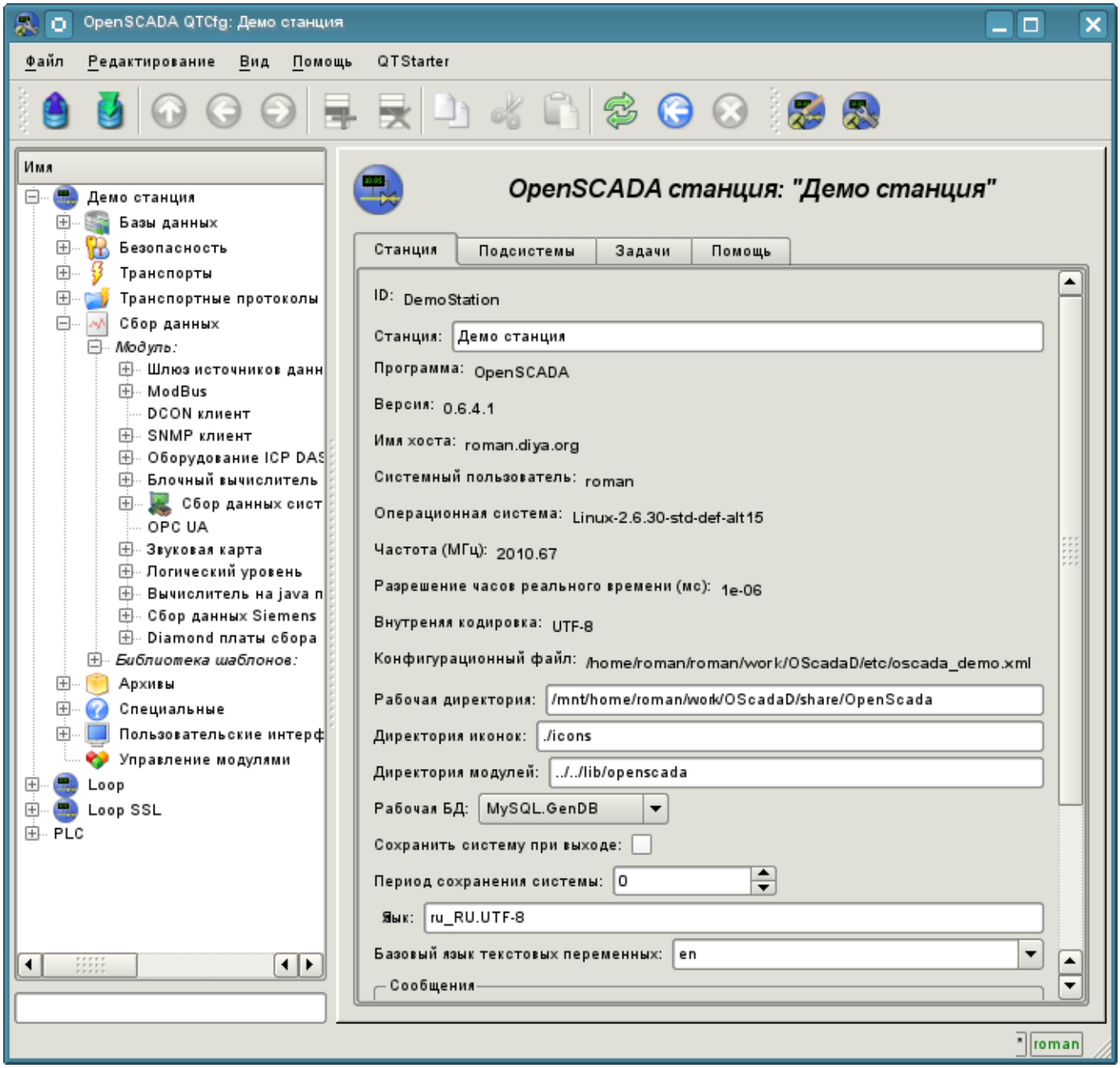


Рис. 4а. Вкладка "Станция" корневой страницы конфигурации системы.

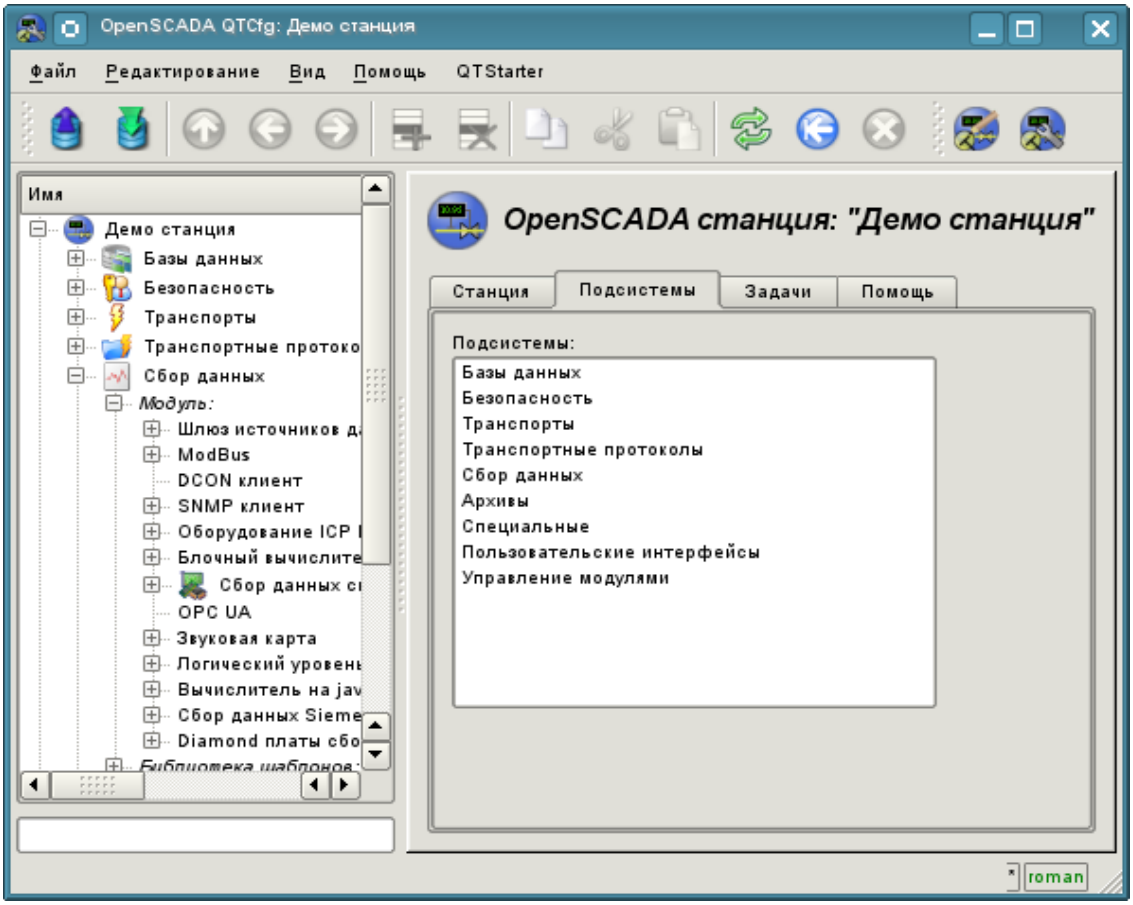
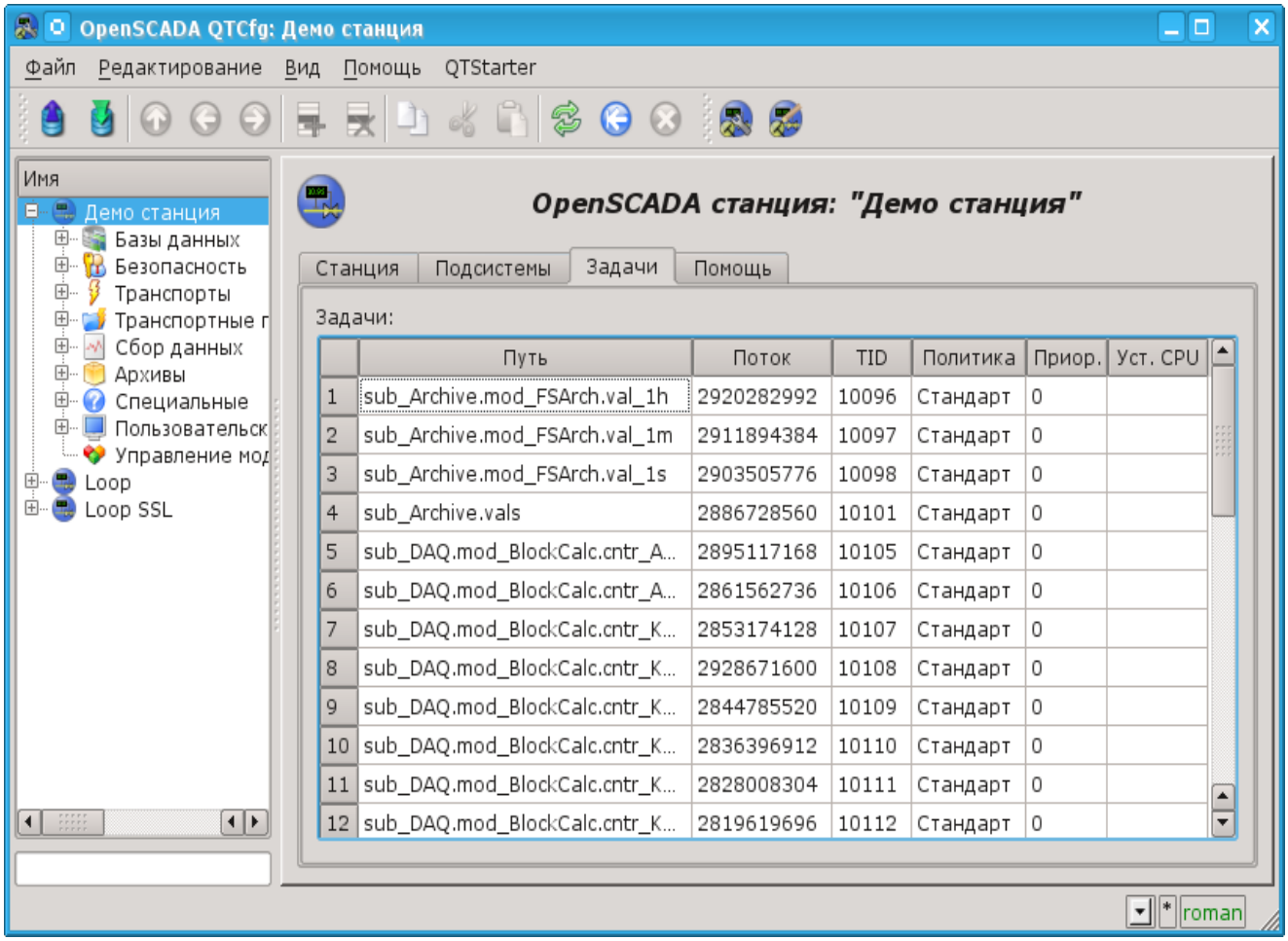


Рис. 4b. Вкладка "Подсистемы" корневой страницы конфигурации системы.



	Путь	Поток	TID	Политика	Приор.	Уст. CPU
1	sub_Archive.mod_FSArch.val_1h	2920282992	10096	Стандарт	0	
2	sub_Archive.mod_FSArch.val_1m	2911894384	10097	Стандарт	0	
3	sub_Archive.mod_FSArch.val_1s	2903505776	10098	Стандарт	0	
4	sub_Archive.vals	2886728560	10101	Стандарт	0	
5	sub_DAQ.mod_BlockCalc.cntr_A...	2895117168	10105	Стандарт	0	
6	sub_DAQ.mod_BlockCalc.cntr_A...	2861562736	10106	Стандарт	0	
7	sub_DAQ.mod_BlockCalc.cntr_K...	2853174128	10107	Стандарт	0	
8	sub_DAQ.mod_BlockCalc.cntr_K...	2928671600	10108	Стандарт	0	
9	sub_DAQ.mod_BlockCalc.cntr_K...	2844785520	10109	Стандарт	0	
10	sub_DAQ.mod_BlockCalc.cntr_K...	2836396912	10110	Стандарт	0	
11	sub_DAQ.mod_BlockCalc.cntr_K...	2828008304	10111	Стандарт	0	
12	sub_DAQ.mod_BlockCalc.cntr_K...	2819619696	10112	Стандарт	0	

Рис. 4c. Вкладка "Задачи" корневой страницы конфигурации системы.

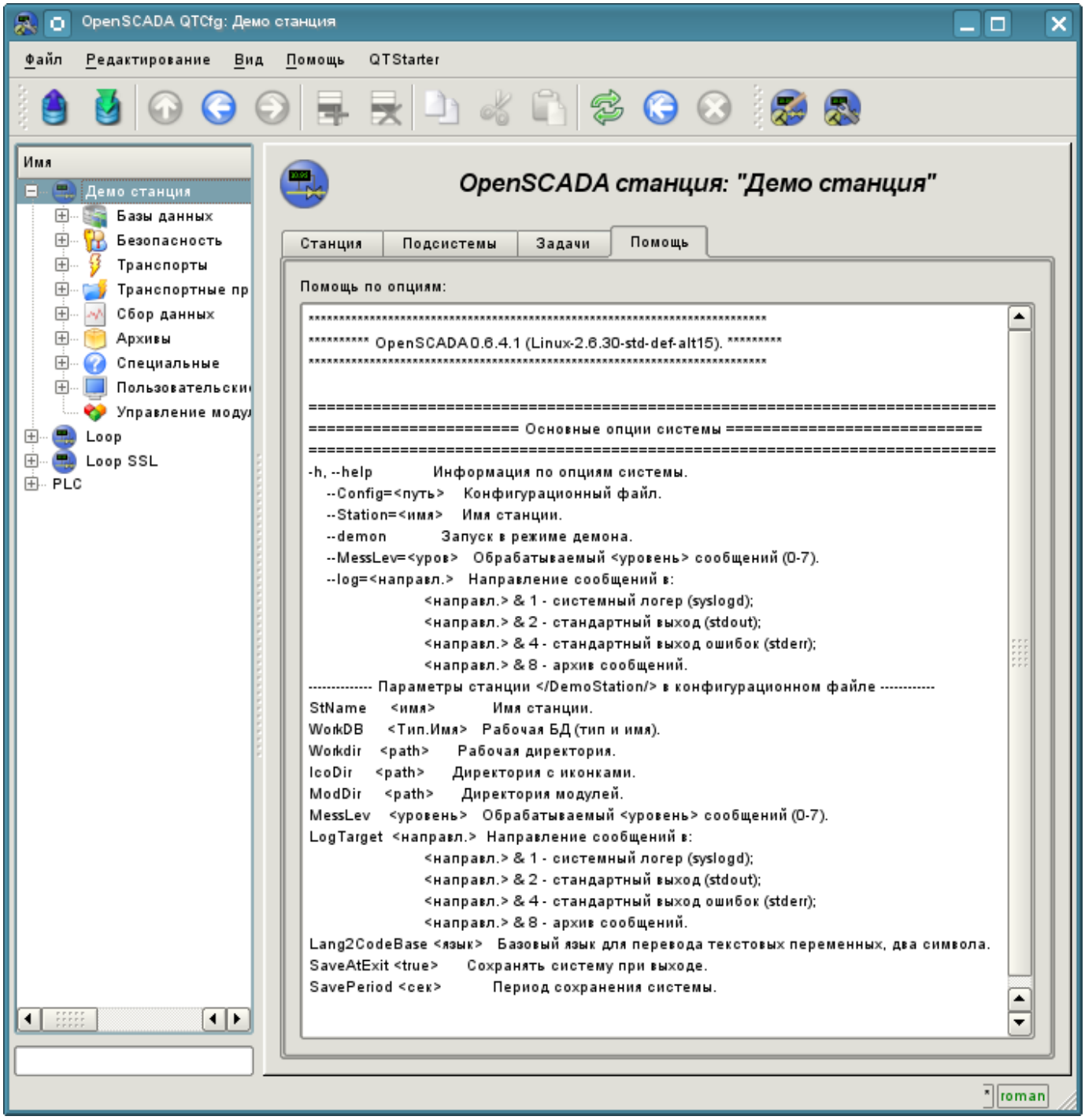


Рис. 4d. Вкладка "Помощь" корневой страницы конфигурации системы.

При рассмотрении страниц конфигурации модульных подсистем будут описаны общие для всех модулей свойства. Однако нужно отметить, что каждый модуль может предоставлять как дополнительные вкладки, так и отдельные поля для конфигурации собственных особенностей функционирования для страниц, объекты которых наследуются модулями. Об особенностях и дополнениях модулей можно ознакомиться в отдельной документации на них.

4.1. Подсистема "БД"

Подсистема является модульной и содержит иерархию объектов изображённую на рис.4.1a. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "БД", содержащая вкладки "Модули" и "Помощь". Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы "БД", доступных на станции. Вкладка "Помощь" содержит краткую помощь для данной страницы.

Для модификации полей страниц этой подсистемы могут потребоваться права привилегированного пользователя или включение вашего пользователя в группу "БД".

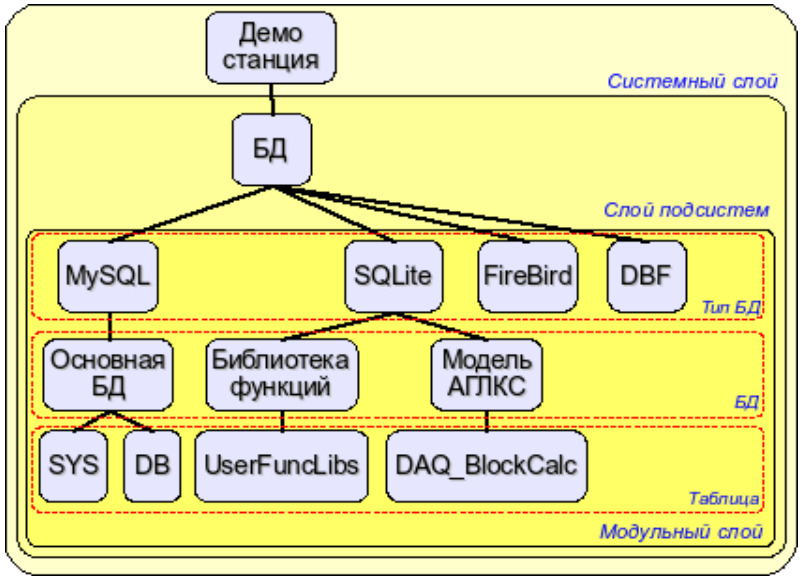


Рис. 4.1a. Иерархическая структура подсистемы "БД".

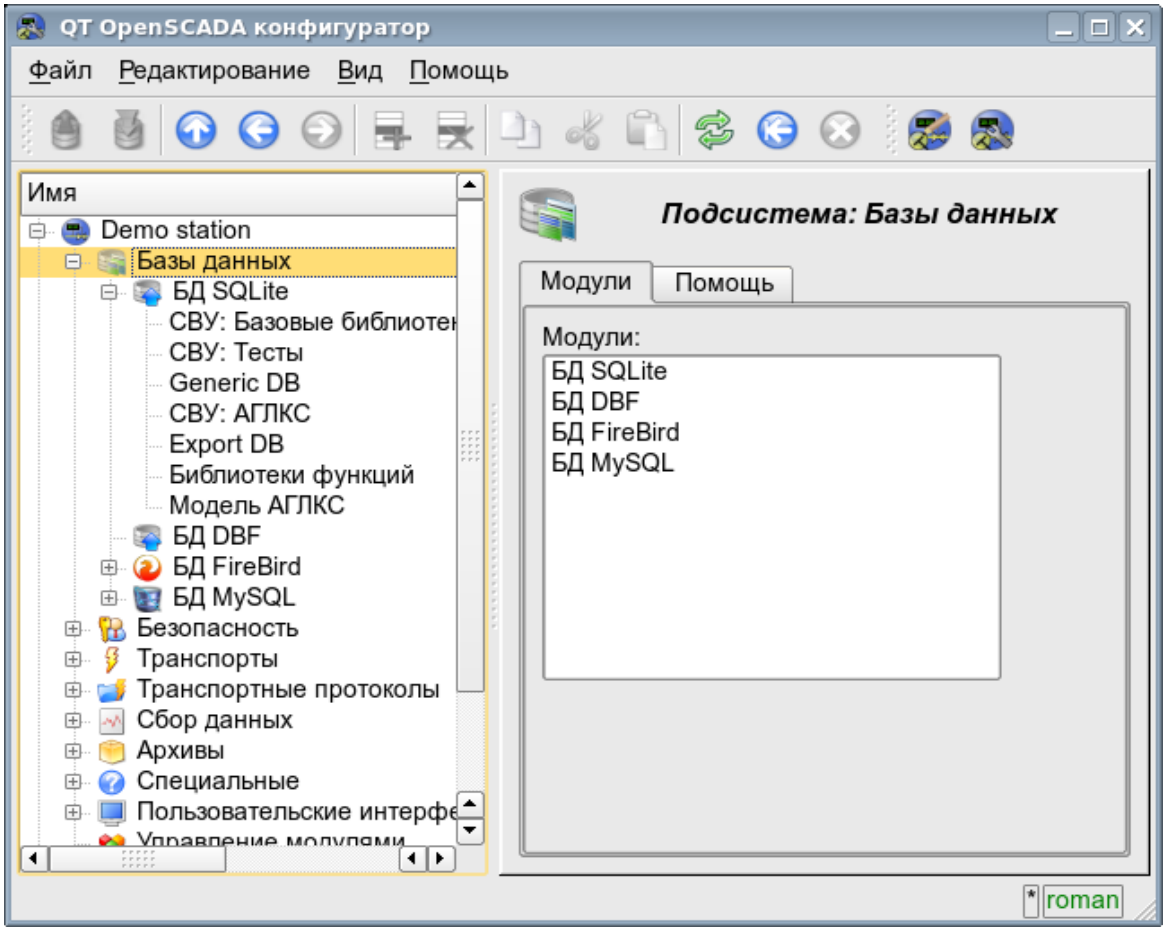


Рис. 4.1b. Вкладка "Модули" корневой страницы подсистемы "БД".

Каждый модуль подсистемы "БД" предоставляет конфигурационную страницу с вкладкам "БД" и "Помощь". Вкладка "БД" (рис.4.1c) содержит список БД, зарегистрированных в модуле и флажок признака полного удаления БД при выполнении команды удаления. В контекстном меню списка БД предоставляется пользователю возможность добавления, удаления и перехода к нужной БД. Во вкладке "Помощь" содержится информация о модуле подсистемы "БД" (рис.4.1d):

- *Модуль* - идентификатор модуля.
- *Имя* - имя модуля.
- *Тип* - тип модуля, идентификатор подсистемы, к которой модуль принадлежит.
- *Источник* - разделяемая библиотека - источник данного модуля.
- *Версия* - версия модуля.
- *Автор* - автор модуля.
- *Описание* - краткое описание модуля.
- *Лицензия* - лицензионное соглашение распространения модуля.

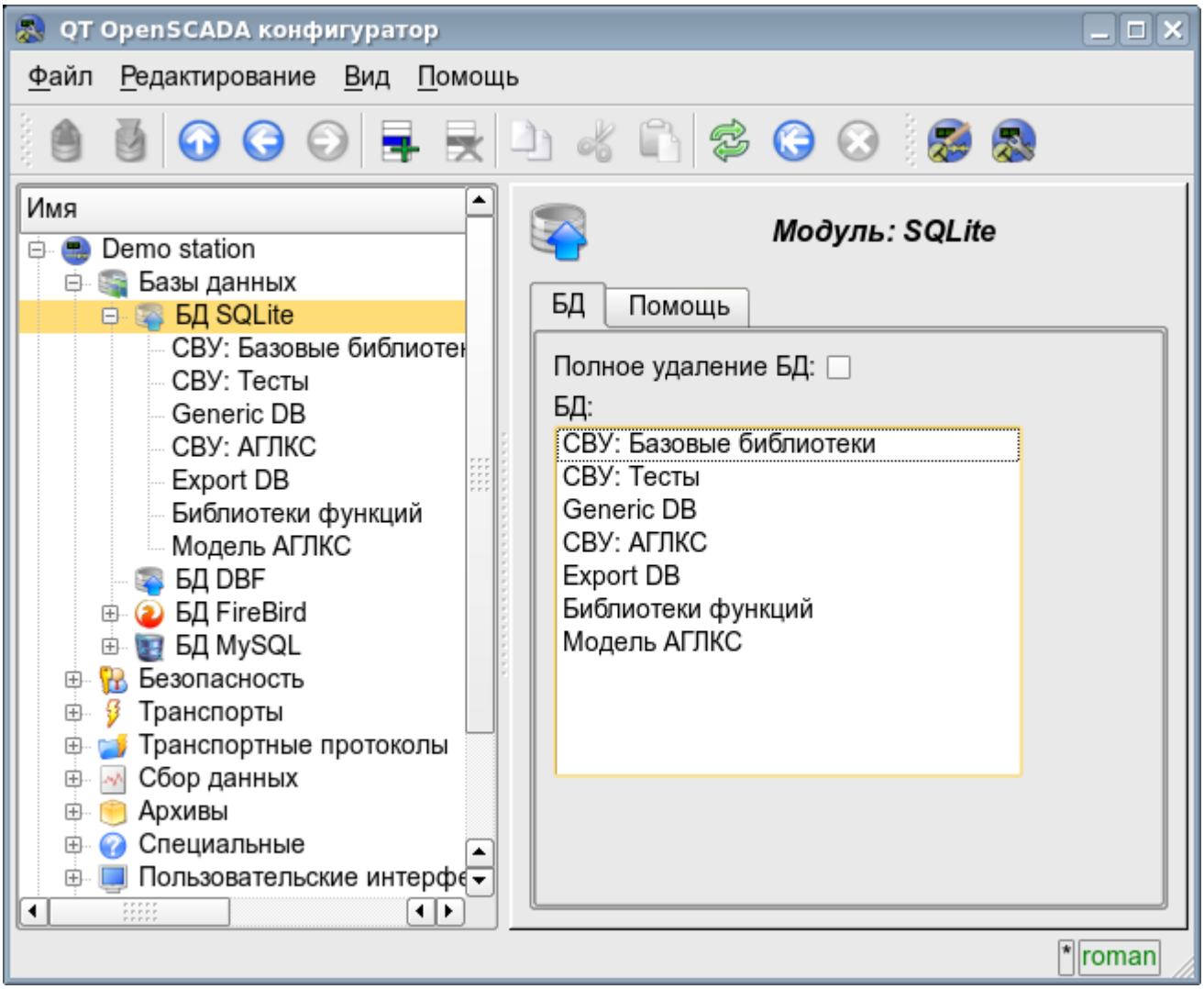


Рис. 4.1c. Вкладка "БД" модуля подсистемы "БД".

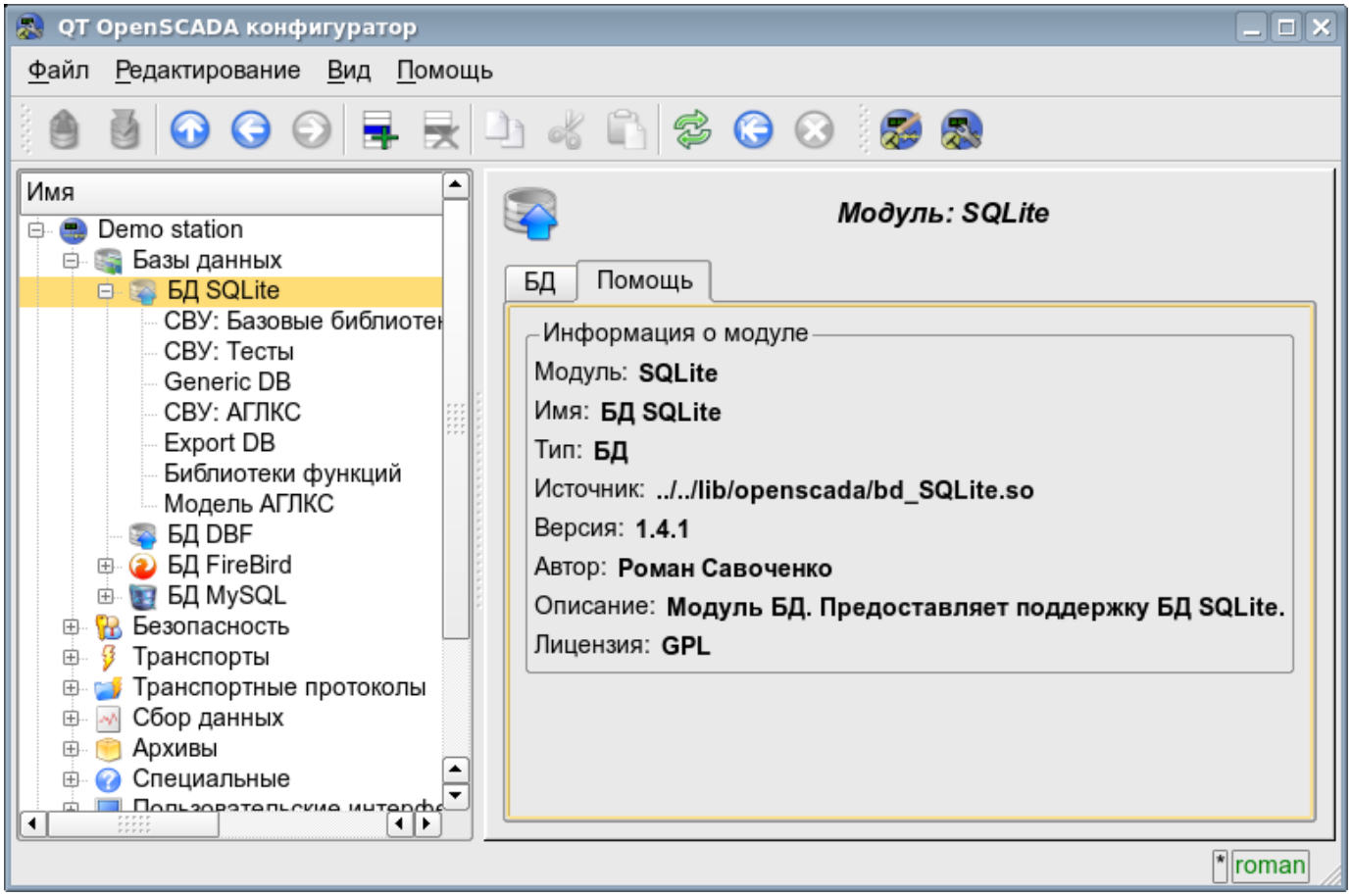


Рис. 4.1d. Вкладка "Помощь" модуля подсистемы "БД".

Каждая БД содержит собственную страницу конфигурации с вкладками "База данных" и "Таблицы". Кроме основных операций можно выполнять копирование содержимого БД стандартной функцией копирования объектов в конфигураторе. Операция копирования содержимого БД подразумевает копирование исходной БД в БД назначения, при этом содержимое БД назначения не очищается перед операцией копирования. Копирование содержимого БД производится при условии включения обоих БД, иначе будет выполняться простое копирование объекта БД.

Вкладка "База данных" (рис.4.1e) содержит основные настройки БД в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние БД:
  - *Включен* - состояние БД "Включен".
  - *Доступные БД* - перечень таблиц, которые содержит БД. Контекстным меню данного свойства предоставляется возможность физического удаления таблиц из БД.
  - *Загрузить систему из БД* - команда для выполнения загрузки из данной БД. Может использоваться при переносе данных в БД между станциями. Например, можно сохранить участок одной станции в экспортную БД, физически перенести БД на другую станцию, подключить её в этой подсистеме и вызвать данную команду.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - содержит информацию об идентификаторе БД.
  - *Имя* - указывает имя БД.
  - *Описание* - краткое описание БД и её назначения.
  - *Адрес* - адрес БД в специфичном для типа БД (модуля) в формате. Описание формата записи адреса БД, как правило, доступно во всплывающей подсказке этого поля.
  - *Кодовая страница* - указывает на кодовую страницу, в которой хранятся и предоставляются текстовые значения БД. Значение кодовой страницы БД в связке с внутренней кодировкой станции используется для прозрачного перекодирования текстовых сообщений при обмене между станцией и БД.
  - *Включать* - указывает на состояние "Включен", в которое переводить БД при загрузке.

Вкладка "Таблицы" (рис.4.1f) содержит список открытых таблиц. При нормальной работе программы эта вкладка пуста, поскольку после завершения работы с таблицами программа их закрывает. Наличие открытых таблиц говорит о том, что программа сейчас с таблицами работает или таблицы открыты пользователем для изучения их содержимого. В контекстном меню перечня открытых таблиц можно открыть таблицу для изучения (команда "Добавить"), закрыть открытую страницу (команда "Удалить") и перейти к изучению содержимого таблицы.

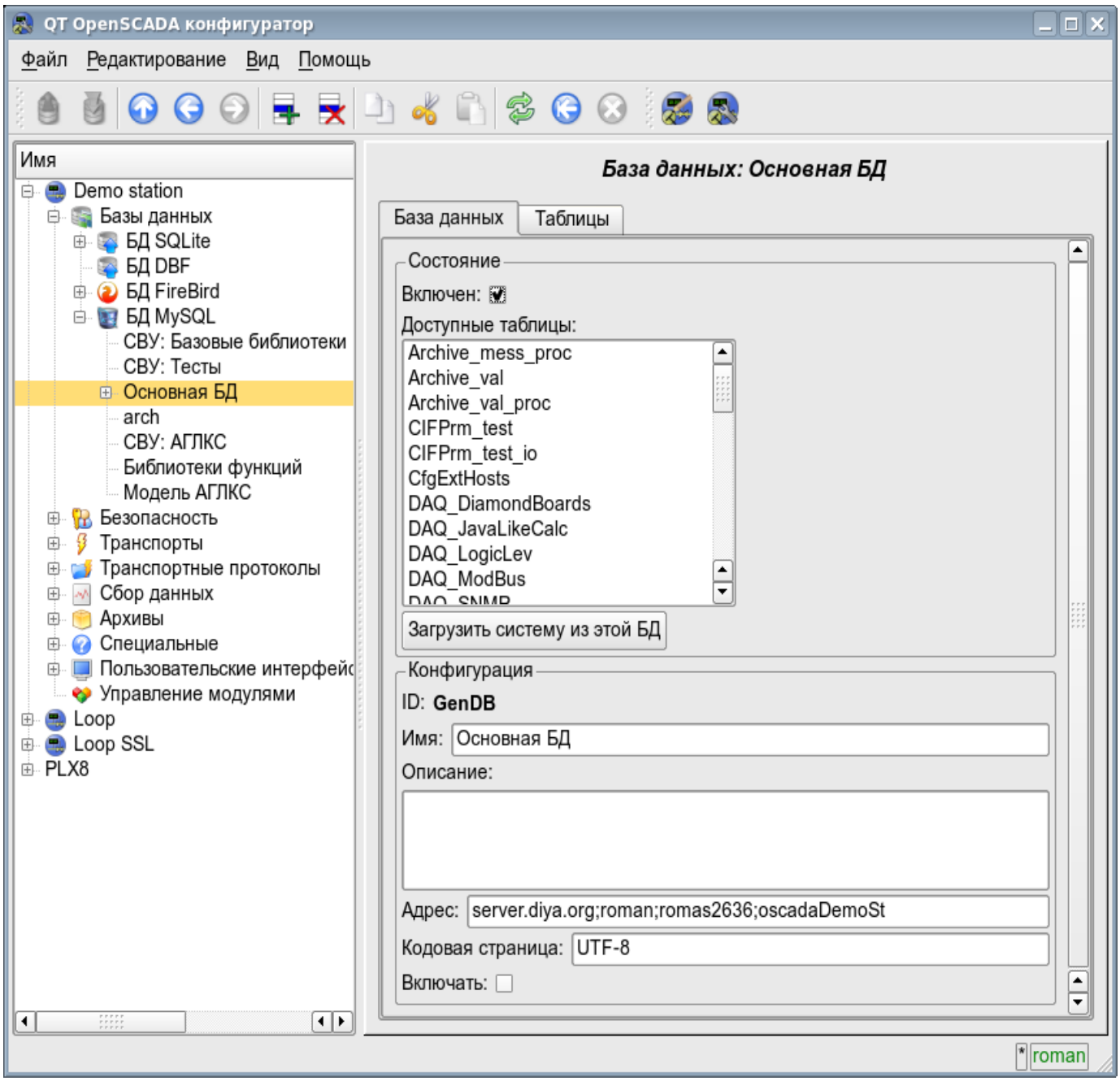


Рис. 4.1e. Вкладка "База данных" БД модуля подсистемы "БД".

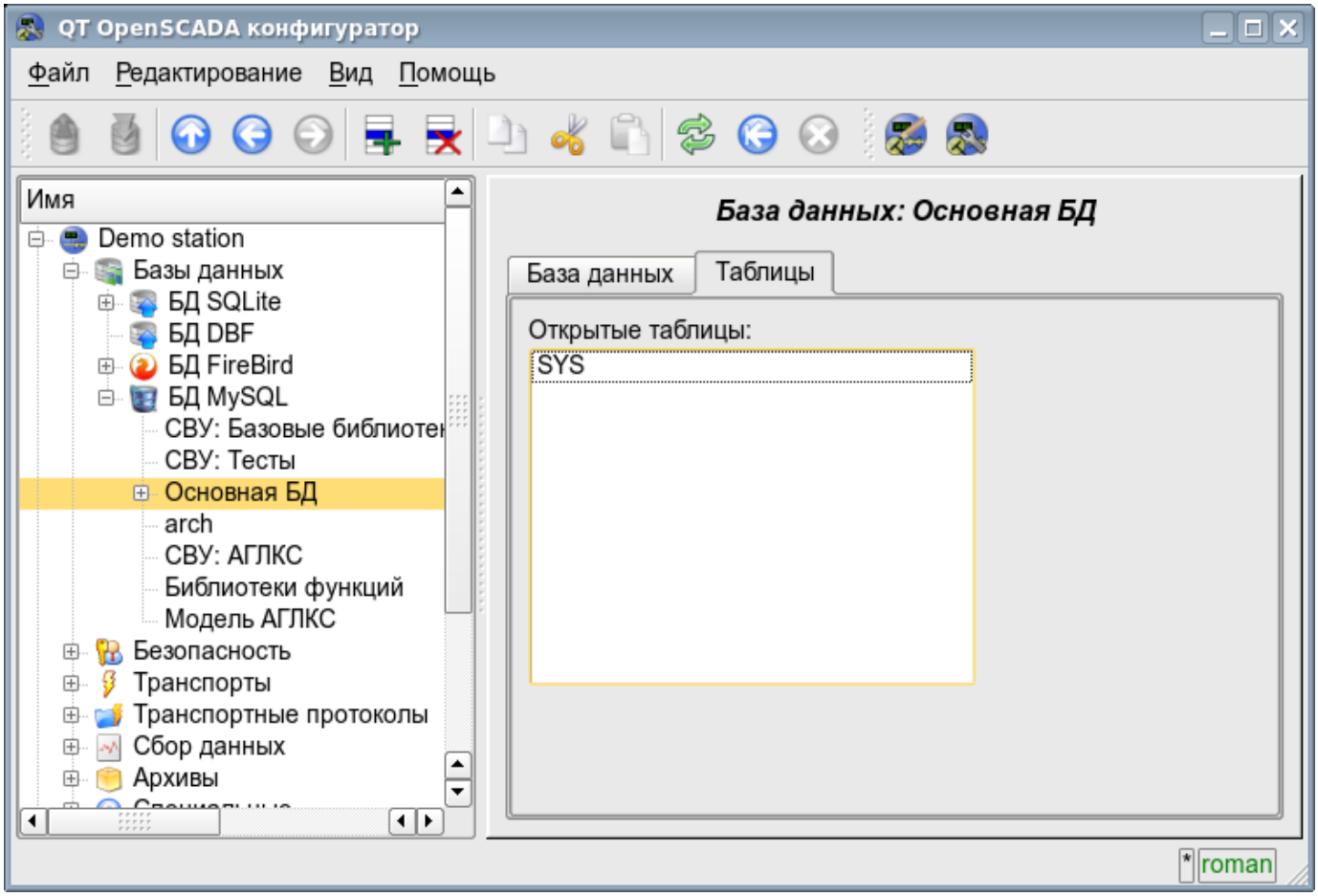


Рис. 4.1f. Вкладка "Таблицы" БД модуля подсистемы "БД".

Страница изучения содержимого таблицы содержит только одну вкладку "Таблица". Вкладка "Таблица" (рис.4.1g) содержит поле имени таблицы и таблицу с содержимым. Таблицей содержимого предоставляются функции:

- редактирование содержимого ячеек таблицы;
- добавление записи (строки);
- удаление записи (строки).

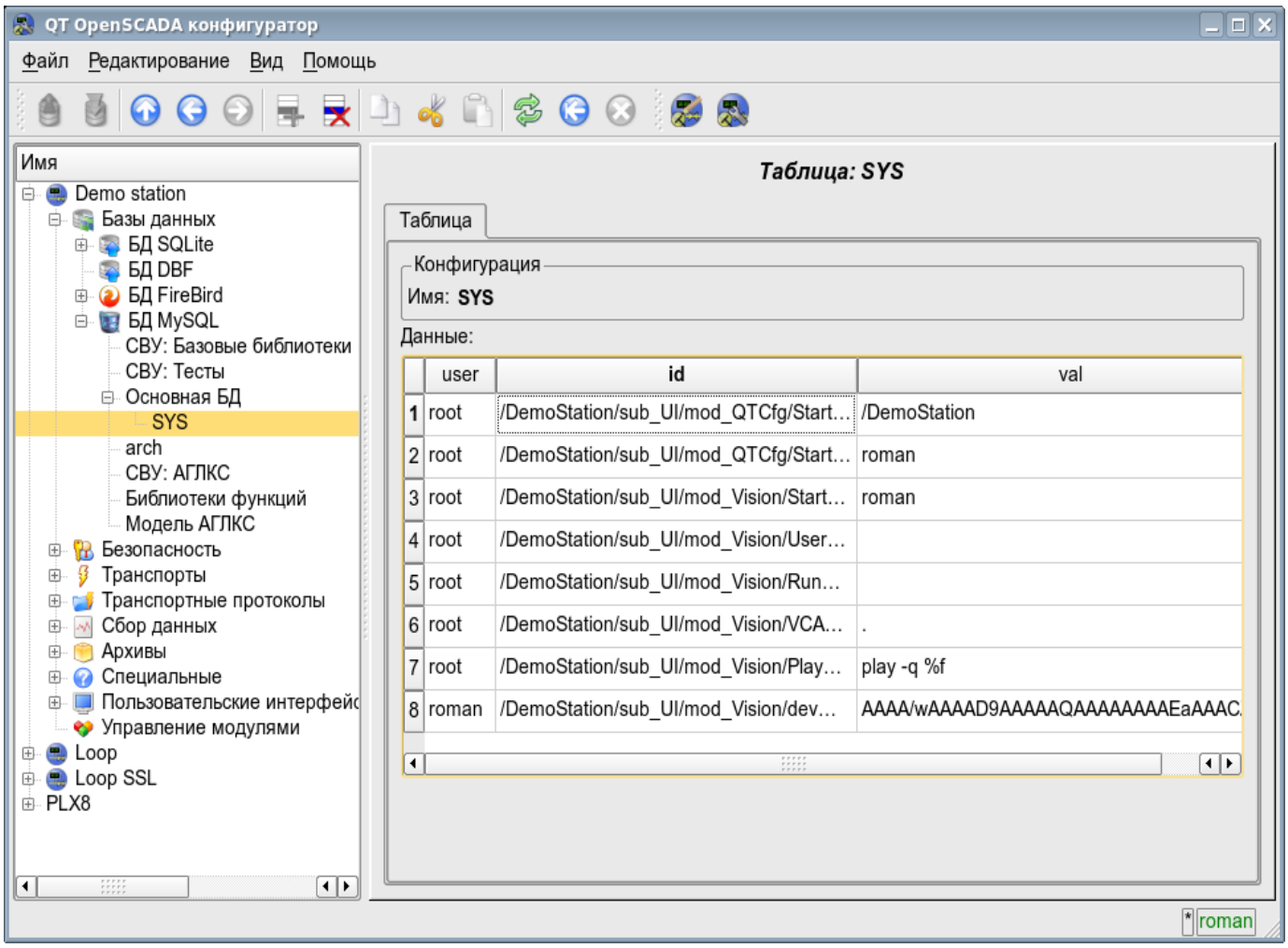


Рис. 4.1g. Вкладка "Таблица" таблицы БД модуля подсистемы "БД".

4.2. Подсистема "Безопасность"

Подсистема не является модульной. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Безопасность", содержащая вкладки "Пользователи и группы пользователей" и "Помощь". Вкладка "Пользователи и группы пользователей" (рис.4.2а) содержит списки пользователей и групп пользователей. Пользователь в группе "Security" и с правами привилегированного пользователя может добавить, удалить пользователя или группу пользователей. Все остальные пользователи могут перейти к странице пользователя или группы пользователя. Вкладка "Помощь" содержит краткую помощь для данной страницы.

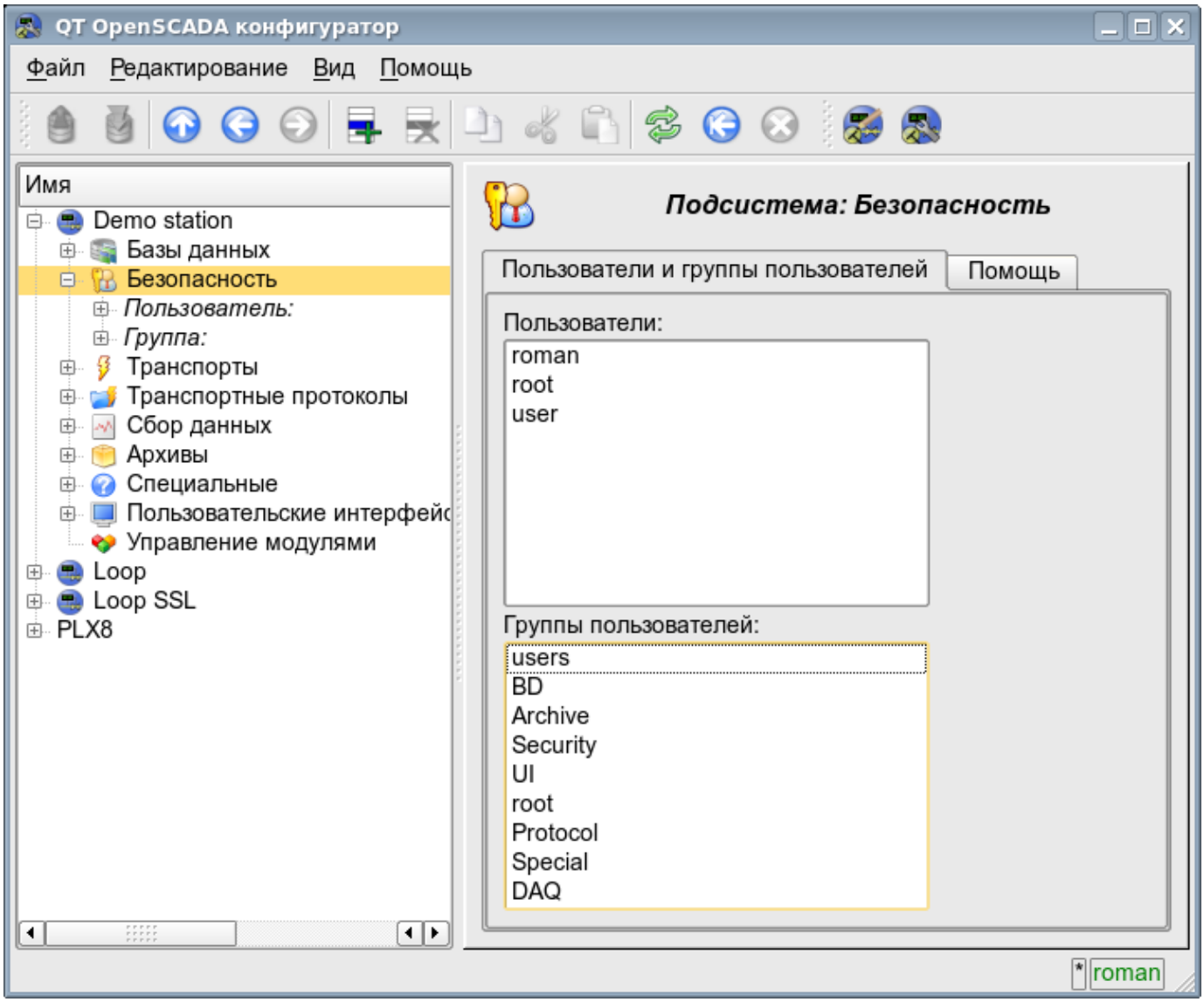


Рис. 4.2а. Вкладка "Пользователи и группы пользователей" корневой страницы подсистемы "Безопасность".

Для конфигурации пользователя предоставляется страница, содержащая только вкладку "Пользователь" (рис.4.2b). Вкладка содержит конфигурационные данные профиля пользователя, которые может изменять сам пользователь, пользователь в группе "Security" или привилегированный пользователь:

- *Имя* - информация об имени (идентификаторе) пользователя.
- *Полное имя* - указывает на полное имя пользователя.
- *Изображение пользователя* - указывает изображение пользователя. Изображение может быть загружено или выгружено.
- *БД пользователя* - адрес БД для хранения данных пользователя.
- *Пароль* - поле для изменения пароля пользователя. Всегда отображает "******".
- *Группы пользователей* - таблица с перечнем групп пользователей станции и признаком принадлежности пользователя к группам.

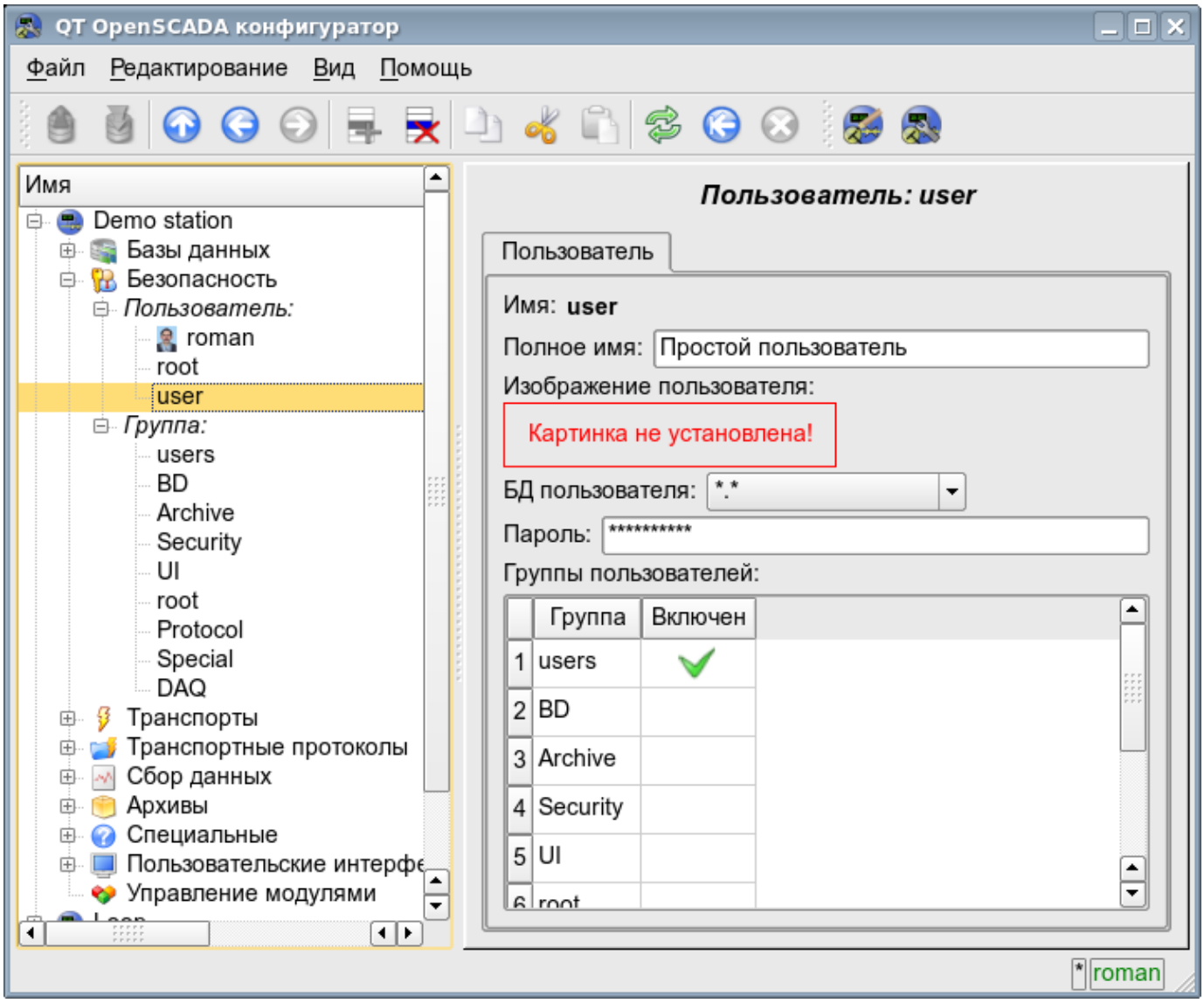


Рис. 4.2b. Вкладка "Пользователь" страницы пользователя подсистемы "Безопасность".

Для конфигурации группы пользователей предоставляется страница, содержащая только вкладку "Группа" (рис.4.2c). Вкладка содержит конфигурационные данные профиля группы пользователей, которые может изменять только привилегированный пользователь:

- *Имя* - информация об имени (идентификаторе) группы пользователей.
- *Полное имя* - указывает на полное имя группы пользователей.
- *БД группы пользователей* - адрес БД для хранения данных группы пользователей.
- *Пользователи* - список пользователей, включенных в данную группу. С помощью контекстного меню списка можно добавить или удалить пользователя в группе.

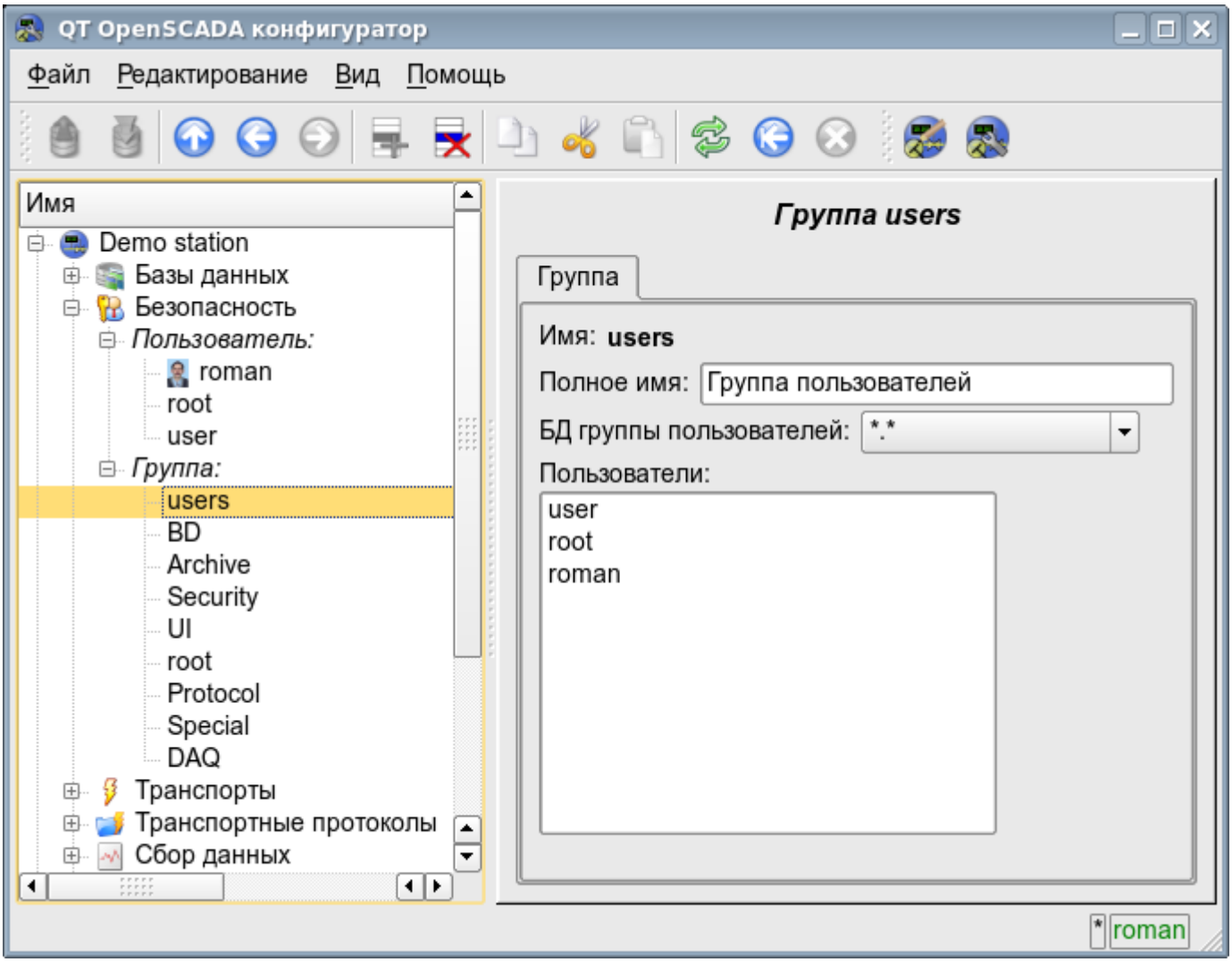


Рис. 4.2c. Вкладка "Группа" страницы группы пользователей подсистемы "Безопасность".

4.3. Подсистема "Транспорты"

Подсистема является модульной и содержит иерархию объектов, изображённую на рис.4.3a. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Транспорты", содержащая вкладки "Подсистема", "Модули" и "Помощь".

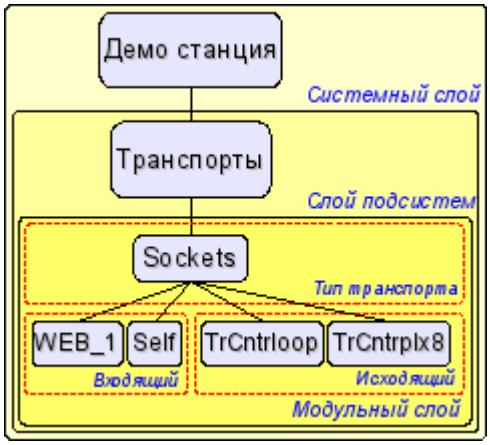


Рис. 4.3a. Иерархическая структура подсистемы "Транспорты".

Вкладка "Подсистема" (рис.4.3b) содержит таблицу конфигурации внешних для данной OpenSCADA станций. Внешние станции могут быть системными и пользовательскими, что выбирается соответствующим параметром. Системные внешние станции доступны только привилегированному пользователю и используются компонентами системного назначения, например, механизмом горизонтального резервирования и модулем DAQ.DAQGate. Пользовательские внешние станции привязаны к пользователю, который их создавал, а значит список пользовательских внешних станций индивидуален для каждого пользователя. Пользовательские внешние станции используются компонентами графического интерфейса, например, UI.QTCfg, UI.WebCfgD и UI.Vision. В таблице внешних станций возможно добавление и удаление записей про станцию, а также их модификация. Каждая запись содержит поля:

- *Id* - идентификатор внешней станции.
- *Имя* - имя внешней станции.
- *Транспорт* - выбор из списка модуля подсистемы "Транспорты" для использования его в доступе к внешней станции.
- *Адрес* - адрес внешней станции в формате, специфичном для выбранного в предыдущем поле модуля подсистемы "Транспорты".
- *Пользователь* - имя/идентификатор пользователя удалённой станции, от имени которого выполнять подключение.
- *Пароль* - пароль пользователя удалённой станции.

Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы "Транспорты" и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

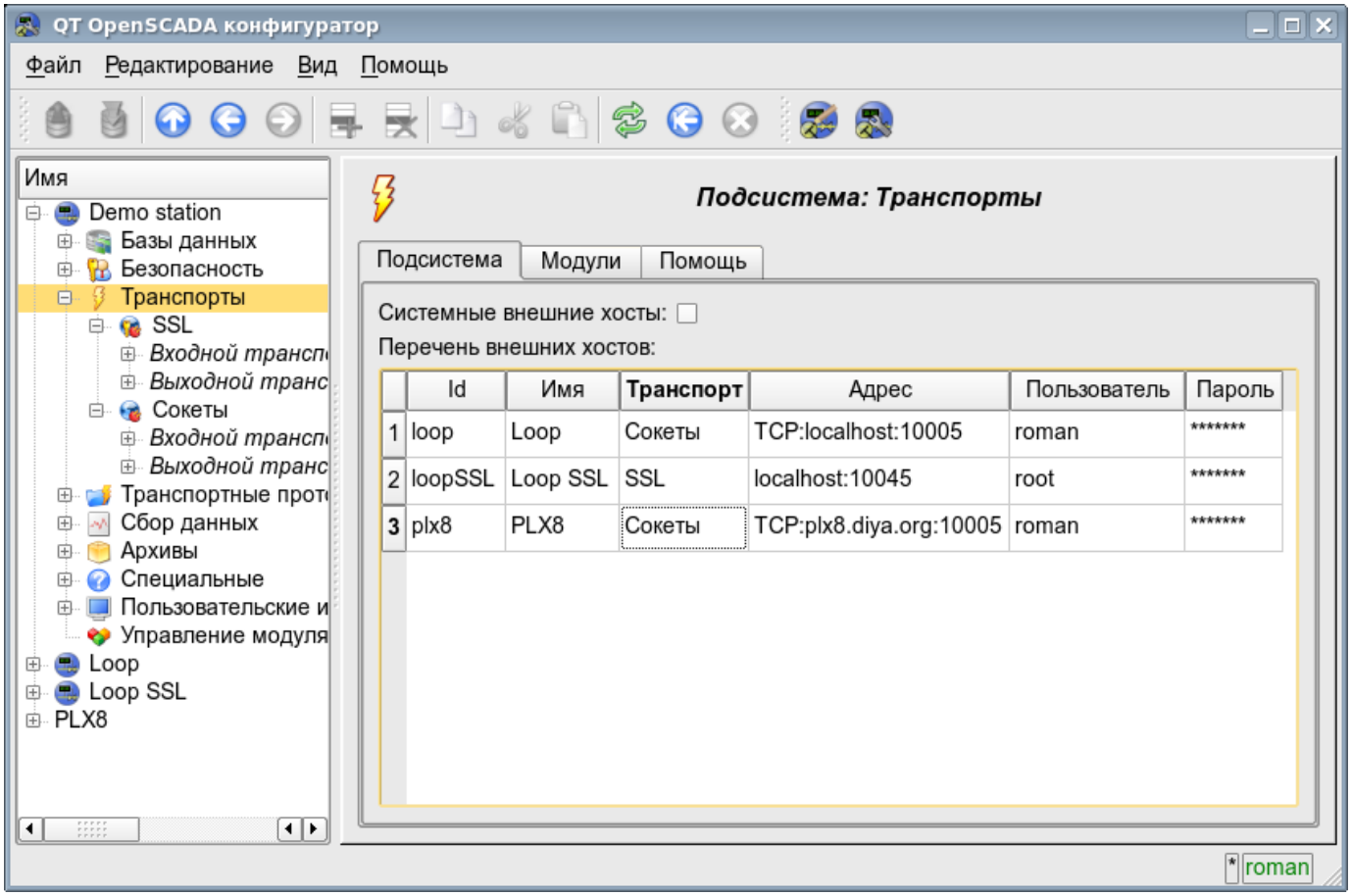


Рис. 4.3b. Вкладка "Подсистема" корневой страницы подсистемы "Транспорты".

Каждый модуль подсистемы "Транспорты" предоставляет конфигурационную страницу с вкладками "Транспорты" и "Помощь". Вкладка "Транспорты" (рис.4.3c) содержит список входящих и исходящих транспортов, зарегистрированных в модуле. В контекстном меню списков транспортов пользователю предоставляется возможность добавления, удаления и перехода к нужному транспорту. Во вкладке "Помощь" содержится информация о модуле подсистемы "Транспорты" (рис.4.1d), состав которой идентичен для всех модулей.

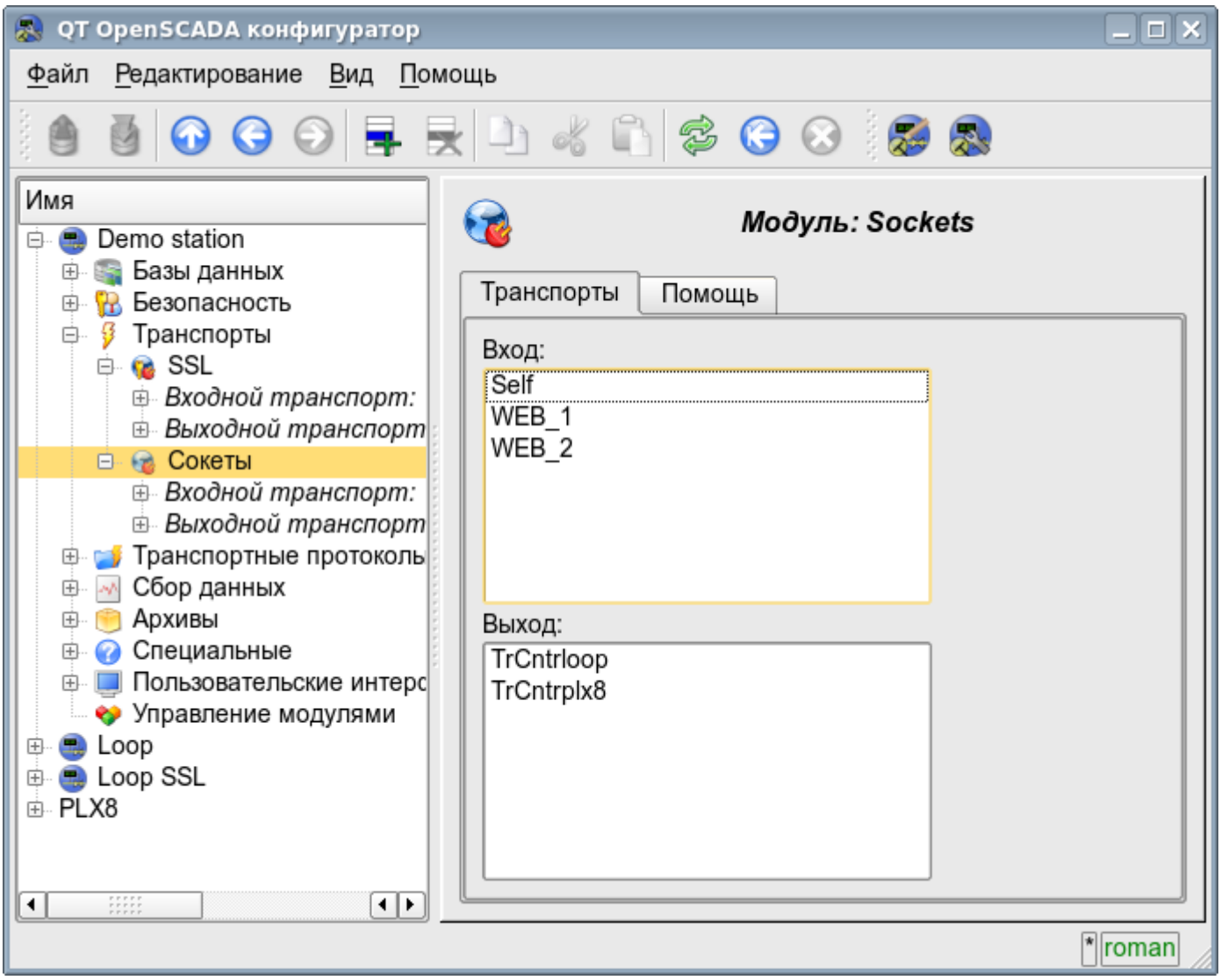


Рис. 4.3c. Вкладка "Транспорты" модуля подсистемы "Транспорты".

Каждый транспорт содержит собственную страницу конфигурации с одной вкладкой "Транспорт". Эта вкладка содержит основные настройки транспорта. Входящий транспорт (рис.4.3d) содержит:

- Раздел "Состояние" - содержит свойства, характеризующие состояние транспорта:
  - *Статус* - информация о текущем состоянии транспорта и статистика его работы.
  - *Выполняется* - состояние транспорта "Выполняется".
  - *БД транспорта* - адрес БД для хранения данных транспорта.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе транспорта.
  - *Имя* - указывает имя транспорта.
  - *Описание* - краткое описание транспорта и его назначения.
  - *Адрес* - адрес транспорта в специфичном для типа транспорта (модуля) формате. Описание формата записи адреса транспорта, как правило, доступно во всплывающей подсказке этого поля.
  - *Транспортный протокол* - указывает на модуль транспортного протокола (подсистема "Транспортные протоколы"), который должен работать в связке с данным входным транспортом. Т.е. полученные неструктурированные данные этот модуль будет направлять на структуризацию и обработку указанному модулю транспортного протокола.
  - *Запускать* - указывает на состояние "Выполняется", в которое переводить транспорт при загрузке.

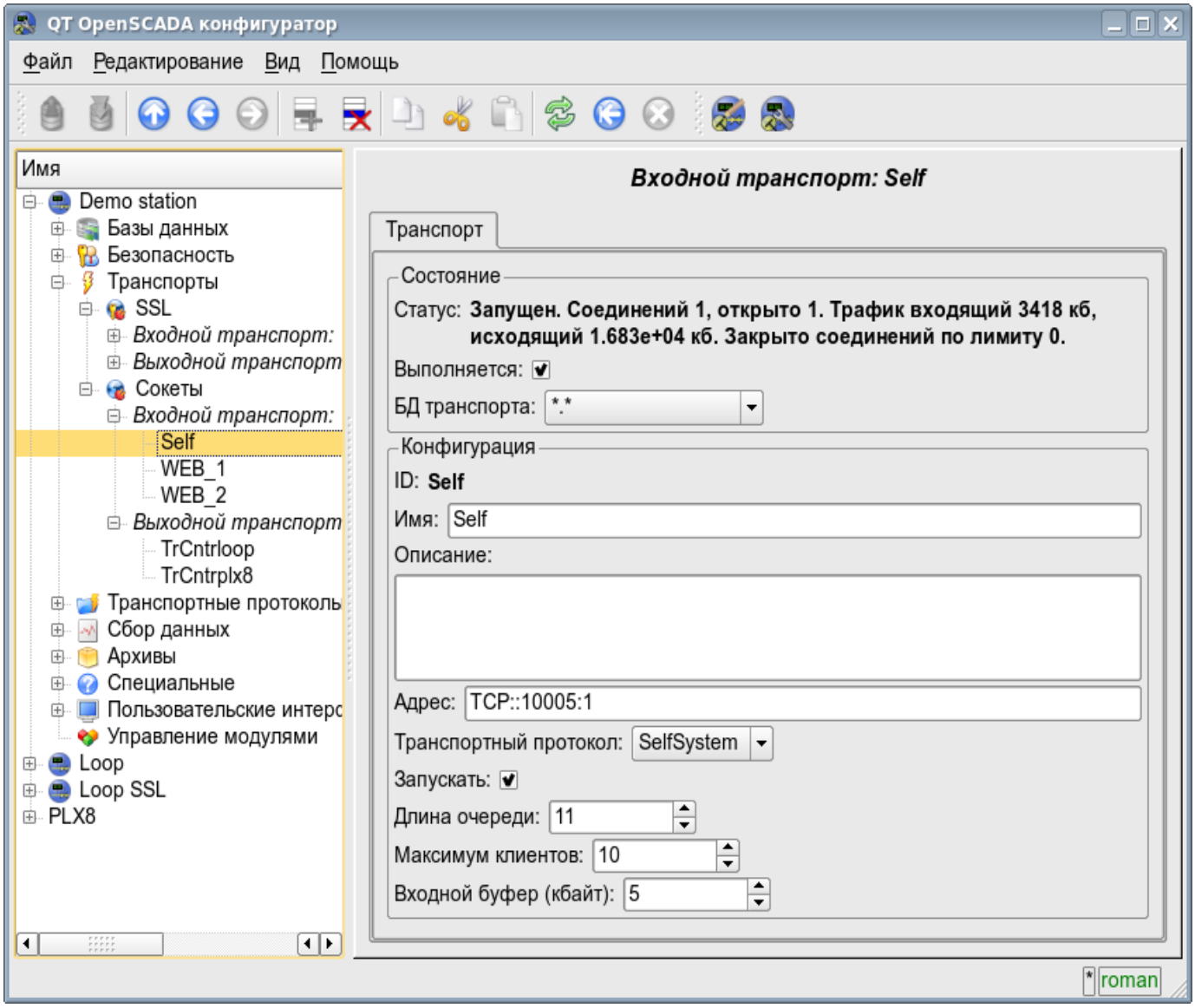


Рис. 4.3d. Вкладка "Транспорт" страницы входящего транспорта модуля подсистемы "Транспорты".

Исходящий транспорт (рис.4.3е) содержит:

- Раздел "Состояние" - содержит свойства, характеризующие состояние транспорта:
  - *Статус* - информация о текущем состоянии транспорта и статистика его работы.
  - *Выполняется* - состояние транспорта "Выполняется".
  - *БД транспорта* - адрес БД для хранения данных транспорта.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе транспорта.
  - *Имя* - указывает имя транспорта.
  - *Описание* - краткое описание транспорта и его назначения.
  - *Адрес* - адрес транспорта в специфичном для типа транспорта (модуля) формате. Описание формата записи адреса транспорта, как правило, доступно во всплывающей подсказке этого поля.
  - *Запускать* - указывает на состояние "Выполняется", в которое переводить транспорт при загрузке.

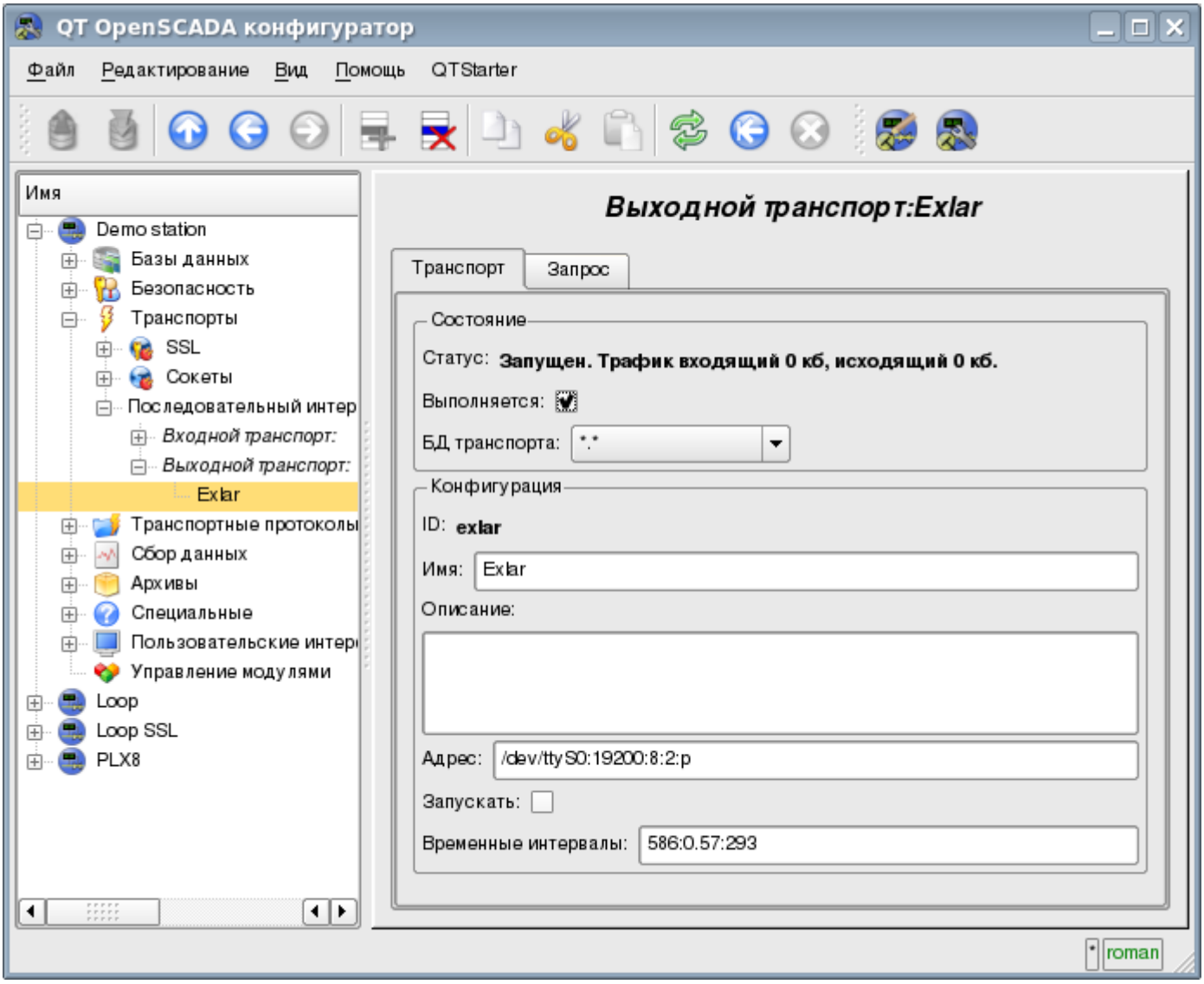


Рис. 4.3е. Вкладка "Транспорт" страницы исходящего транспорта модуля подсистемы "Транспорты".

Исходящий транспорт, в дополнение, предоставляет вкладку формирования пользовательского запроса через данный транспорт (рис.4.3f). Вкладка предназначена для наладки связи, а также для отладки протоколов и содержит:

- *Время (мс)* - информация о времени, затраченном на запрос и получение ответа.
- *Режим* - указывает режим данных, из списка "Текст" и "Бинарный", в котором будет формироваться запрос и предоставляться ответ. В бинарном режиме данные записываются парами чисел в шестнадцатеричном исчислении, т.е байтами, разделёнными пробелами.
- *Отправить* - команда отправить запрос.
- *Запрос* - содержит запрос в выбранном режиме представления данных.
- *Ответ* - предоставляет ответ в выбранном режиме представления данных.

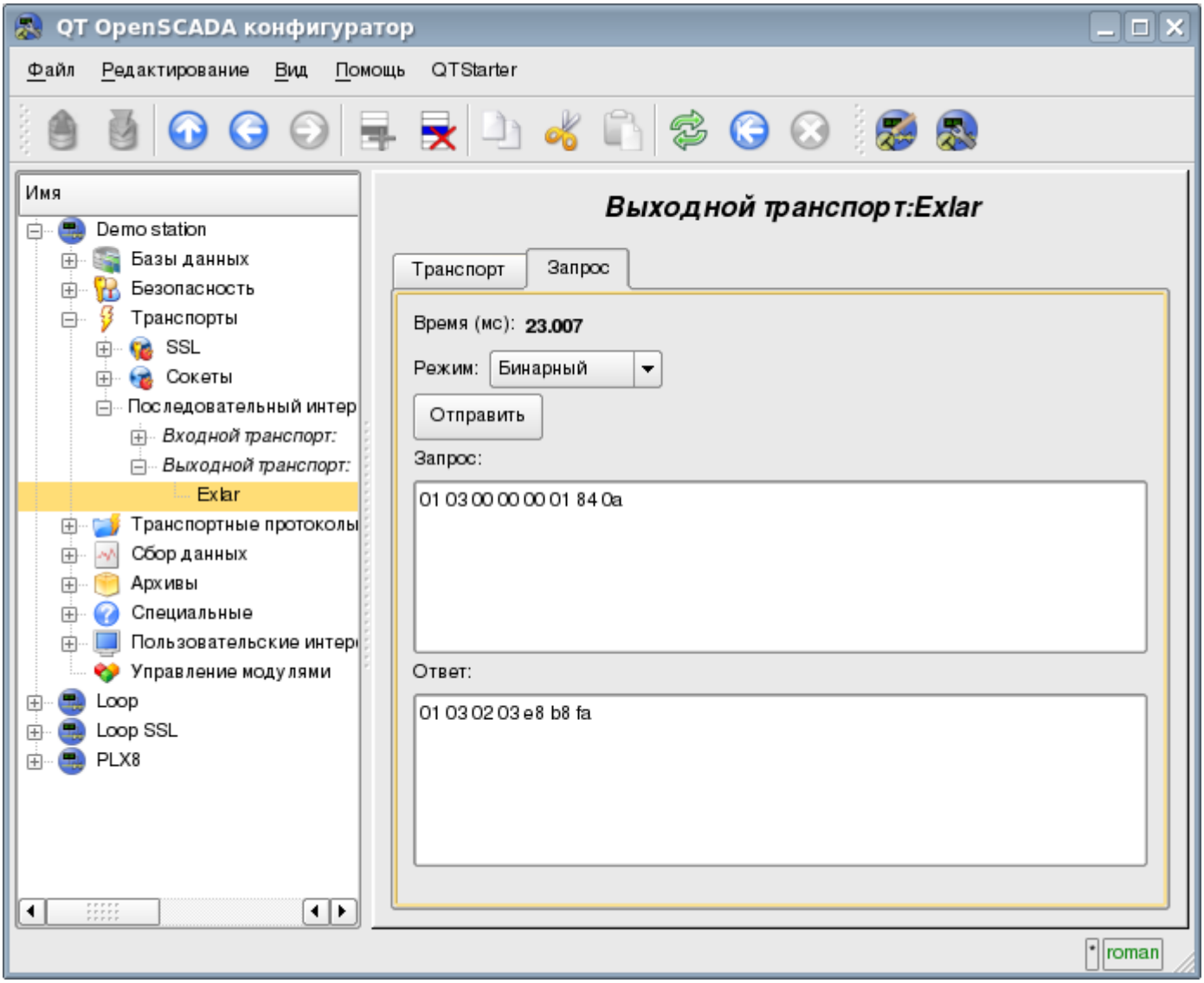


Рис. 4.3f. Вкладка "Запрос" страницы исходящего транспорта модуля подсистемы "Транспорты".

4.4. Подсистема "Транспортные протоколы"

Подсистема является модульной. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Транспортные протоколы", содержащая вкладки "Модули" и "Помощь". Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы "Транспортные протоколы" и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждый модуль подсистемы "Транспортные протоколы" предоставляет конфигурационную страницу с одной вкладкой "Помощь". Во вкладке "Помощь" содержится информация о модуле подсистемы "Транспортные протоколы" (рис.4.1d), состав которой идентичен для всех модулей.

4.5. Подсистема "Сбор данных"

Подсистема является модульной и содержит иерархию объектов, изображённую на рис.4.5a. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Сбор данных", содержащая вкладки "Библиотеки шаблонов", "Модули" и "Помощь".

Для получения доступа на модификацию объектов этой подсистемы необходимы права пользователя в группе "DAQ" или права привилегированного пользователя.

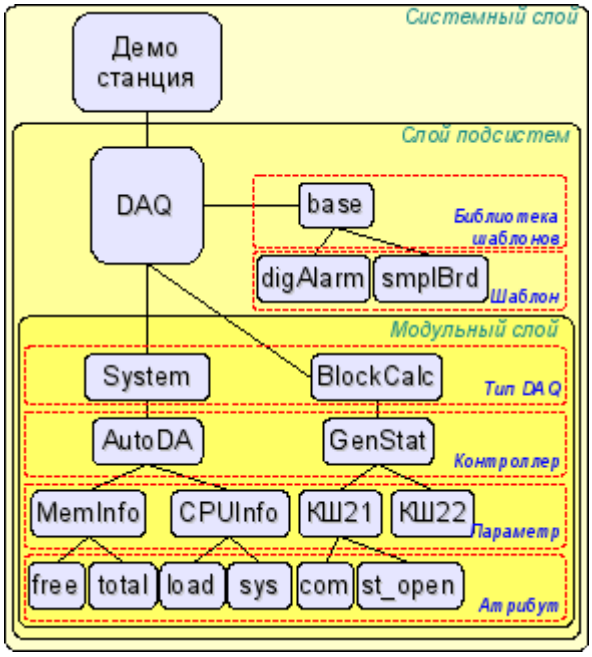


Рис. 4.5a. Иерархическая структура подсистемы "Сбор данных".

Вкладка "Резервирование" (рис.4.5b) содержит конфигурацию резервирования источников данных подсистемы "Сбор данных" станции в составе настроек:

- *Статус* - содержит информацию о работе схемы резервирования, обычно это время, затраченное на исполнения одного цикла задачи обслуживания резерва.
- *Уровень станции* - указывает уровень данной станции в схеме резервирования (0-255).
- *Период задачи резервирования* - указывает периодичность исполнения задачи резервирования в секундах (1-255).
- *Интервал времени восстановления соединения* - указывает через какой интервал времени осуществлять попытку восстановления соединения с потерянной резервной станцией в секундах (0-255).
- *Глубина времени восстановления данных* - указывает на максимальную глубину архивных данных для восстановления из архива удалённой станции при запуске в часах (0-12).
- *Станции* - содержит таблицу с информацией о резервных станциях. Станции можно добавлять и удалять посредством контекстного меню. Идентификатор добавленных станций нужно выбрать из списка доступных системных станций OpenSCADA. Таблица предоставляет следующую информацию о станции:
  - *ID* - идентификатор системной станции OpenSCADA, должен быть изменён после добавления путём выбора из перечня доступных;
  - *Имя* - имя системной станции OpenSCADA;
  - *Жив* - признак наличия связи с резервной станцией;
  - *Уровень* - уровень удалённой станции в схеме резервирования;
  - *Счётчик* - счётчик запросов к резервной станции или времени ожидания в случае отсутствия связи;
  - *Запущен* - список доступных контроллеров с признаком (+) локального исполнения на удалённой станции.
- *Переход к конфигурации перечня удалённых станций* - команда для перехода на страницу конфигурации удалённых OpenSCADA станций, в подсистеме "Транспорты".

- *Контроллеры* - содержит таблицу с перечнем контроллеров, доступных для резервирования, и текущее их состояние:
  - *Контроллер* - полный идентификатор контроллера;
  - *Имя* - имя контроллера;
  - *Запущен* - признак исполнения контроллера локальной станцией;
  - *Резервирование* - режим резервирования контроллера, может быть изменёна из перечня: "Выключен", "Асимметричное" и "Симметричное";
  - *Предпочтение исполнения* - конфигурация предпочтительного исполнения на указанной станции, может быть изменён; зарезервированные значения указывают на: <Высокий уровень> - исполнение на станции с наивысшим уровнем, <Низкий уровень> - исполнение на станции с самым низким уровнем, <Оптимально> - выбор для исполнения наименее нагруженной станции.
  - *Удалённый* - признак, указывающий на исполнение контроллера удалённой станцией и перевод локальной в режим синхронизации данных из удалённой станции.

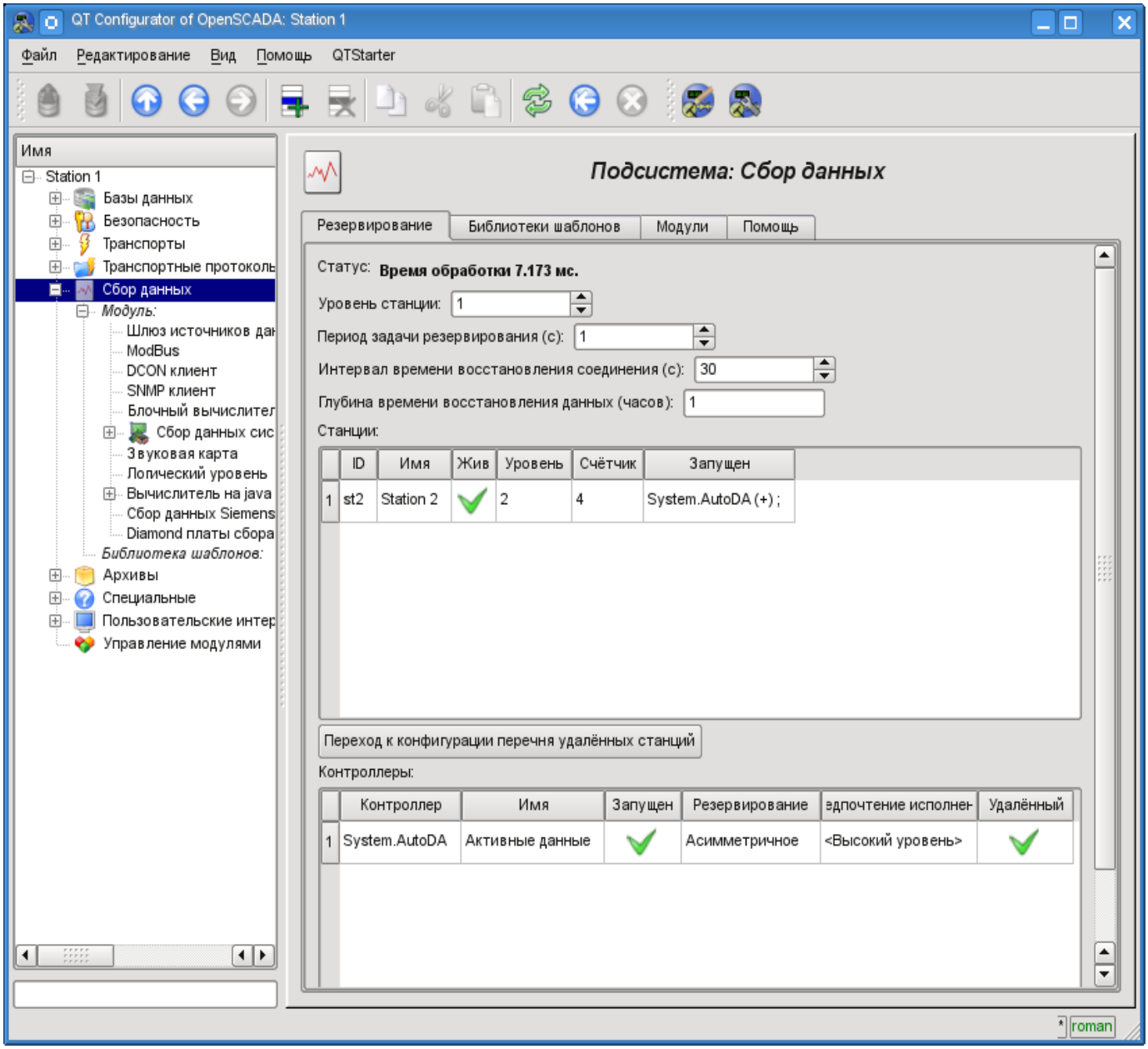


Рис. 4.5b. Вкладка "Резервирование" подсистемы "Сбор данных".

Вкладка "Библиотеки шаблонов" (рис.4.5c) содержит список библиотек шаблонов для параметров этой подсистемы. В контекстном меню списка библиотек шаблонов пользователю предоставляется возможность добавления, удаления и перехода к нужной библиотеке. Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы "Транспорты" и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

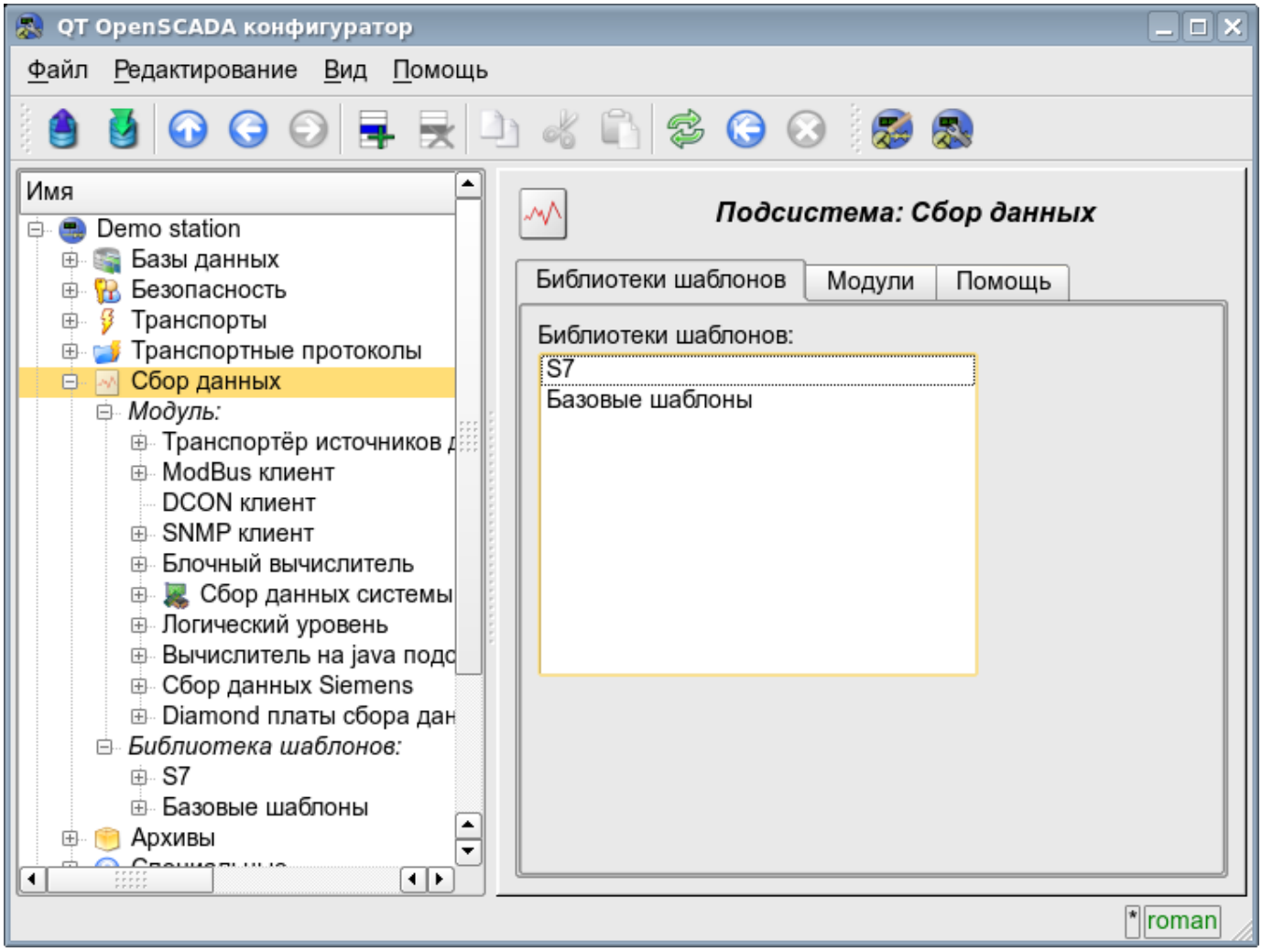


Рис. 4.5c. Вкладка "Библиотеки шаблонов" подсистемы "Сбор данных".

Каждая библиотека шаблонов подсистемы "Сбор данных" предоставляет конфигурационную страницу с вкладками "Библиотека" и "Шаблоны параметров". Вкладка "Библиотека" (рис.4.5d) содержит основные настройки библиотеки в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние библиотеки:
  - *Доступен* - состояние библиотеки "Доступен".
  - *БД библиотеки* - адрес БД для хранения данных библиотеки и шаблонов.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе библиотеки.
  - *Имя* - указывает имя библиотеки.
  - *Описание* - краткое описание библиотеки и её назначения.

Вкладка "Шаблоны параметров" (рис.4.5e) содержит список шаблонов в библиотеке. В контекстном меню списка пользователю предоставляется возможность добавления, удаления и перехода к нужному шаблону.

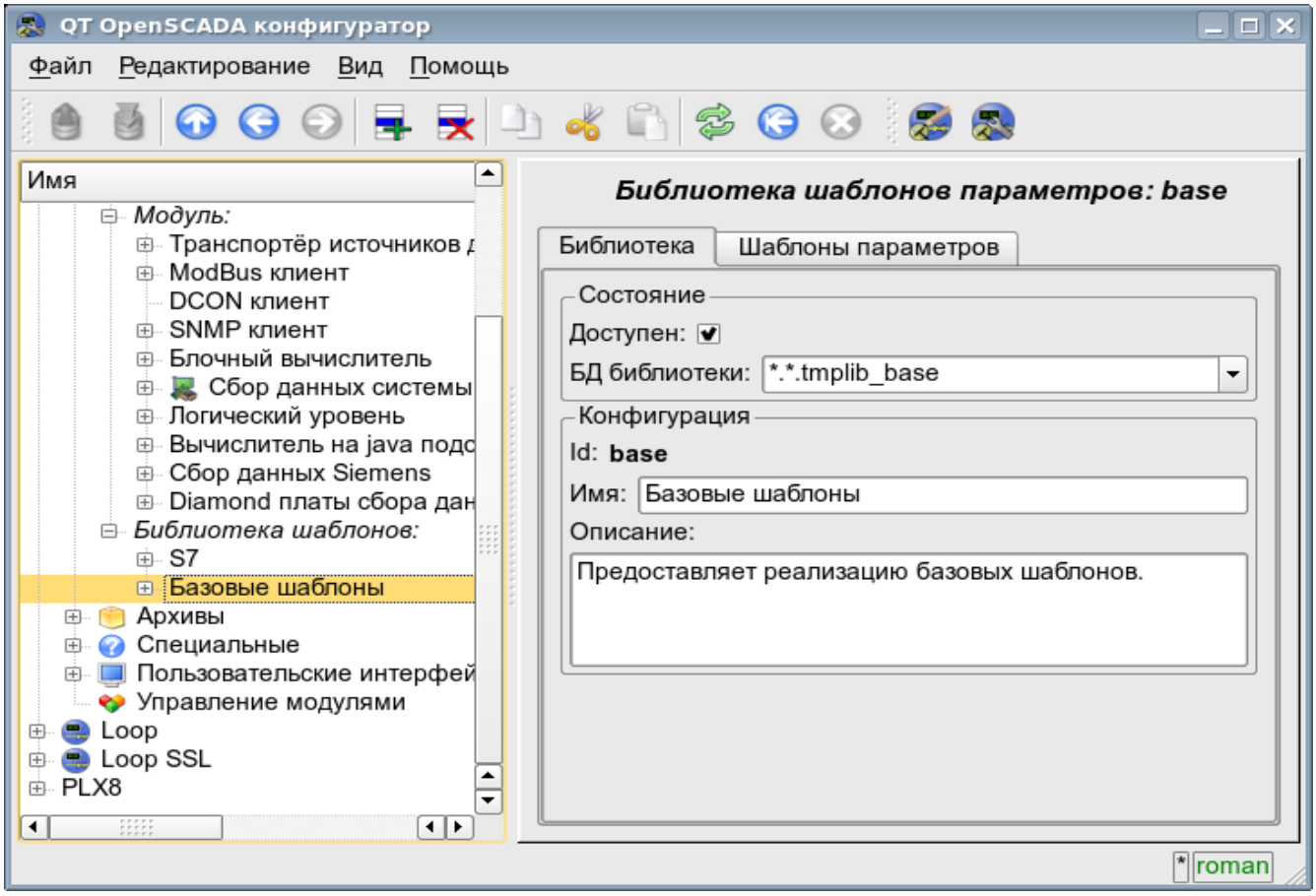


Рис. 4.5d. Основная вкладка конфигурации библиотеки шаблонов подсистемы "Сбор данных".

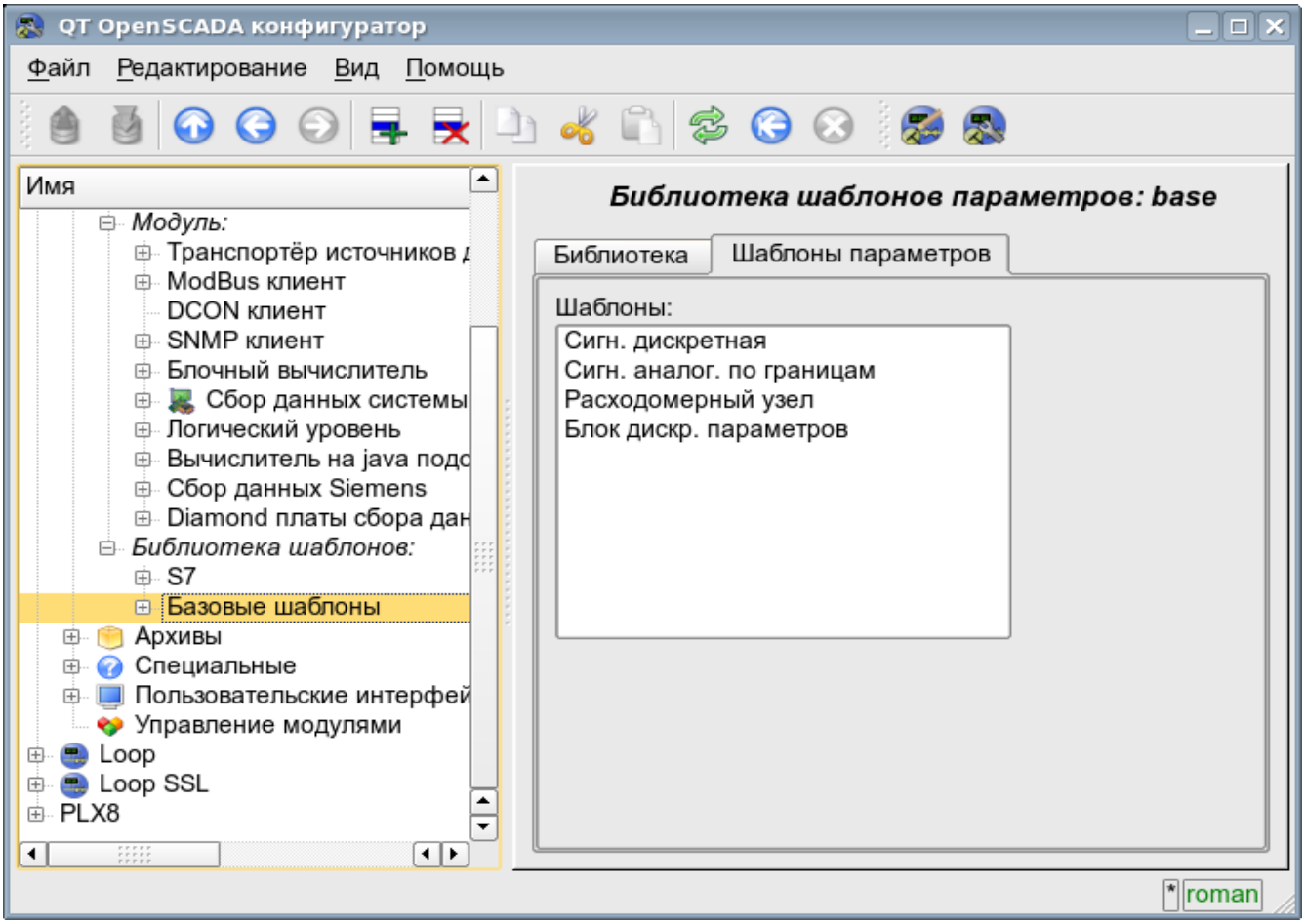


Рис. 4.5e. Вкладка списка шаблонов в библиотеке шаблонов подсистемы "Сбор данных".

Каждый шаблон библиотеки шаблонов предоставляет конфигурационную страницу с вкладками "Шаблон" и "IO". Вкладка "Шаблон" (рис.4.5f) содержит основные настройки шаблона в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние шаблона:
  - *Доступен* - состояние шаблона "Доступен".
  - *Использовано* - количество использования шаблона. Позволяет определить факт использования и, как следствие, возможность редактирования шаблона.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе шаблона.
  - *Имя* - указывает имя шаблона.
  - *Описание* - краткое описание шаблона и его назначения.

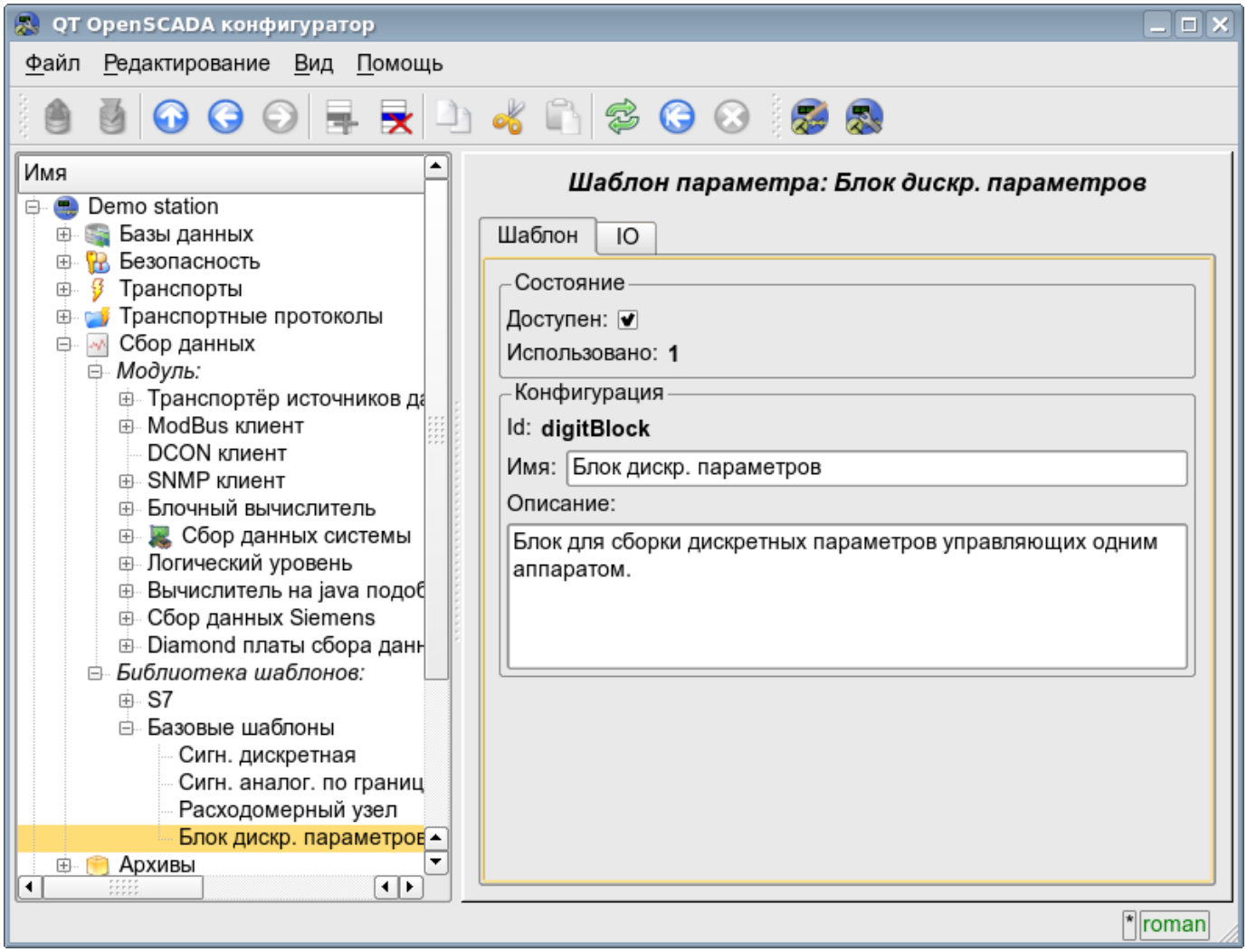


Рис. 4.5f. Главная вкладка конфигурации шаблона параметров подсистемы "Сбор данных".

Вкладка "IO" (рис.4.5g) содержит конфигурацию атрибутов (IO) шаблонов и программу шаблона на одном из языков пользовательского программирования OpenSCADA, например, DAQ.JavaLikeCalc.JavaScript. В таблицу атрибутов шаблона пользователь может, посредством контекстного меню, добавить, вставить, удалить, передвинуть вверх или вниз запись атрибута, а также отредактировать поля атрибута:

- *Id* - идентификатор атрибута.
- *Имя* - имя атрибута.
- *Тип* - выбор типа значения атрибута из списка: "Вещественный", "Целый", "Логический", "Строка".
- *Режим* - выбор режима атрибута из списка: "Вход", "Выход".
- *Атрибут* - режим атрибута параметра, реализованного на основе шаблона из списка: "Не атрибут", "Только для чтения", "Полный доступ". Для атрибутов шаблона, у которых это поле установлено, будет создаваться соответствующий атрибут у параметра контроллера этой подсистемы.

- *Конфигурация* - режим конфигурации атрибута во вкладке конфигурации шаблона у параметра контроллера этой подсистемы из списка: "Константа", "Публичная константа", "Связь". В режимах "Публичная константа" и "Связь" во вкладке конфигурации шаблона будут добавлены эти атрибуты для установки константы или указания внешней связи параметра.
- *Значение* - значение атрибута по умолчанию или шаблон ссылки для доступа по ссылке. Формат шаблона ссылки зависит от компонента, который его использует. Обычно для модуля DAQ.LogicLev шаблон ссылки записывается в виде: **{Параметр}|{атрибут}**. Поле **{Параметр}** - указывает имя параметра как контейнера атрибутов. Атрибуты с одинаковым значением **{Параметр}** будут группироваться и позволят назначаться только указанием контейнера атрибутов, а отдельные атрибуты будут связаны с атрибутами контейнера в соответствии с полем **{атрибут}**.

С синтаксисом языка программы шаблона можно ознакомиться в документации модуля, предоставляющего интерпретатор выбранного языка. Например, типичным языком пользовательского программирования OpenSCADA является DAQ.JavaLikeCalc.JavaScript

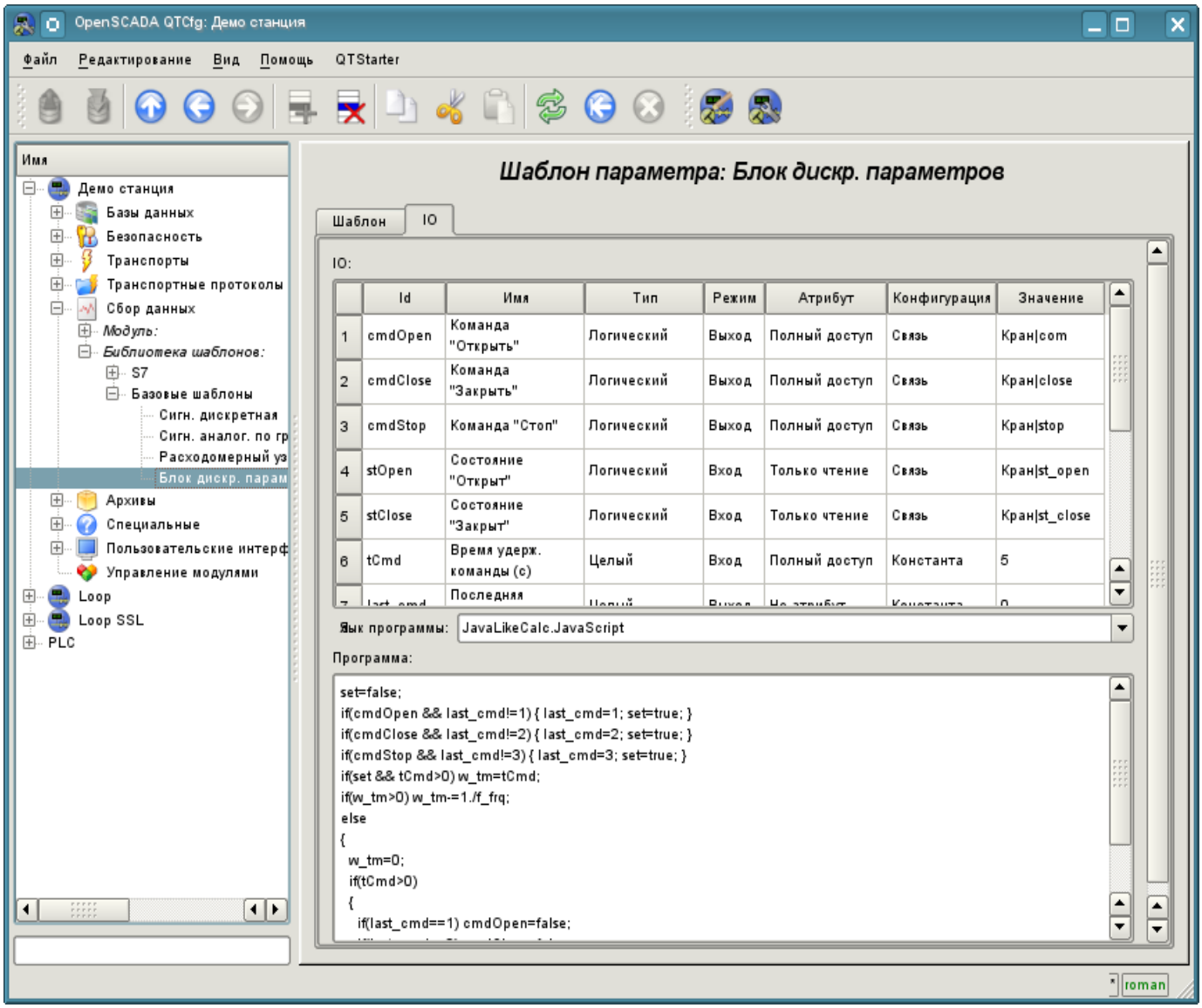


Рис. 4.5g. Вкладка конфигурации атрибутов и программы шаблона подсистемы "Сбор данных".

Каждый модуль подсистемы "Сбор данных" предоставляет конфигурационную страницу с вкладками "Контроллеры" и "Помощь". Вкладка "Контроллеры" (рис.4.5h) содержит список контроллеров, зарегистрированных в модуле. В контекстном меню списка пользователю предоставляется возможность добавления, удаления и перехода к нужному контроллеру. Во вкладке "Помощь" содержится информация о модуле подсистемы "Сбор данных" (рис.4.1d), состав которой идентичен для всех модулей.

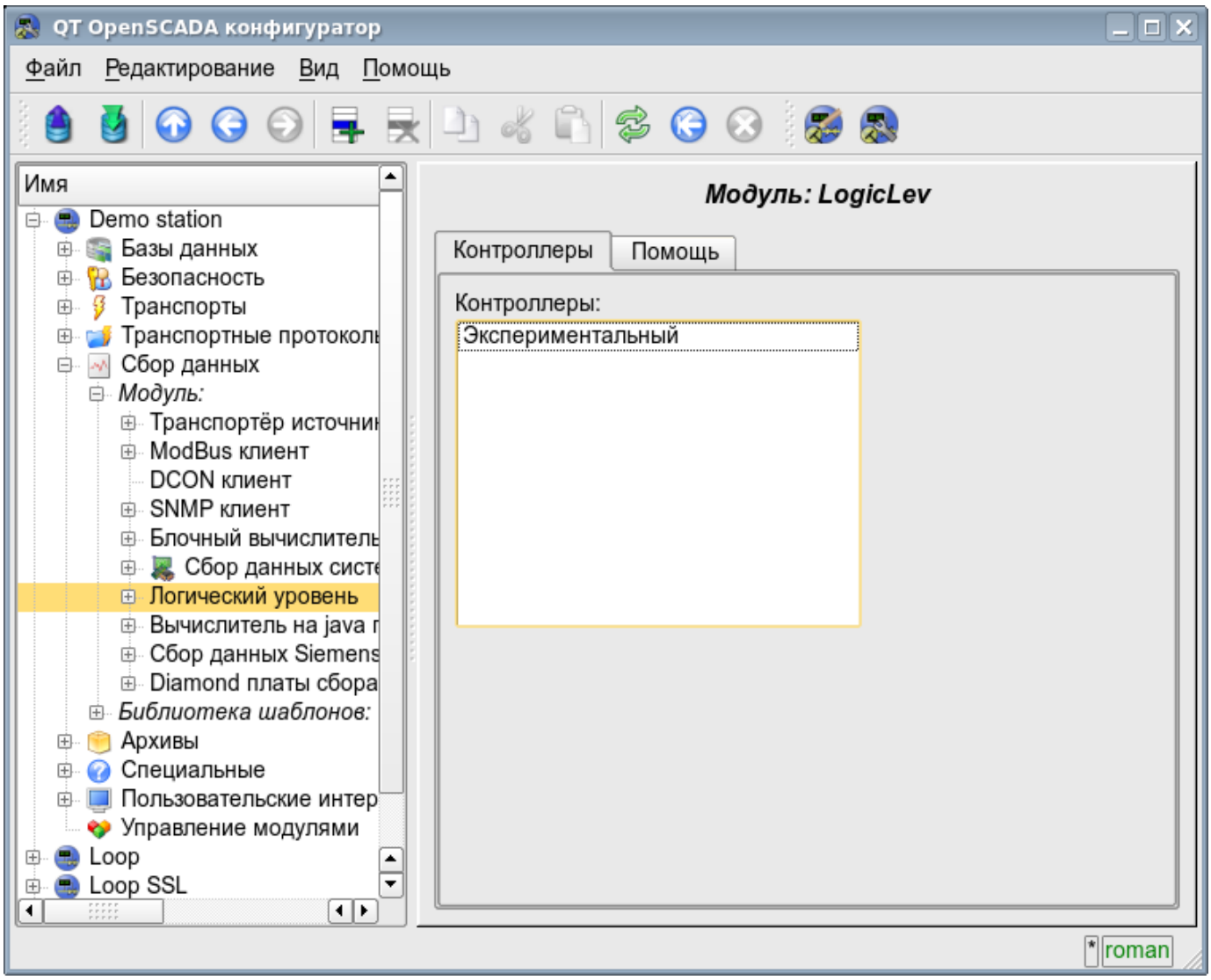


Рис. 4.5h. Вкладка "Контроллеры" модуля подсистемы "Сбор данных".

Каждый контроллер содержит собственную страницу конфигурации с вкладками "Контроллер" и "Параметры".

Вкладка "Контроллер" (рис.4.5i) содержит основные настройки. Состав этих настроек может несколько отличаться от одного модуля этой подсистемы к другому, о чём можно узнать в собственной документации модулей. В качестве примера рассмотрим настройки контроллера у модуля контроллера логического DAQ.LogicLev:

- Раздел "Состояние" - содержит свойства, характеризующие состояние контроллера:
  - *Статус* - указывает статус контроллера. В нашем случае контроллер исполняется и время вычисления составляет 0.394644 милисекунды.
  - *Включен* - состояние контроллера "Включен". Включенный контроллер предоставляет возможность создания параметров и их конфигурации.
  - *Запущен* - состояние контроллера "Запущен". Исполняющийся контроллер выполняет физический сбор данных и/или включает механизмы доступа к этим данным.
  - *БД контроллера* - адрес БД для хранения данных контроллера и его параметров.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:

- *ID* - информация об идентификаторе контроллера.
- *Имя* - указывает имя контроллера.
- *Описание* - краткое описание контроллера и его назначения.
- *Включать* - указывает на состояние "Включать," в которое переводить контроллер при загрузке.
- *Запускать* - указывает на состояние "Запущен", в которое переводить контроллер при загрузке.
- *Режим горизонтального резервирования* - включает контроллер в схему горизонтального резервирования сбора данных подсистемы "Сбор данных" и указывает режим резервирования: "Асимметричный" или "Симметричный".
- *Предпочтение исполнения данного контроллера* - указывает предпочтения исполнения данного контроллера на станции в схеме резервирования.
- *Таблица параметров* - имя таблицы, в которой сохранять параметры (имеются в виду объекты параметров сбора данных) контроллера.
- *Период получения данных (мс)* - периодичность задачи сбора данных. В нашем примере это периодичность вычисления шаблона.
- *Уровень приоритета задачи получения данных* - устанавливает приоритетность задачи сбора данных этого контроллера. Используется при планировании задач операционной системой. В случае исполнения станции от имени суперпользователя "root" это поле включает планирование задачи контроллера в режиме реального времени и с указанным приоритетом.

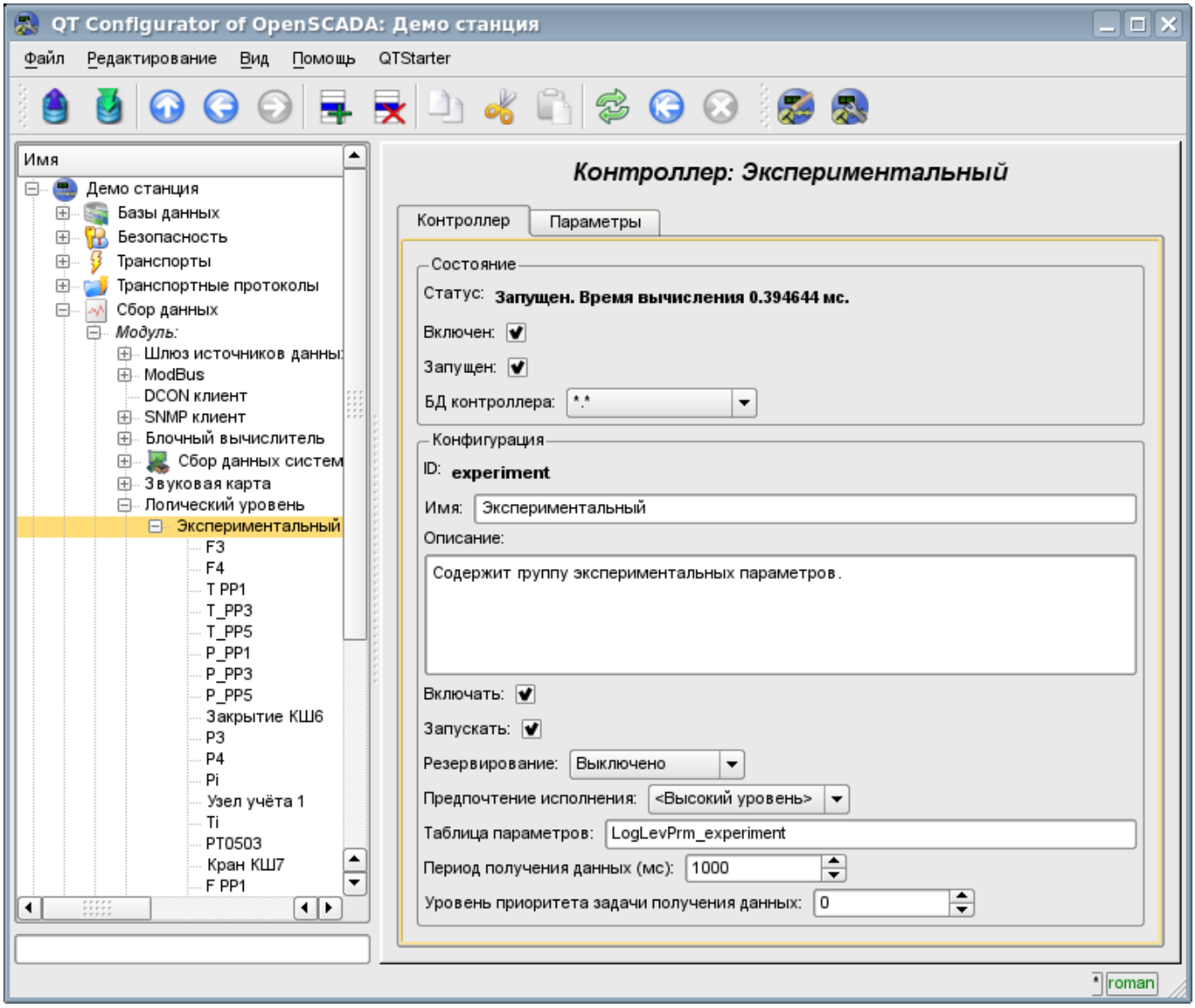


Рис. 4.5i. Главная вкладка конфигурации контроллера подсистемы "Сбор данных".

Вкладка "Параметры" (рис.4.5j) содержит список параметров в контроллере, а также информацию об общем количестве и количестве включенных параметров. В контекстном меню списка пользователю предоставляется возможность добавления, удаления и перехода к нужному параметру.

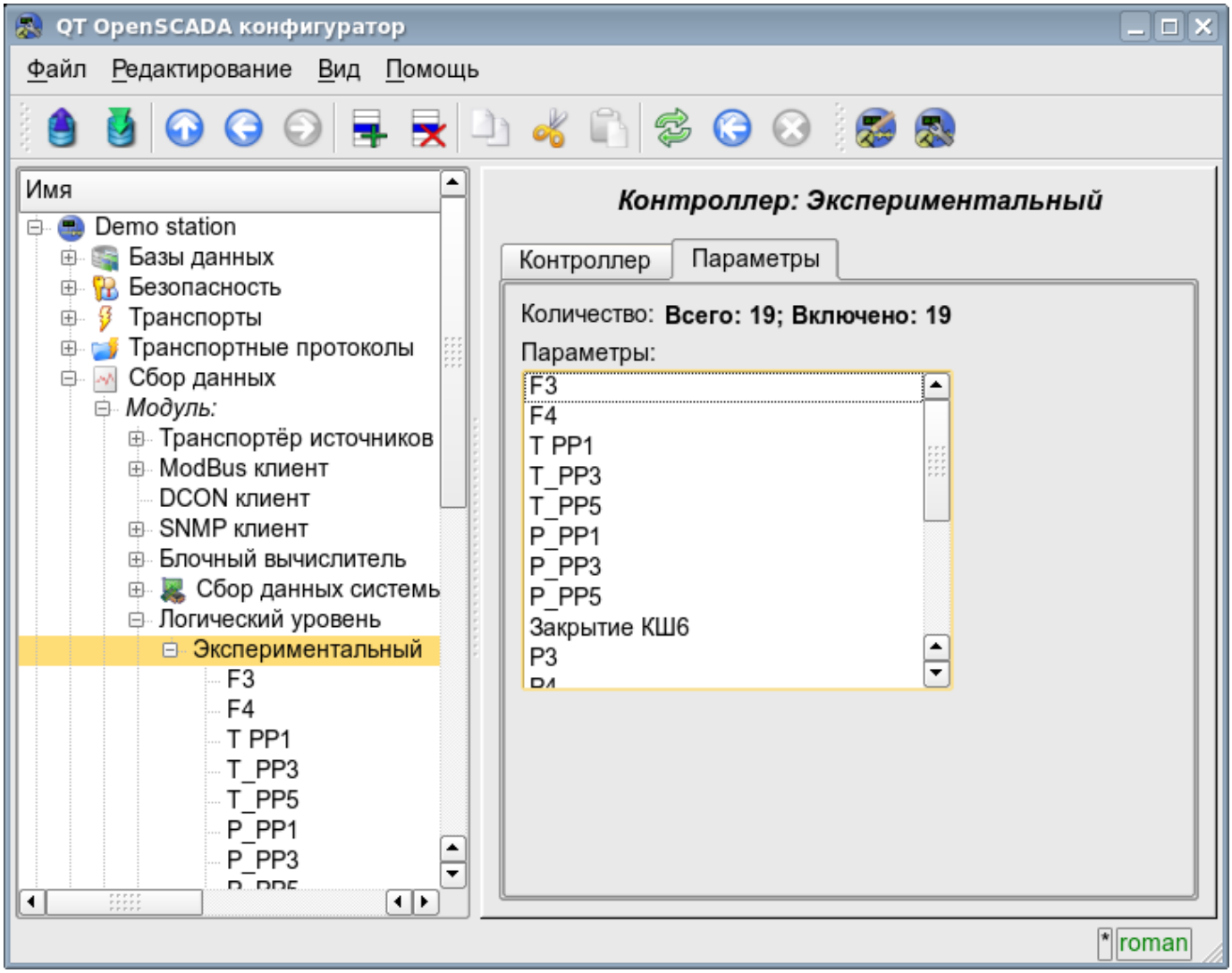


Рис. 4.5j. Вкладка "Параметры" страницы конфигурации контроллера подсистемы "Сбор данных".

Параметры контроллеров подсистемы "Сбор данных" предоставляют конфигурационную страницу с вкладками "Параметр", "Атрибуты", "Архивация" и "Конфигурация шаблона". Вкладка "Конфигурация шаблона" не является стандартной, а присутствует только в модулях подсистемы "Сбор данных", которые реализуют механизмы работы по шаблону в контексте источника данных, ими обслуживаемого. В данный обзор эта вкладка включена для обеспечения логической завершённости обзора конфигурации шаблонов параметров подсистемы "Сбор данных" как финальный этап - использования.

Вкладка "Параметр" (рис.4.5k) содержит основные настройки в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние параметра:
  - *Тип* - информация о типе параметра.
  - *Включен* - состояние параметра "Включен". Включенный параметр используется контроллером для сбора данных.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - содержит информацию об идентификаторе параметра.
  - *Имя* - указывает имя параметра.
  - *Описание* - краткое описание параметра и его назначения.
  - *Включать* - указывает на состояние "Включать", в которое переводить параметр при загрузке.
  - *Режим* - содержит два поля: непосредственно режим и его конфигурацию. В случае

с параметром контроллера модуля данного типа это режим работы "по шаблону" и адрес ранее нами рассмотренного шаблона.

Вкладка "Атрибуты" (рис.4.5l) содержит атрибуты параметра и их значения в соответствии с конфигурацией используемого шаблона и вычисления его программы.

Вкладка "Архивация" (рис.4.5m) содержит таблицу с атрибутами параметра в колонках, и архиваторами в строках. Пользователь имеет возможность установить архивацию нужного атрибута требуемым архиватором просто изменив ячейку на пересечении.

Вкладка "Конфигурация шаблона" (рис.4.5n) содержит конфигурационные поля в соответствии с шаблоном. В примере это групповая связь на внешний параметр. Эту связь можно установить, просто указав путь к параметру, если флаг "Показывать только атрибуты" не установлен, или же установить адреса атрибутов по отдельности, если флаг установлен.

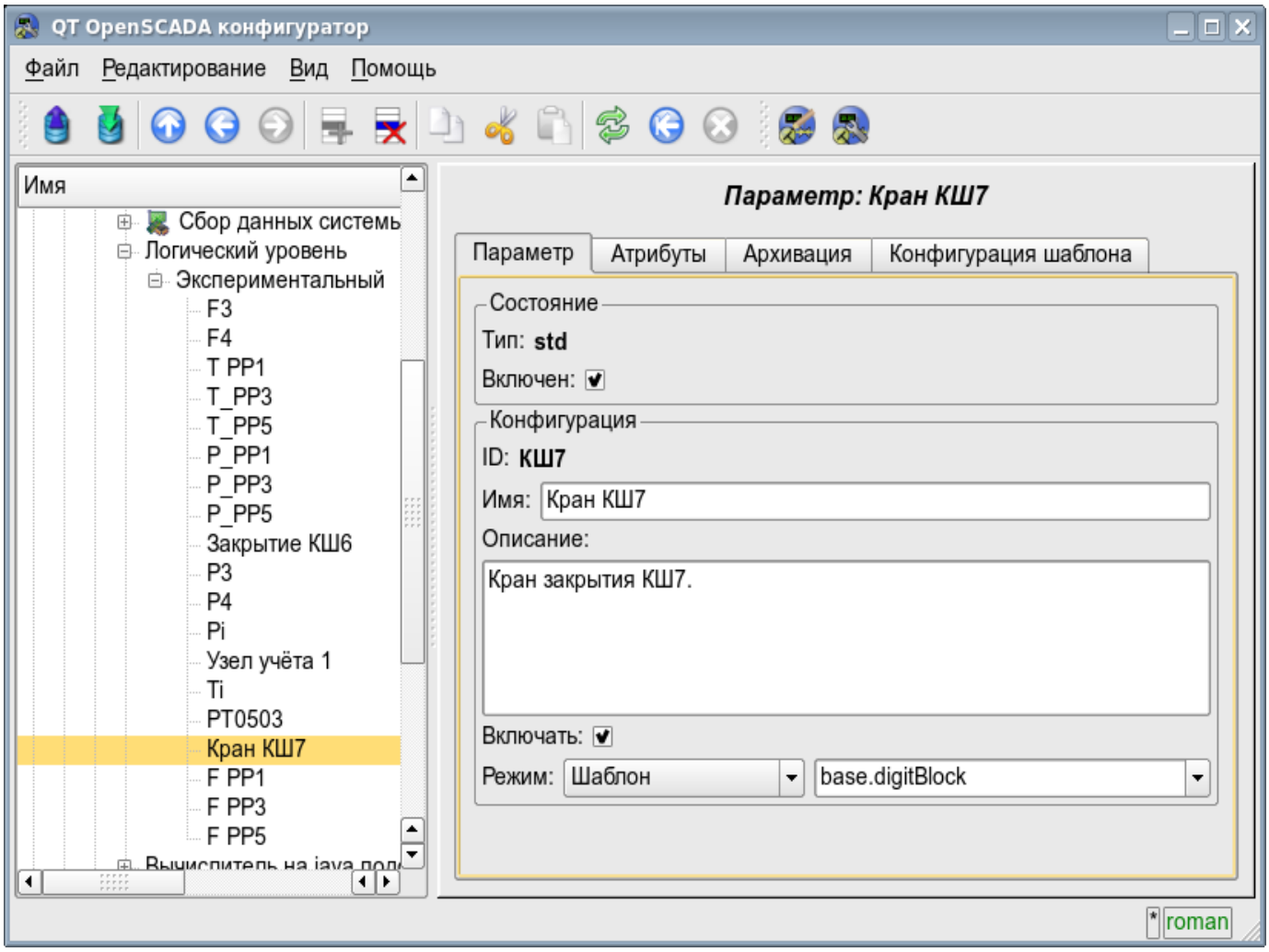


Рис. 4.5k. Главная вкладка конфигурации параметра контроллера подсистемы "Сбор данных".

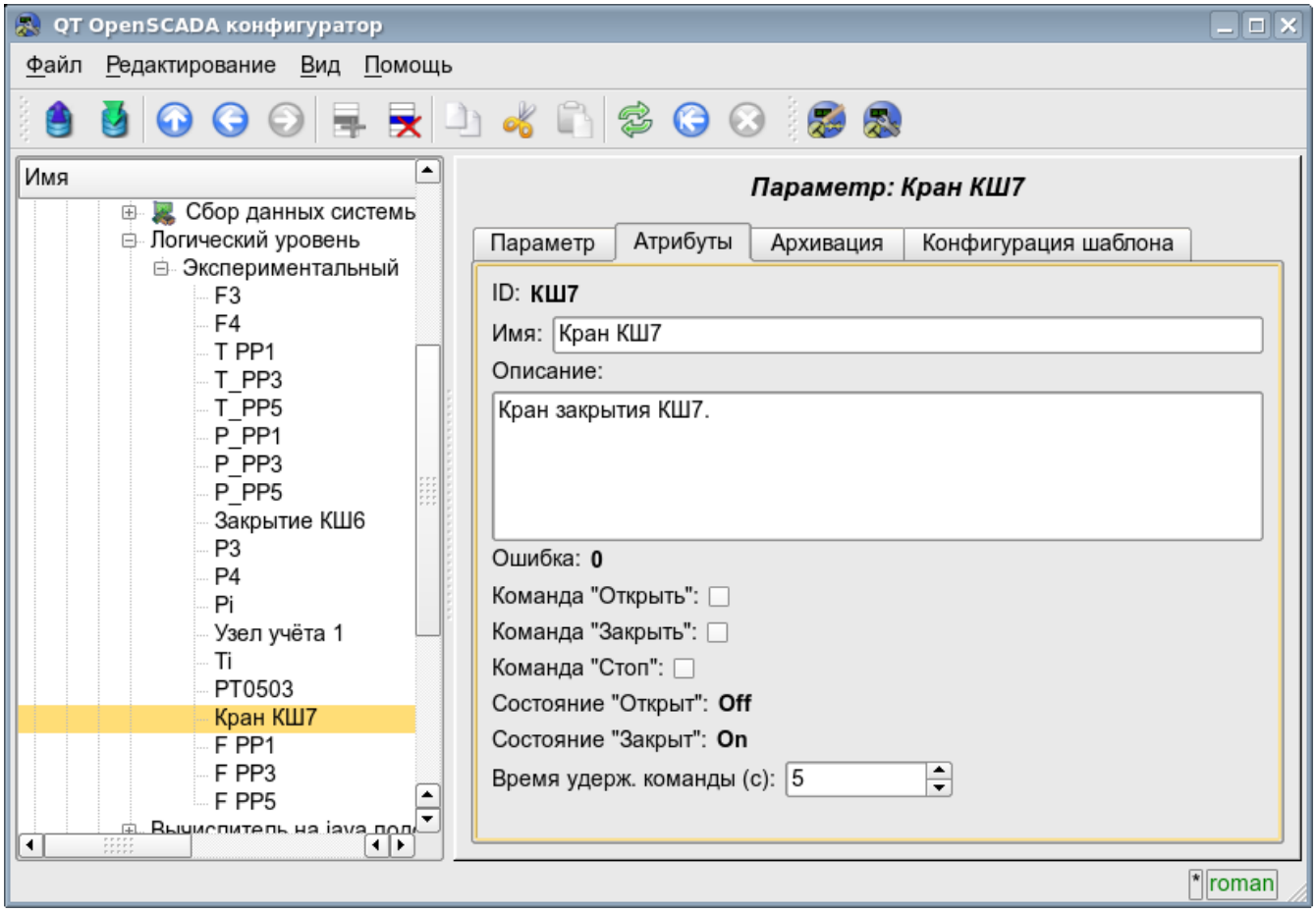


Рис. 4.5l. Вкладка "Атрибуты" параметра контроллера подсистемы "Сбор данных".

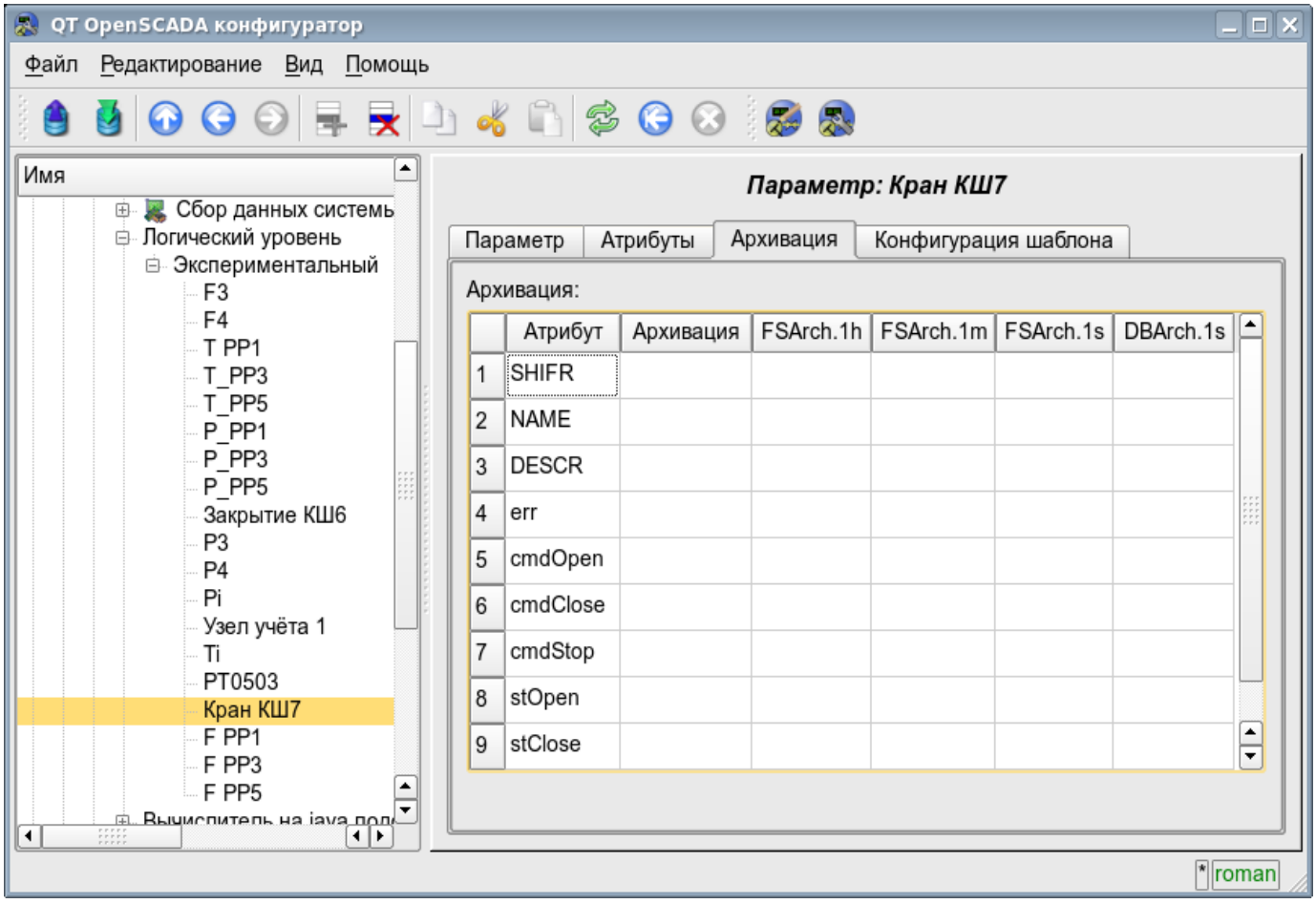


Рис. 4.5m. Вкладка "Архивация" параметра контроллера подсистемы "Сбор данных".

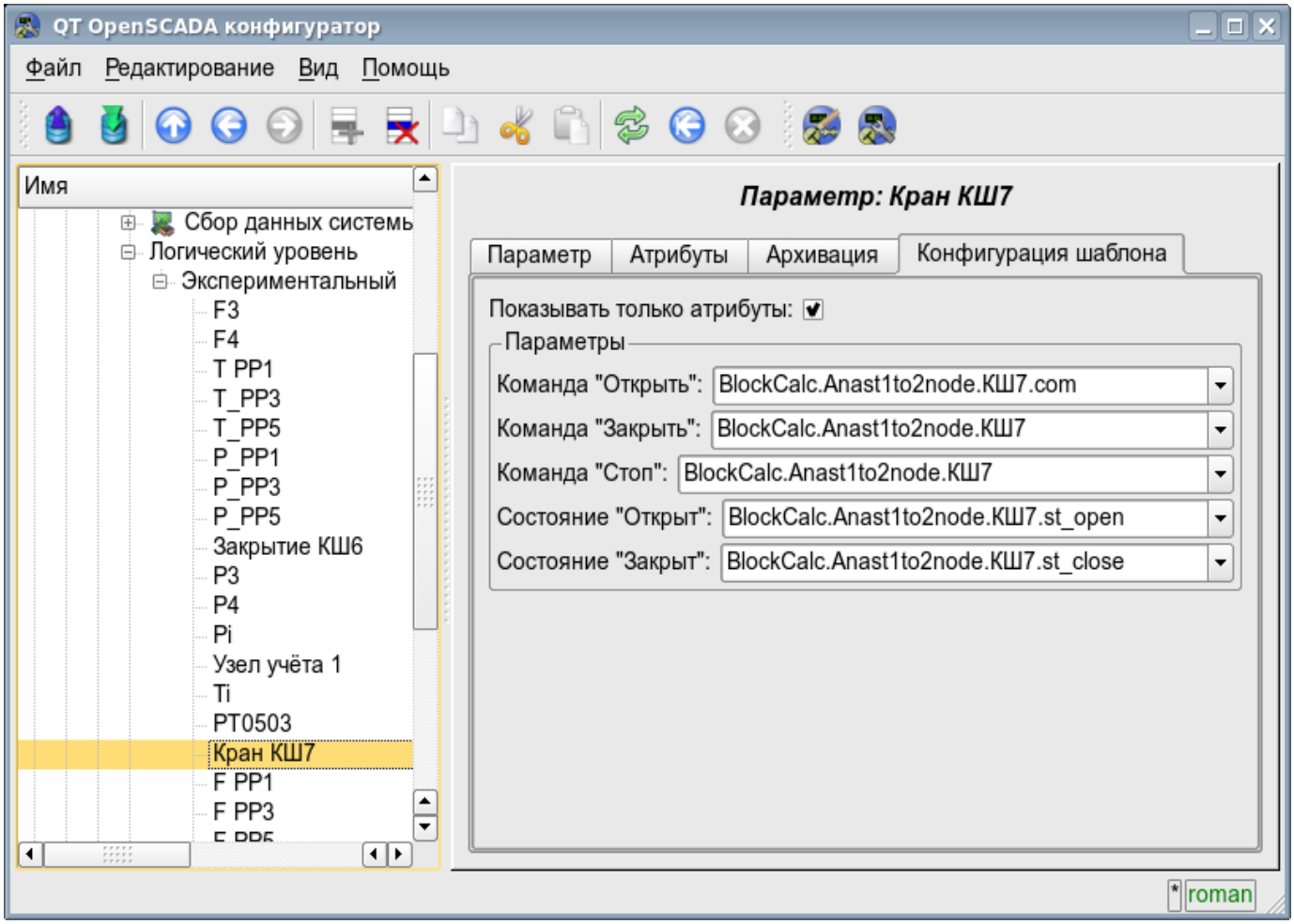


Рис. 4.5n. Вкладка "Конфигурация шаблона" параметра контроллера подсистемы "Сбор данных".

4.6. Подсистема "Архивы"

Подсистема является модульной и содержит иерархию объектов, изображённую на рис.4.6a. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Архивы", содержащая вкладки "Архив сообщений", "Архивы значений", "Модули" и "Помощь".

Для получения доступа на модификацию объектов этой подсистемы необходимы права пользователя в группе "Archive" или права привилегированного пользователя.

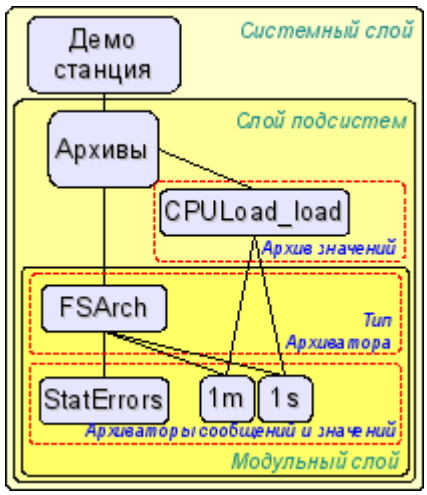


Рис. 4.6a. Иерархическая структура подсистемы "Архивы".

Вкладка "Архив сообщений" (рис.4.6b) содержит конфигурацию архива сообщений и форму запроса сообщений из архива.

Конфигурация архива сообщений представлена полями:

- *Макс. кол. запраш. сообщений* - указывает глобальное ограничение на максимальное количество сообщений, обрабатываемое одним запросом.
- *Размер буфера сообщений* - указывает на размерность области оперативной памяти зарезервированное на промежуточный буфер сообщений. Сообщения из буфера запрашиваются для просмотра и архивируются архиваторами сообщений.
- *Период архивирования сообщений (с)* - периодичность, с которой архиваторы выбирают сообщения из буфера для их архивирования.

Форма запроса сообщений содержит конфигурационные поля запроса и таблицу результата. Конфигурационные поля запроса:

- *Время* - указывает время запроса.
- *Размер (сек)* - указывает размер или глубину запроса в секундах.
- *Шаблон категории* - указывает категорию запрошенных сообщений. В категории можно указывать элементы выборки по шаблону, а именно символы "*" - для любой подстроки и "?" - для любого символа.
- *Уровень* - указывает на минимальный уровень сообщений, т.е. запрос будет обработан для сообщений с уровнем, более или равным указанному.
- *Архиватор* - указывает архиватор сообщений, для которого обрабатывать запрос. Если значение отсутствует, то запрос будет обработан для буфера и всех архиваторов. Если указано <buffer>, то запрос будет обработан только для буфера сообщений.

Таблица результата содержит строки сообщений с колонками:

- *Время* - время сообщения.
- *Категория* - категория сообщения.
- *Уровень* - уровень сообщения.
- *Сообщение* - текст сообщения.

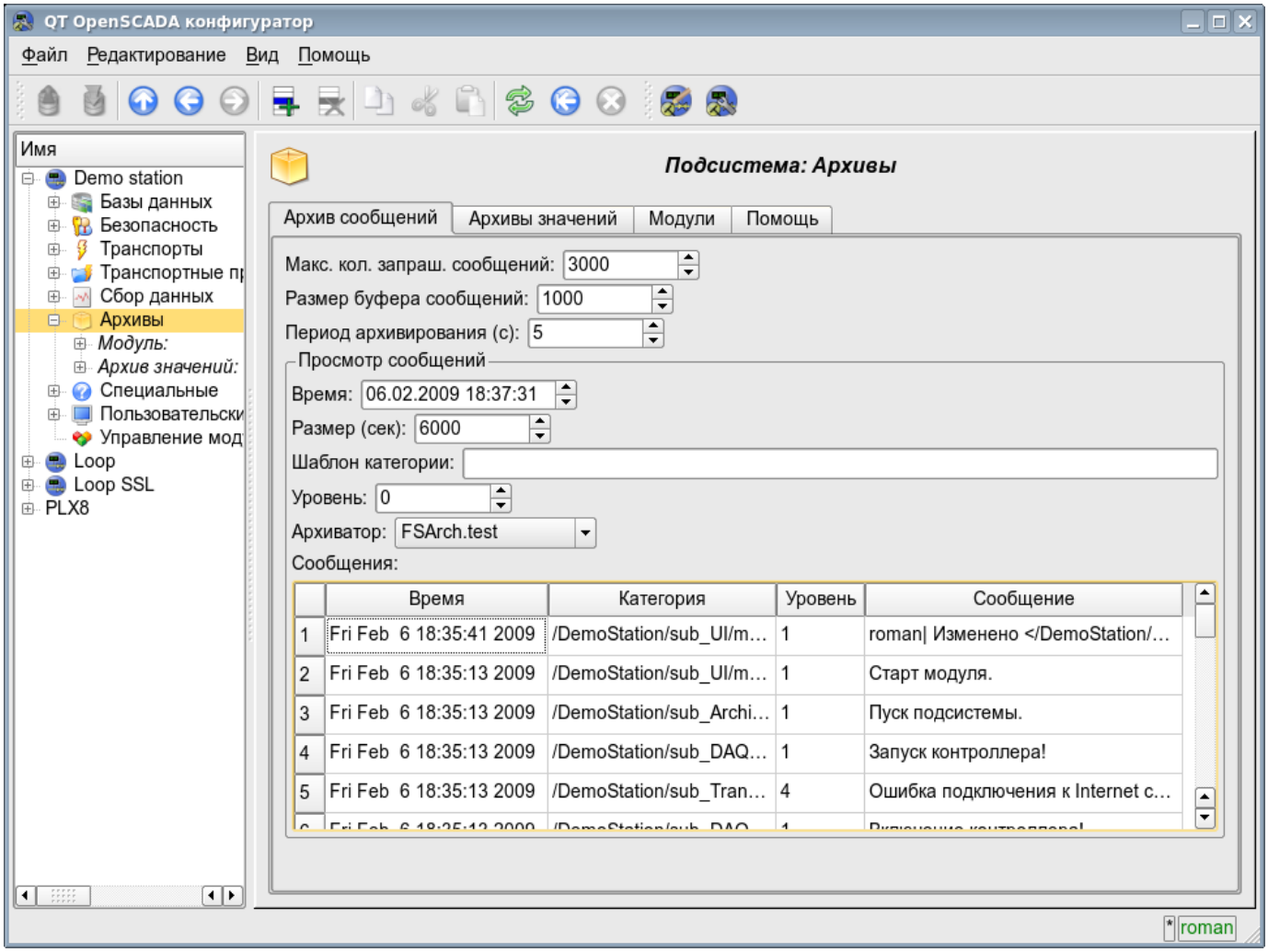


Рис. 4.6b. Вкладка "Архив сообщений" подсистемы "Архивы".

Вкладка "Архивы значений" (рис.4.6c) содержит общую конфигурацию архивирования значений и список архивов значений. В контекстном меню списка значений пользователю предоставляется возможность добавления, удаления и перехода к нужному архиву. Общая конфигурация архивирования представлена полями:

- *Период получения данных (мс)* - указывает периодичность задачи активного архивирования. Фактически, максимальная детализация или минимальный период, активных архивов определяется этим значением.
- *Уровень приоритета задачи получения данных* - устанавливает приоритетность задачи активного архивирования. Используется при планировании задач операционной системой. В случае исполнения станции от имени суперпользователя "root" это поле включает планирование задачи архивирования в режиме реального времени и с указанным приоритетом.

Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы "Архивы" и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

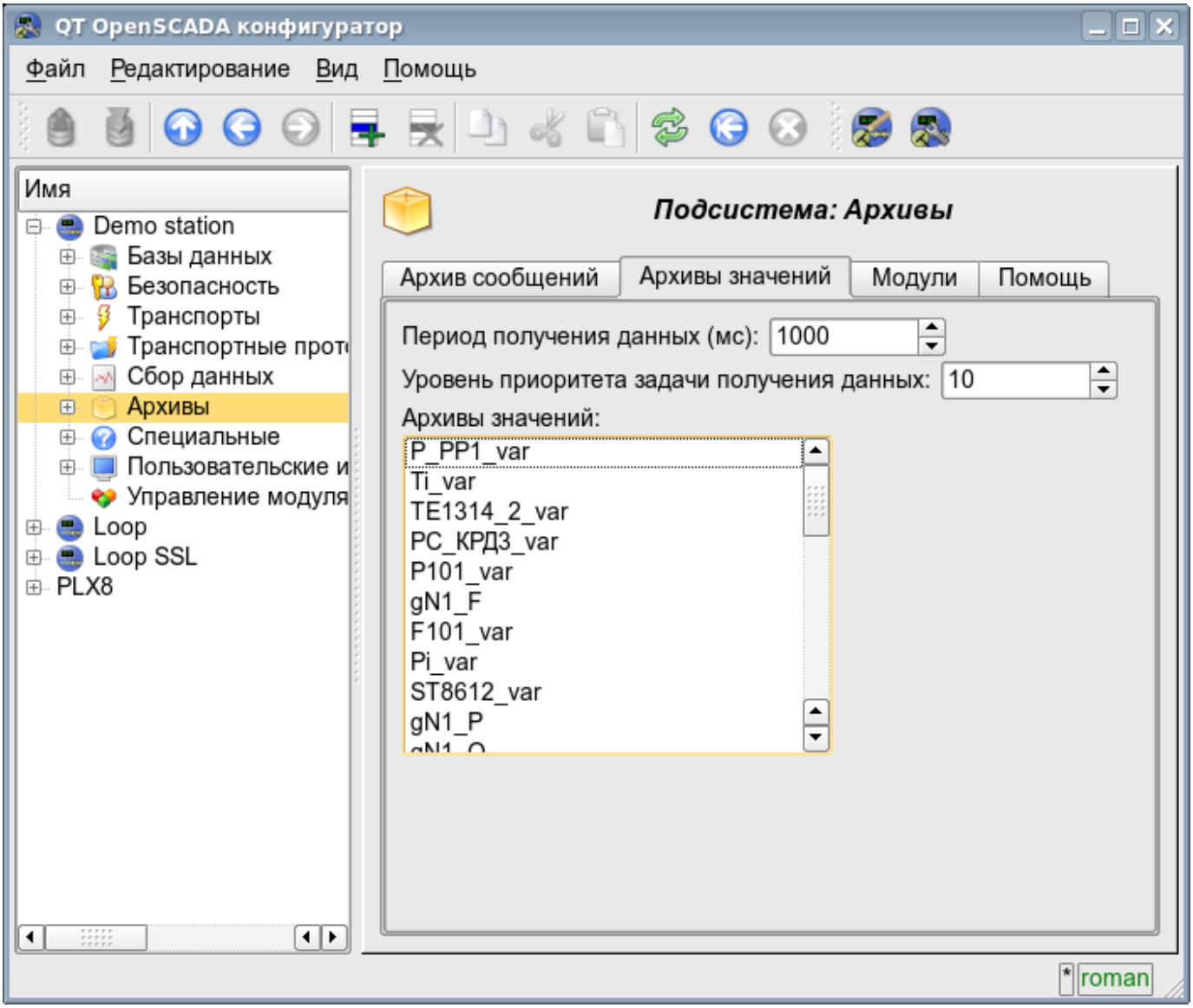


Рис. 4.6c. Вкладка "Архивы значений" подсистемы "Архивы".

Архив значений подсистемы "Архивы" предоставляет конфигурационную страницу с вкладками "Архив", "Архиваторы" и "Значения".

Вкладка "Архив" (рис.4.6d) содержит основные настройки архива в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние архива:
  - *Выполняется* - состояние параметра "Выполняется". Исполняющийся архив собирает данные в буфер и обслуживается архиваторами.
  - *БД архива* - адрес БД для хранения данных архива.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе архива.
  - *Имя* - указывает имя архива.
  - *Описание* - краткое описание архива и его назначения.
  - *Запускать* - указывает на состояние "Выполняется", в которое переводить архив при загрузке.
  - *Тип значений* - указывает на тип значений, хранящихся в архиве, из списка: "Логический", "Целый", "Вещественный" и "Строка".
  - *Источник* - указывает на тип и адрес источника. Тип источника указывается из списка: "Пассивный", "Пассивный атрибут параметра" или "Активный атрибут параметра". Пассивный архив не имеет ассоциированного источника значений; данные в такой архив источник передаёт самостоятельно. Типы с атрибутом параметра в поле адреса указывают на параметр подсистемы "Сбор данных" как источник. Пассивный атрибут параметра направляет данные в архив самостоятельно с собственным

периодом сбора данных. Активный атрибут параметра опрашивается задачей архивирования этой подсистемы.

- *Период буфера (сек)* - указывает на периодичность значений в буфере архива.
- *Размер буфера (единиц)* - указывает размерность или глубину буфера архива. Размерность обычно устанавливается в пересчёте на 60 сек периодичности задачи архивирования с запасом.
- *Жесткая сетка времени буфера* - указывает на режим буфера. Режим жёсткой сетки подразумевает резервирование памяти под каждое значение, но без метки времени. Такой режим исключает возможность упаковки смежно-одинаковых значений, но экономит на хранении метки времени. Иначе буфер работает в режиме хранения значения и метки времени и поддерживает упаковку смежно-одинаковых значений.
- *Высокое разрешение времени буфера* - указывает на возможность хранения значений с периодичностью до 1 микросекунды, иначе значения могут храниться с периодичностью до 1 секунды.

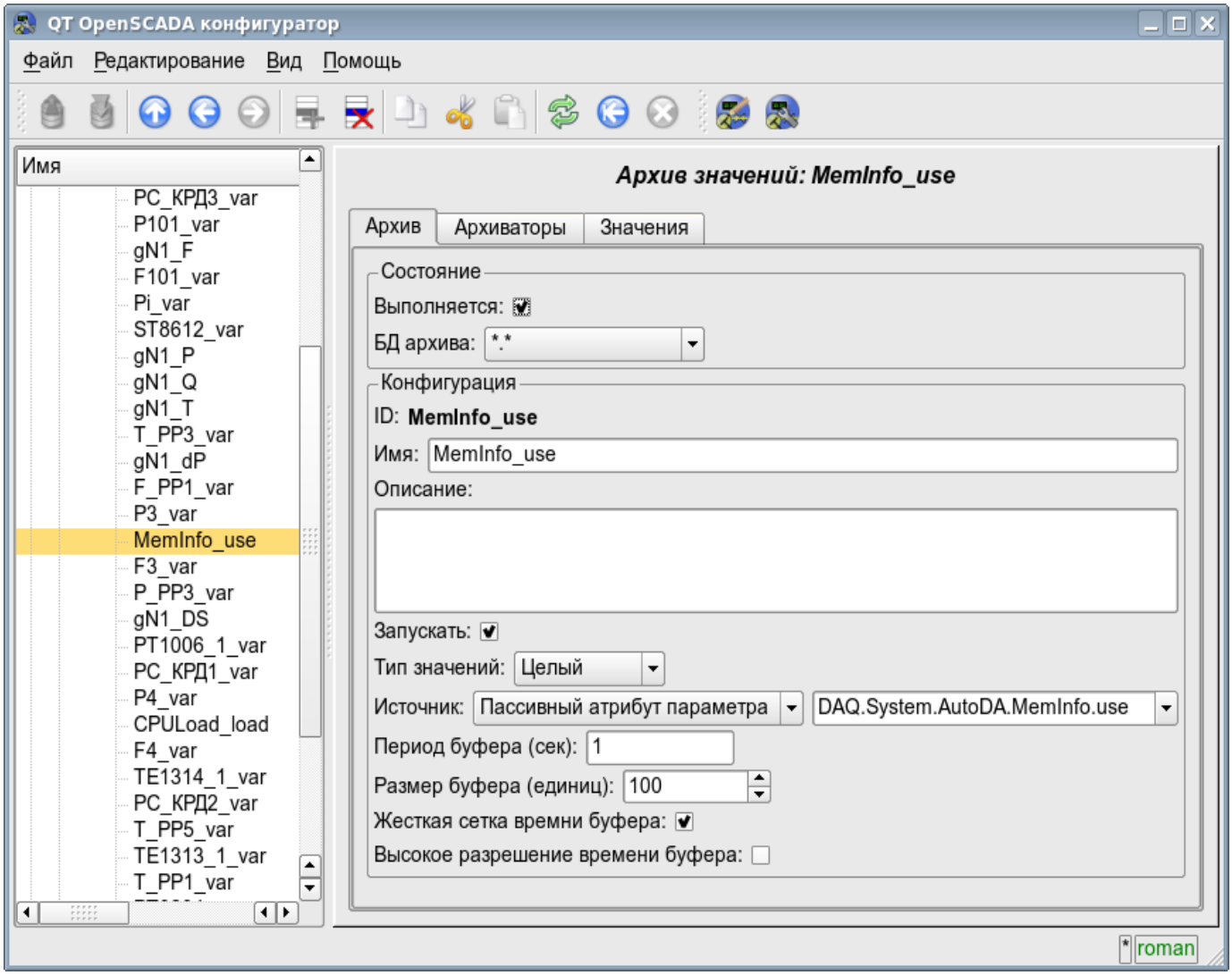


Рис. 4.6d. Основная вкладка конфигурации архива значений подсистемы "Архивы".

Вкладка "Архиваторы" (рис.4.6е) содержит таблицу с конфигурацией процесса обработки данного архива доступными архиваторами. В строках расположены доступные архиваторы, а в колонках параметры:

- *Архиватор* - информация об адресе архиватора.
- *Запущен* - информация о состоянии "Запущен" архиватора.
- *Обработка* - признак обработки данного архива архиватором. Поле доступно для модификации пользователем.
- *Период (с)* - информация о периодичности архиватора.
- *Начало* - дата начала данных архива в архиваторе.
- *Конец* - дата конца данных архива в архиваторе.

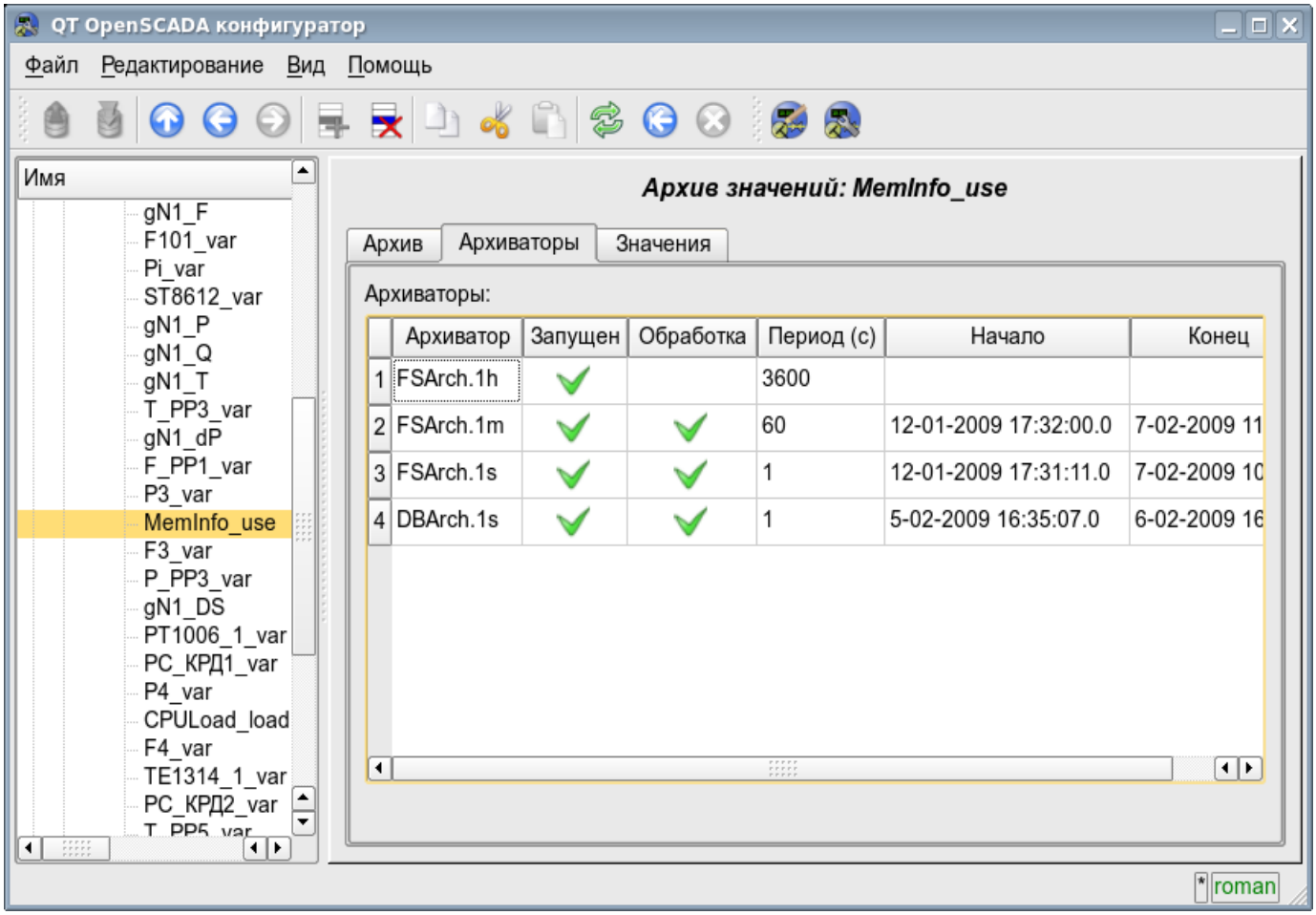


Рис. 4.6е. Вкладка "Архиваторы" архива значений подсистемы "Архивы".

Вкладка "Значения" (рис.4.6f) содержит запрос значений в архиве и результат в виде таблицы значений или изображения тренда. Запрос значений содержит поля:

- *Время* - указывает время запроса. Содержит два поля: поле дата+время и микросекунды.
- *Размер* - указывает размер или глубину запроса в секундах.
- *Архиватор* - указывает архиватор значений для которого обрабатывать запрос. Если значение отсутствует, то запрос будет обработан для буфера и всех архиваторов. Если указано <buffer>, то запрос будет обработан только для буфера архива.
- *Показать график* - указывает на необходимость представления данных архива в виде графика (тренда), иначе результат представляется в таблице, содержащей только время и значение. В случае установки этого поля формируется и отображается график, кроме того появляются дополнительные конфигурационные поля настройки изображения:
  - *Размер изображения* - указывает ширину и высоту формируемого изображения в пикселах.
  - *Шкала значения* - указывает нижнюю и верхнюю границу шкалы значения. Если оба значения установлены в 0 или равны, то шкала будет определяться автоматически в зависимости от значений.

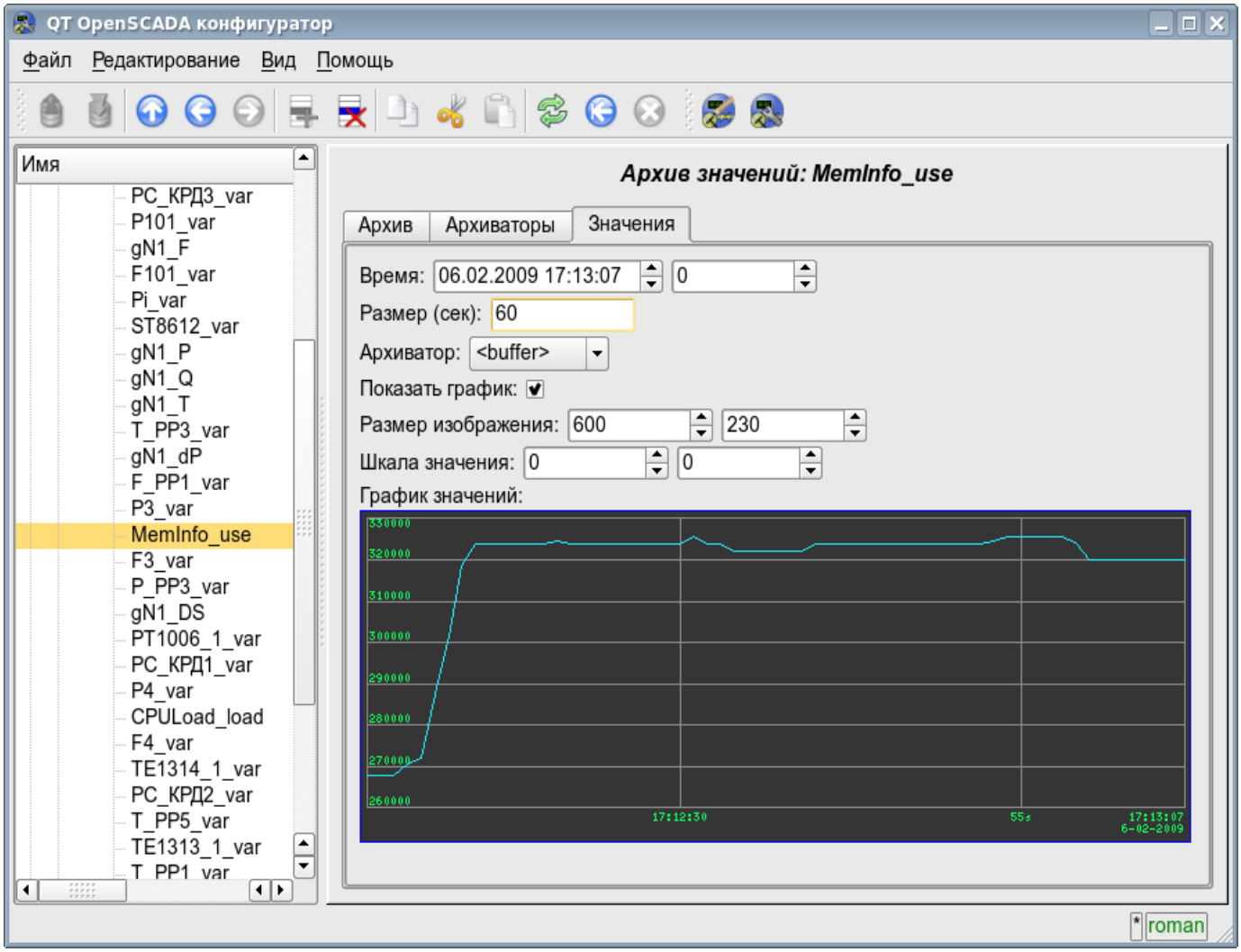


Рис. 4.6f. Вкладка "Значения" архива значений подсистемы "Архивы".

Каждый модуль подсистемы "Архивы" предоставляет конфигурационную страницу с вкладками "Архиваторы" и "Помощь". Вкладка "Архиваторы" (рис.4.6g) содержит списки архиваторов сообщений и значений, зарегистрированных в модуле. В контекстном меню списков пользователю предоставляется возможность добавления, удаления и перехода к нужному контроллеру. Во вкладке "Помощь" содержится информация о модуле подсистемы "Архивы" (рис.4.1d), состав которой идентичен для всех модулей.

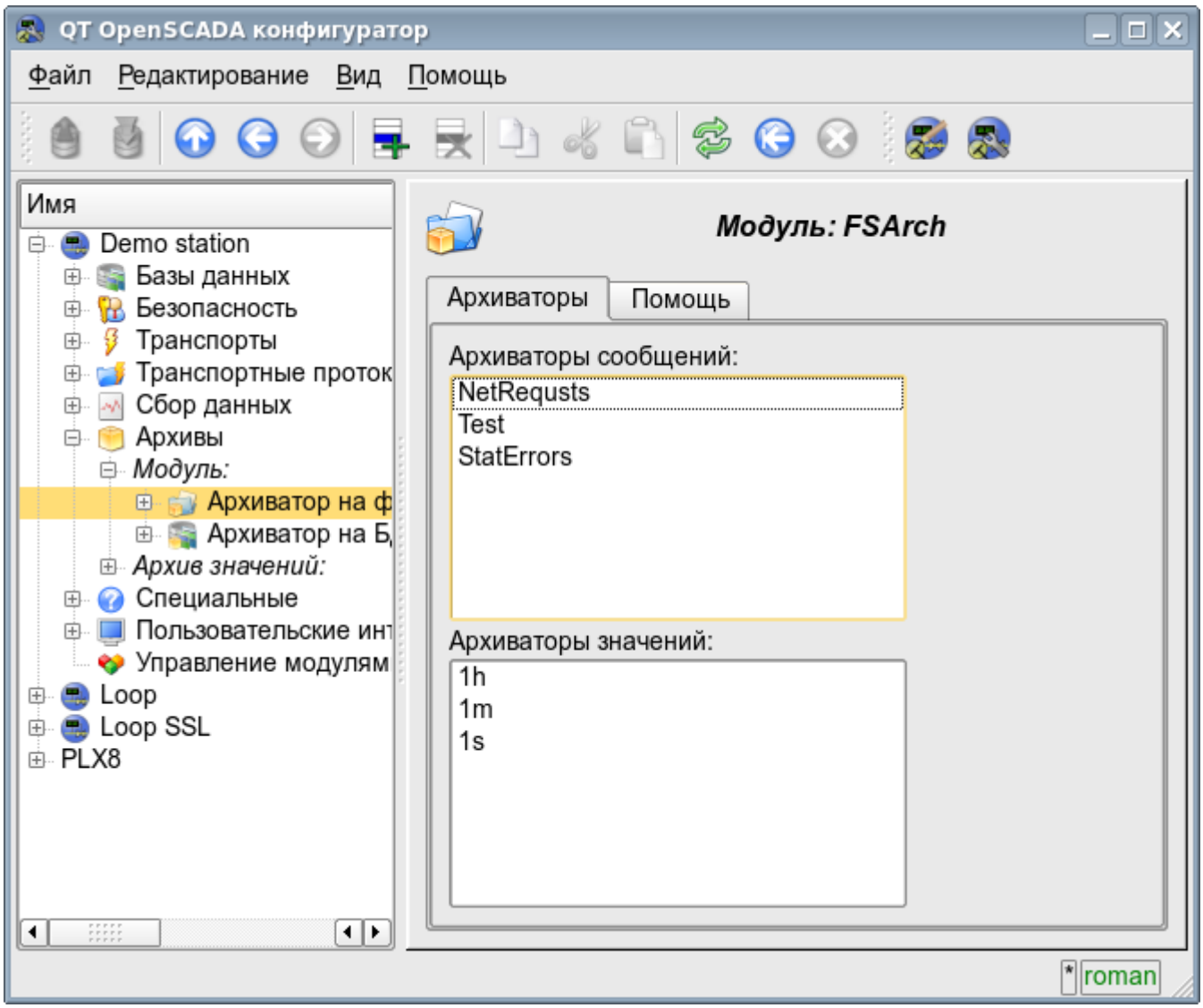


Рис. 4.6g. Вкладка "Архиваторы" модуля подсистемы "Архивы".

Архиваторы сообщений содержат собственную страницу конфигурации с вкладками "Архиватор" и "Сообщения".

Вкладка "Архиватор" (рис.4.6h) содержит основные настройки. Состав этих настроек может несколько отличаться от одного модуля этой подсистемы к другому о чём можно узнать в собственной документации модулей. В качестве примера рассмотрим настройки архиватора сообщений у модуля архива на файловую систему Arch.FSArch Настройки:

- Раздел "Состояние" - содержит свойства характеризующие состояние архиватора:
  - *Выполняется* - состояние архиватора "Выполняется". Исполняющийся архиватор обрабатывает буфер архива сообщений и помещает его данные в своё хранилище, а также обслуживает запросы на доступ к данным в хранилище.
  - *БД архиватора* - адрес БД для хранения данных архиватора.
  - *Конец* - дата+время последних данных в хранилище архиватора.
  - *Начало* - дата+время первых данных в хранилище архиватора.
  - *Размер файлов архиватора (кБ)* - информация о суммарном размере файлов архиватора с данными.
  - *Время архивирования (мс)* - время, затраченное на архивирование данных архива

сообщений.

- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе архиватора.
  - *Имя* - указывает имя архиватора.
  - *Описание* - краткое описание архиватора и его назначения.
  - *Адрес* - адрес хранилища в специфичном для типа архиватора (модуля) формате. Описание формата записи адреса архиватора, как правило, доступно во всплывающей подсказке этого поля. В примере это относительный путь к директории хранилища.
  - *Уровень сообщений* - указывает на уровень сообщений архиватора. Сообщения с уровнем более или равным указанному обрабатываются архиватором.
  - *Категории сообщений* - список категорий, разделённый символом ';' сообщений. Сообщения, попавшие под шаблоны категорий, будут обрабатываться архиватором. В категории можно указывать элементы выборки по шаблону, а именно символы '*' - для любой подстроки и '?' - для любого символа.
  - *Запускать* - указывает на состояние "Выполняется", в которое переводить архиватор при загрузке.
- Раздел "Дополнительные опции" - специализированный для модуля раздел, о содержимом которого, можно ознакомиться в документации на модуль.

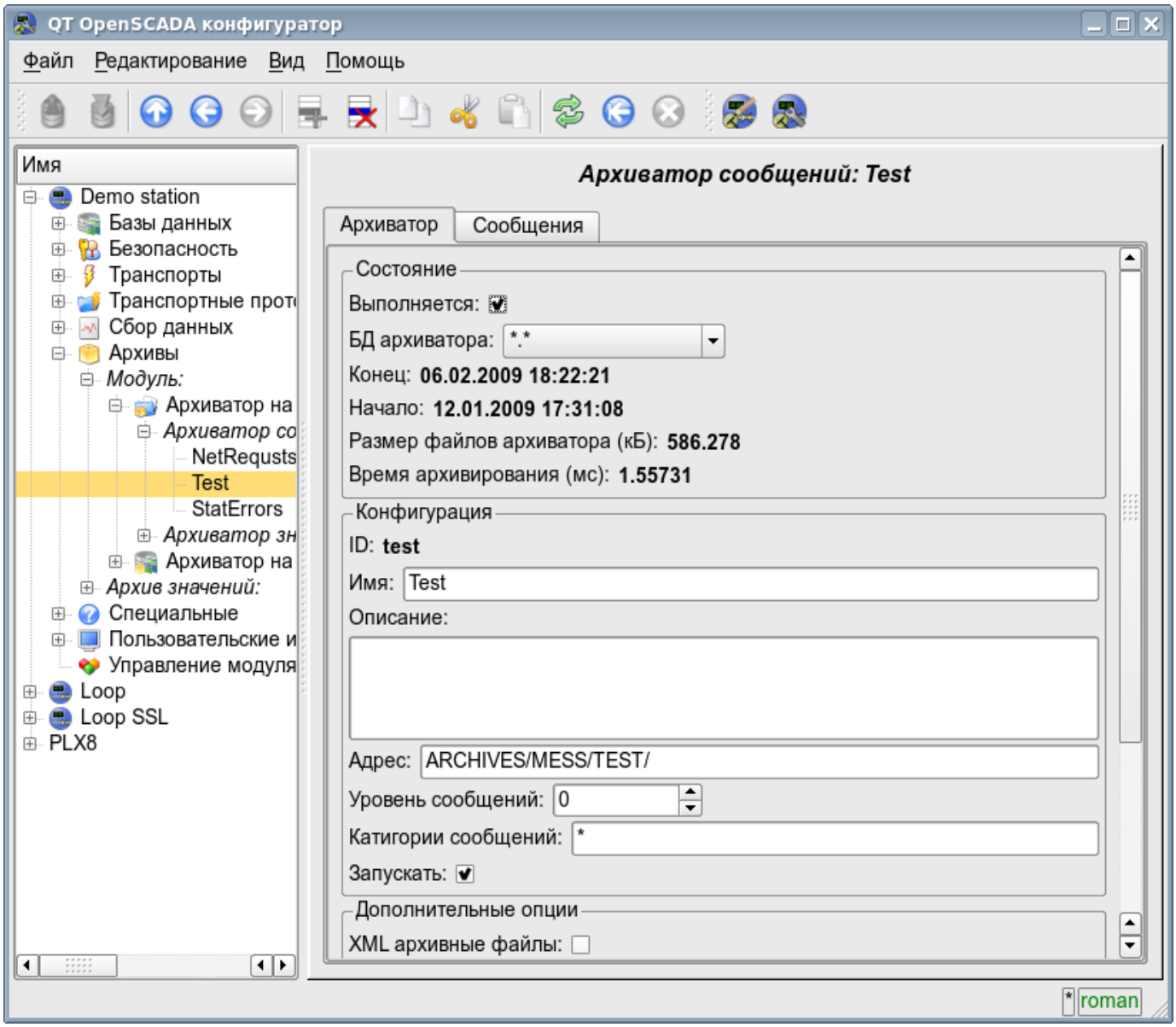


Рис. 4.6h. Главная вкладка конфигурации архиватора сообщений подсистемы "Архивы".

Вкладка "Сообщения" (рис.4.6i) содержит форму запроса сообщений из архива данного архиватора:

- *Время* - указывает время запроса.
- *Размер (сек)* - указывает размер или глубину запроса, в секундах.
- *Шаблон категории* - указывает категорию запрошенных сообщений. В категории можно указывать элементы выборки по шаблону, а именно символы '*' - для любой подстроки и '?' - для любого символа.
- *Уровень* - указывает на минимальный уровень сообщений, т.е. запрос будет обработан для сообщений с уровнем более или равному указанному.

Таблица результата содержит строки сообщений с колонками:

- *Время* - время сообщения.
- *Категория* - категория сообщения.
- *Уровень* - уровень сообщения.
- *Сообщение* - текст сообщения.

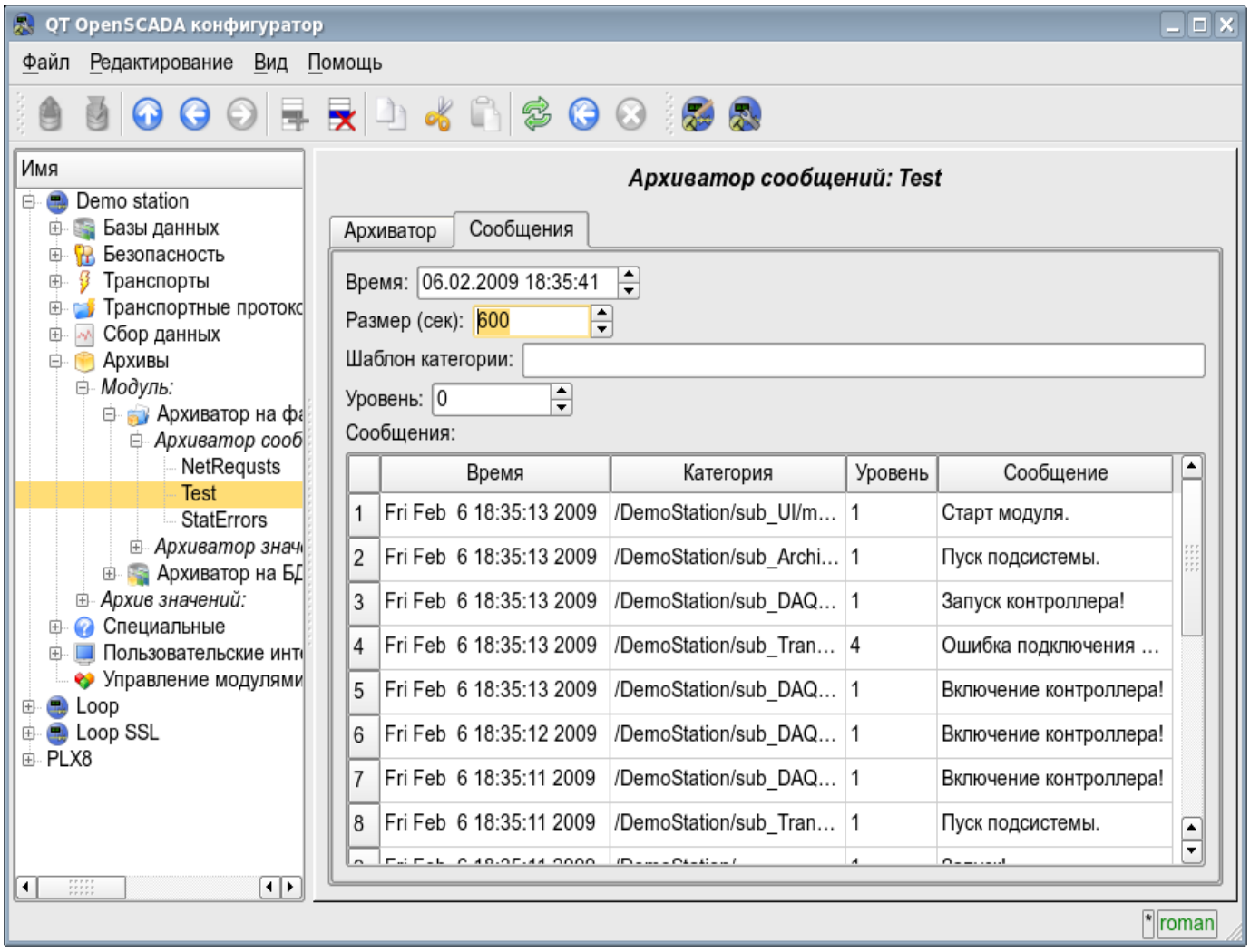


Рис. 4.6i. Вкладка запроса сообщений "Сообщения" архиватора сообщений подсистемы "Архивы".

Архиваторы значений содержат собственную страницу конфигурации с вкладками "Архиватор" и "Архивы".

Вкладка "Архиватор" (рис.4.6j) содержит основные настройки. Состав этих настроек может несколько отличаться от одного модуля этой подсистемы к другому, о чём можно узнать в собственной документации модулей. В качестве примера рассмотрим настройки архиватора значений у модуля архива на файловую систему Arch.FSArch Настройки:

- Раздел "Состояние" - содержит свойства характеризующие состояние архиватора:
  - *Выполняется* - состояние архиватора "Выполняется". Исполняющийся архиватор обрабатывает буфера архивов значений и помещает их данные в своё хранилище, а

также обслуживает запросы на доступ к данным в хранилище.

- *Время архивирования (мс)* - информация о времени затраченном на архивирование данных буферов архивов. Периодичность архивирования указывается в поле "Период архивирования" раздела "Конфигурация" этой вкладки.
- *БД архиватора* - адрес БД для хранения данных архиватора.

- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе архиватора.
  - *Имя* - указывает имя архиватора.
  - *Описание* - краткое описание архиватора и его назначения.
  - *Периодичность значений (сек)* - указывает периодичность значений, которые содержаться в хранилище архиватора.
  - *Период архивирования (сек)* - указывает периодичность задачи архивирования данных буферов архивов. Размерность буферов архивов во временном выражении должна быть не менее, а лучше несколько больше, периодичности задачи архивирования.
  - *Адрес* - адрес хранилища в специфичном для типа архиватора (модуля) формате. Описание формата записи адреса архиватора, как правило, доступно во всплывающей подсказке этого поля. В примере это относительный путь к директории хранилища.
  - *Запускать* - указывает на состояние "Выполняется", в которое переводить архиватор при загрузке.
- Раздел "Дополнительные опции" - специализированный для модуля раздел, о содержимом которого можно ознакомиться в документации на модуль.

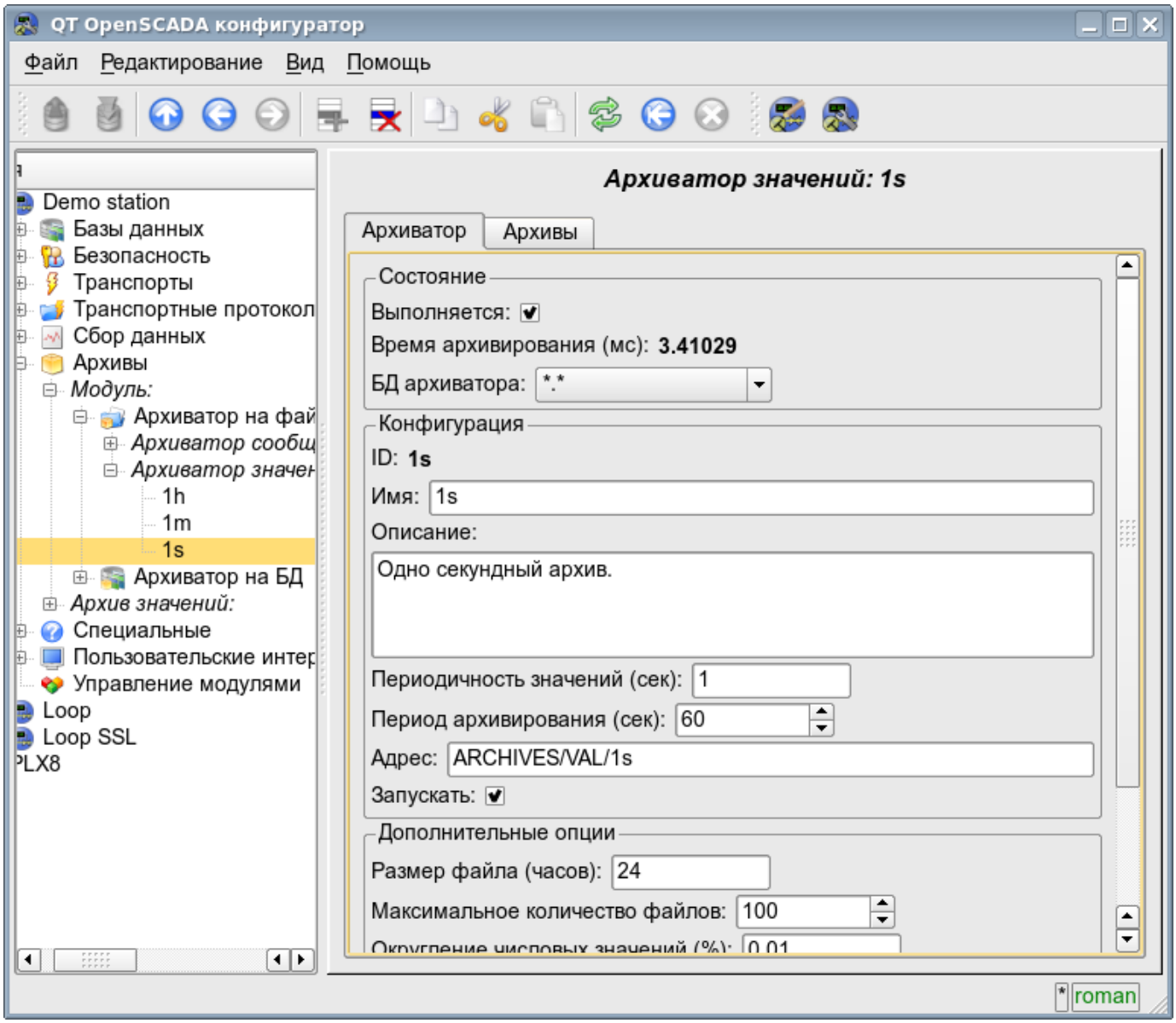


Рис. 4.6j. Главная вкладка конфигурации архиватора значений подсистемы "Архивы".

Вкладка "Архивы" (рис.4.6k) содержит таблицу с информацией об архивах, обрабатываемых архиватором. В строках таблица содержит архивы, а в колонках информацию:

- *Архив* - имя архива.
- *Период (с)* - периодичность архива в секундах.
- *Размер буфера* - размерность буфера в единицах.
- *Размер файлов (Мб)* - специфичное для модуля Arch.FSArch поле с информацией о суммарной размерности файлов хранилища архиватора для архива.

В случае с модулем Arch.FSArch в этой вкладке ещё содержится форма экспорта данных архиватора.

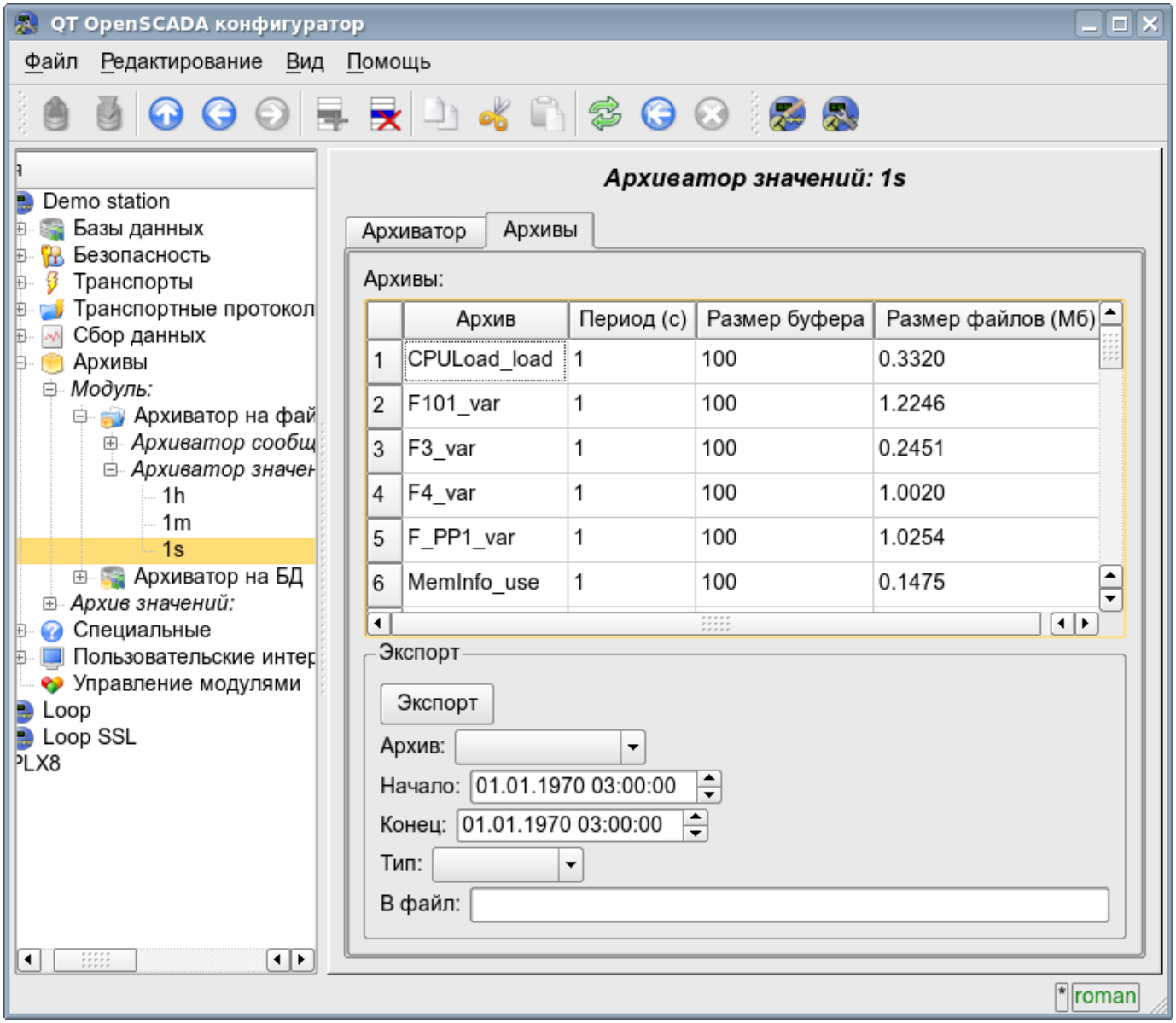


Рис. 4.6k. Вкладка "Архивы" архиватора значений подсистемы "Архивы".

4.7. Подсистема "Пользовательские интерфейсы"

Подсистема является модульной. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Пользовательские интерфейсы", содержащая вкладки "Модули" и "Помощь". Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждый модуль подсистемы "Пользовательские интерфейсы" предоставляет конфигурационную страницу с вкладками "Пользовательский интерфейс" и "Помощь". Вкладка "Пользовательский интерфейс" (рис.4.7a) предоставляет параметр для контроля за состоянием "Выполняется" модуля, а также разделы конфигурации специализированные для модулей этой подсистемы. Во вкладке "Помощь" содержится информация о модуле подсистемы "Пользовательские интерфейсы" (рис.4.1d), состав которой идентичен для всех модулей.

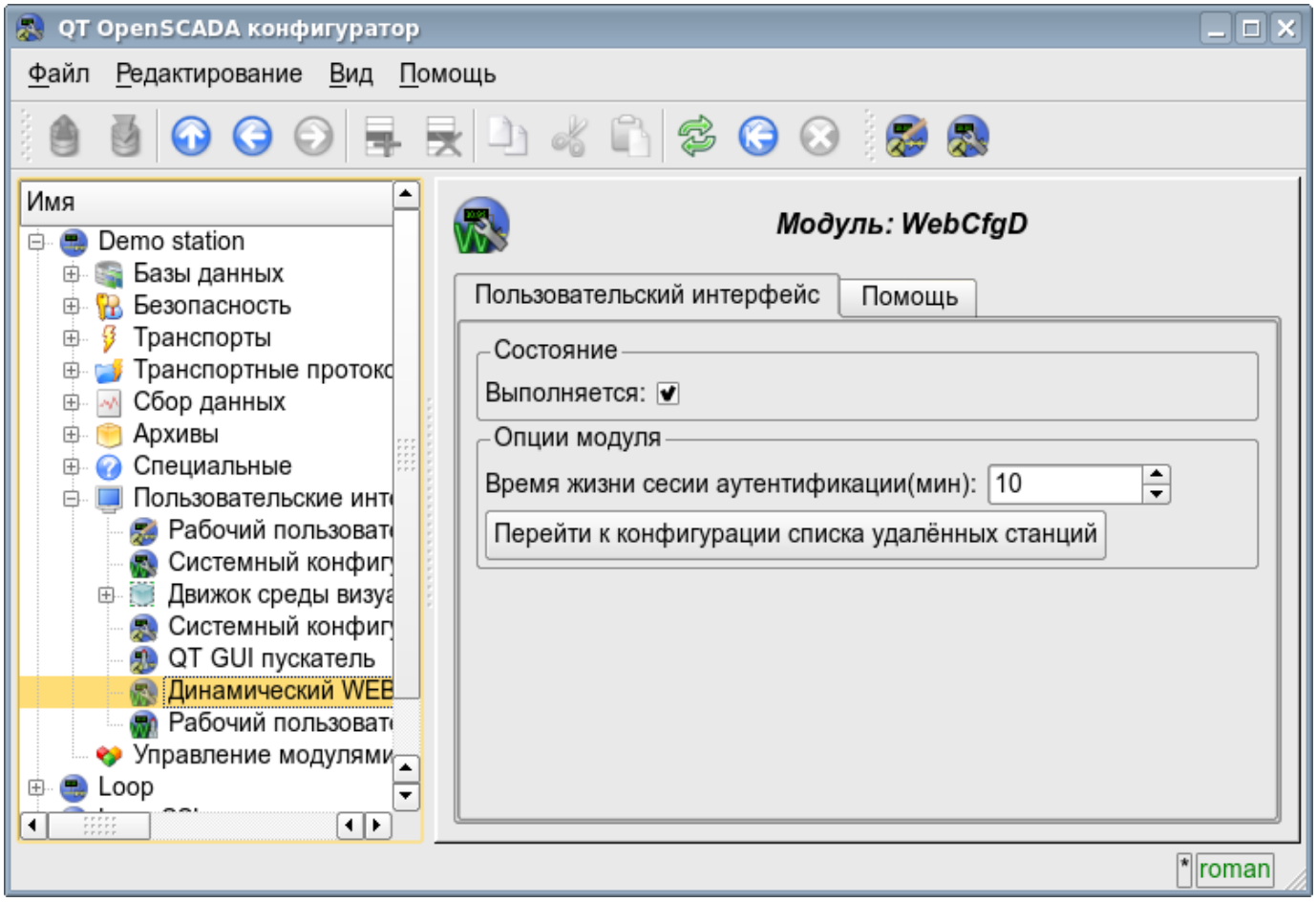


Рис. 4.7a. Вкладка "Пользовательский интерфейс" модуля подсистемы "Пользовательские интерфейсы".

4.8. Подсистема "Специальные"

Подсистема является модульной. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Специальные", содержащая вкладки "Модули" и "Помощь". Вкладка "Модули" (рис.4.1b) содержит список модулей подсистемы и идентична для всех модульных подсистем. Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждый модуль подсистемы "Специальные" предоставляет конфигурационную страницу с вкладками "Специальный модуль" и "Помощь". Вкладка "Специальный модуль" (рис.4.8a) предоставляет параметр для контроля за состоянием "Выполняется" модуля, а также разделы конфигурации специализированные для модулей этой подсистемы. Во вкладке "Помощь" содержится информация о модуле подсистемы "Специальные" (рис.4.1d), состав которой идентичен для всех модулей.

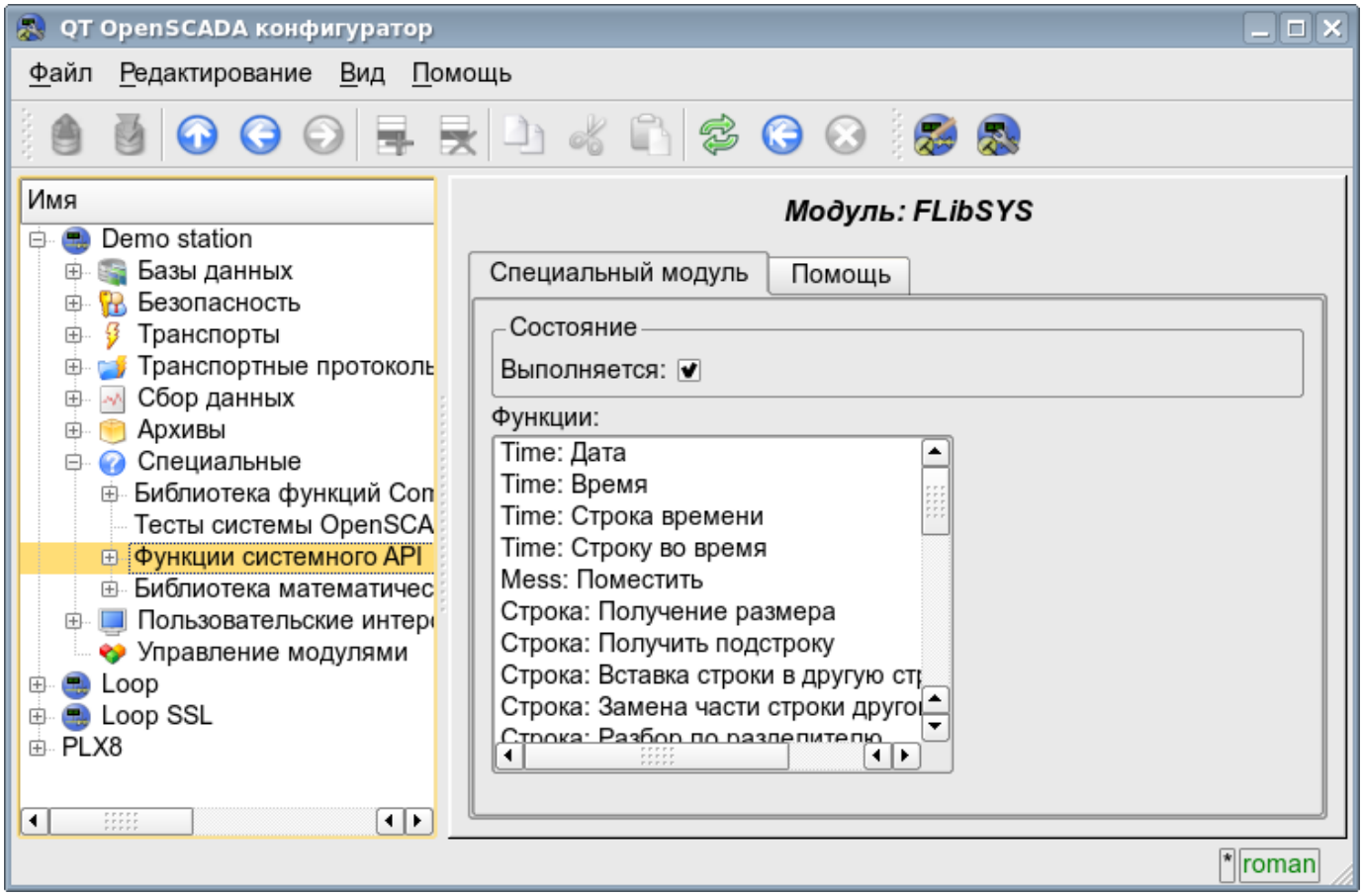


Рис. 4.8a. Вкладка "Специальный модуль" модуля подсистемы "Специальные".

4.9. Подсистема "Управление модулями"

Подсистема не является модульной. Для конфигурации подсистемы предусмотрена страница подсистемы "Управление модулями", содержащая вкладки "Подсистема" и "Помощь". Вкладка "Подсистема" (рис.4.9а) содержит основные настройки подсистемы. Вкладка "Помощь" содержит краткую помощь для данной страницы. Состав вкладки "Подсистема":

- *Путь к разделяемым библиотекам (модулям)* - информация о расположении директории с модулями системы OpenSCADA. Устанавливается параметром *<ModDir>* станции, конфигурационного файла.
- *Разрешённые модули* - информация о списке, разделённом символом ';', модулей, разрешённых для автоматического подключения и обновления. Значение '*' используется для разрешения всех модулей. Устанавливается параметром *<ModAllow>* раздела подсистемы, sub_ModSched, станции, конфигурационного файла.
- *Запрещённые модули* - информация о списке, разделённом символом ';', модулей, запрещённых для автоматического подключения и обновления. Устанавливается параметром *<ModDeny>* раздела подсистемы "sub_ModSched" станции конфигурационного файла. Список запрещённых модулей имеет больший приоритет чем разрешённых.
- *Период проверки модулей (сек)* - указывает на периодичность проверки модулей на факт их обновления. Модули, допустимые для автоматического подключения и обновления, будут автоматически обновлены.
- *Проверить модуль сейчас* - команда выполнить проверку модулей на факт их обновления. Модули, допустимые для автоматического подключения и обновления, будут автоматически обновлены.
- *Разделяемые библиотеки (модули)* - таблица с перечнем разделяемых библиотек с модулями, обнаруженными OpenSCADA. В строках расположены модули, а в колонках информация о них:
  - *Путь* - информация о полном пути к разделяемой библиотеке.
  - *Время* - информация о времени последней модификации файла разделяемой библиотеки.
  - *Модули* - информация о перечне модулей в разделяемой библиотеке.
  - *Включен* - состояние "Включен" разделяемой библиотеки. Привилегированным пользователям предоставляется возможность ручного включения/выключения разделяемых библиотек путём изменения этого поля.

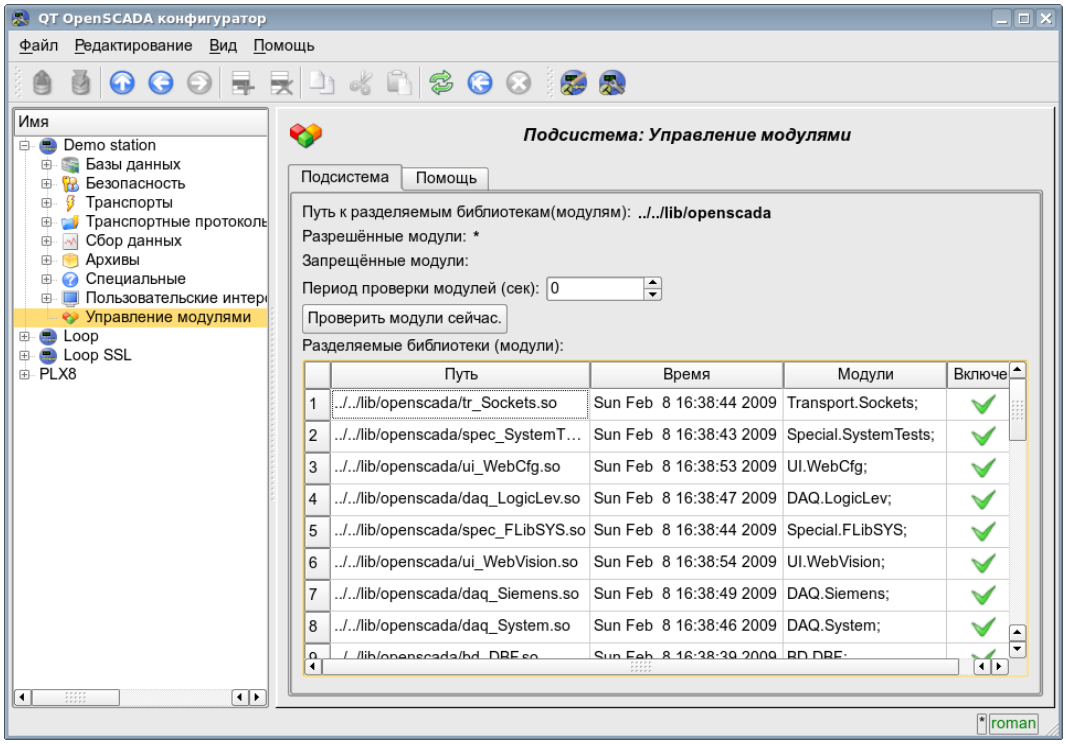


Рис. 4.9а. Главная вкладка конфигурации подсистемы "Управление модулями".

4.10. Конфигурационный файл OpenSCADA и параметры командной строки вызова OpenSCADA

Конфигурационный файл системы OpenSCADA предназначен для хранения системной и общей конфигурации OpenSCADA-станции. Только в конфигурационном файле и через параметры командной строки можно указать часть ключевых системных параметров станции, поэтому знакомство со структурой конфигурационного файла необходимо для специалистов развёртывающих решения на основе OpenSCADA.

Называться конфигурационный файл системы OpenSCADA может по разному, однако принято название oscada.xml и производные от него. Конфигурационный файл обычно указывается, при запуске станции, параметром командной строки *--Config=/home/roman/roman/work/OScadaD/etc/oscada_demo.xml*. Для удобства вызова создаются скрипты запуска станции с нужным конфигурационным файлом, например скрипт (openscada_demo) вызова демонстрационной станции:

```
#!/bin/sh
openscada --Config=/etc/oscada_demo.xml $@
```

Если конфигурационный файл не указан то используется стандартный конфигурационный файл: /etc/oscada.xml .

Структурно конфигурационный файл организован на расширяемом языке разметки текста XML. Следовательно требуется жёсткое соблюдение правил синтаксиса XML. Пример образца типового конфигурационного файла OpenSCADA, с узлами конфигурации большинства компонентов OpenASCADA, приведен ниже:

```
<?xml version="1.0" encoding="UTF-8" ?>
<OpenSCADA>
	<!-- This is the OpenSCADA configuration file. -->
	<station id="DemoStation">
		<!-- Discribe internal parameter for station. Station this only OpenSCADA programm. -->
		<prm id="StName">Demo station</prm>
		<prm id="StName_ru">Демо станция</prm>
		<prm id="StName_uk">Демо станція</prm>
		<prm id="WorkDB">SQLite.GenDB</prm>
		<prm id="Workdir">~/.openscada</prm>
		<prm id="IcoDir">./icons</prm>
		<prm id="ModDir">/usr/lib/openscada</prm>
		<prm id="LogTarget">10</prm>
		<prm id="MessLev">0</prm>
		<prm id="Lang2CodeBase">en</prm>
		<prm id="SaveAtExit">0</prm>
		<prm id="SavePeriod">0</prm>

		<node id="sub_BD">
			<prm id="SYSStPref">0</prm>
			<tbl id="DB">
				<fld ID="GenDB" TYPE="SQLite" NAME="Generic DB" NAME_ru="Основная БД"
					NAME_uk="Основна БД" ADDR="./DEMO/DemoSt.db" CODEPAGE="UTF-8"/>
			</tbl>
		</node>

		<node id="sub_Security">
			<!--
			<tbl id="Security_user">
				<fld
					NAME="root"
					DESCR="Super user"
					DESCR_ru="Супер пользователь"
					DESCR_uk="Супер користувач"
					PASS="openscada"/>
				<fld
					NAME="user"
					DESCR="System user"
					DESCR_ru="Системный пользователь"
					DESCR_uk="Системний користувач"
					PASS=""/>
			</tbl>
			<tbl id="Security_grp">
```

```
                <fld
                        NAME="root"
                        DESCR="Super users groups"
                        DESCR_ru="Группа суперпользователей"
                        DESCR_uk="Група суперкористувачів"
                        USERS="root;user"/>
        </tbl>-->
</node>

<node id="sub_ModSched">
        <prm id="ModAllow">*</prm>
        <prm id="ModDeny"></prm>
        <prm id="ChkPer">0</prm>
</node>

<node id="sub_Transport">
        <!--
        <tbl id="Transport_in">
                <fld
                        ID="WEB_1"
                        MODULE="Sockets"
                        NAME="Generic WEB interface"
                        NAME_ru="Основной WEB интерфейс"
                        NAME_uk="Основний WEB інтерфейс"
                        DESCRIPT="Generic transport for WEB interface."
                        DESCRIPT_ru="Основной транспорт для WEB интерфейса."
                        DESCRIPT_uk="Основний транспорт для WEB інтерфейсу."
                        ADDR="TCP::10002:0"
                        PROT="HTTP"
                        START="1"/>
                <fld
                        ID="WEB_2"
                        MODULE="Sockets"
                        NAME="Reserve WEB interface"
                        NAME_ru="Резервный WEB интерфейс"
                        NAME_uk="Резервний WEB інтерфейс"
                        DESCRIPT="Reserve transport for WEB interface."
                        DESCRIPT_ru="Резервный транспорт для WEB интерфейса."
                        DESCRIPT_uk="Резервний транспорт для WEB інтерфейсу."
                        ADDR="TCP::10004:0"
                        PROT="HTTP"
                        START="1"/>
        </tbl>
        <tbl id="Transport_out">
                <fld
                        ID="testModBus"
                        MODULE="Sockets"
                        NAME="Test ModBus"
                        NAME_ru="Тест ModBus"
                        NAME_uk="Тест ModBus"
                        DESCRIPT="Data exchange by protocol ModBus test."
                        DESCRIPT_ru="Тест обмена по протоколу ModBus."
                        DESCRIPT_uk="Тест обміну за протоколом ModBus."
                        ADDR="TCP:localhost:10502"
                        START="1"/>
        </tbl>-->
</node>

<node id="sub_DAQ">
        <!--
        <tbl id="tmplib">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" DB="tmplib_test2"/>
        </tbl>
        <tbl id="tmplib_test2">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" DB="test2"
                        PROGRAM="JavaLikeCalc.JavaScript&#010;cnt=5*i"/>
        </tbl>
        <tbl id="tmplib_test2_io">
                <fld TMPL_ID="test2" ID="i" NAME="I" NAME_ru="I" NAME_uk="I"
                        TYPE="4" FLAGS="160" VALUE="" POS="0"/>
                <fld TMPL_ID="test2" ID="cnt" NAME="Cnt" NAME_ru="Cnt" NAME_uk="Cnt"
                        TYPE="4" FLAGS="32" VALUE="" POS="0"/>
        </tbl>-->

        <node id="mod_LogicLev">
                <!--
                <tbl id="DAQ">
                        <fld
                                ID="test2"
                                NAME="Test 2"
```

```
                        NAME_ru="Тест 2"
                        NAME_uk="Тест 2"
                        DESCR=""
                        DESCR_ru=""
                        DESCR_uk=""
                        ENABLE="1"
                        START="1"
                        PRM_BD="test2prm"
                        PERIOD="1000"
                        PRIOR="0"/>
        </tbl>
        <tbl id="test2prm">
                <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" EN="1" MODE="2"
                        PRM="test2.test2"/>
        </tbl>-->
</node>

<node id="mod_System">
        <!--
        <tbl id="DAQ">
                <fld
                        ID="DataOS"
                        NAME="Data OS"
                        NAME_ru="Даные ОС"
                        NAME_uk="Дані ОС"
                        DESCR="Data of services and subsystems OS."
                        DESCR_ru="Данные сервисов и подсистем ОС."
                        DESCR_uk="Дані сервісів та підсистем ОС."
                        ENABLE="1"
                        START="1"
                        AUTO_FILL="0"
                        PRM_BD="DataOSprm"
                        PERIOD="1000" PRIOR="0"/>
        </tbl>
        <tbl id="DataOSprm">
                <fld SHIFR="CPU" NAME="CPU load" NAME_ru="Нагрузка CPU"
                        NAME_uk="Навантаження CPU" DESCR="" DESCR_ru="" DESCR_uk=""
                        EN="1" TYPE="CPU" SUBT="gen"/>
                <fld SHIFR="MEM" NAME="Memory" NAME_ru="Память" NAME_uk="Пам\'ять"
                        DESCR="" DESCR_ru="" DESCR_uk="" EN="1" TYPE="MEM"/>
        </tbl> -->
</node>

<node id="mod_DiamondBoards">
        <!--
        <tbl id="DAQ">
                <fld ID="Athena" NAME="Athena board" NAME_ru="Плата Athena"
                        NAME_uk="Плата Athena" DESCR="" DESCR_ru="" DESCR_uk=""
                        ENABLE="1" START="0" BOARD="25" PRM_BD_A="AthenaAnPrm"
                        PRM_BD_D="AthenaDigPrm" ADDR="640" INT="5" DIO_CFG="0"
                        ADMODE="0" ADRANGE="0" ADPOLAR="0" ADGAIN="0"
                        ADCONVRATE="1000"/>
        </tbl>
        <tbl id="AthenaAnPrm">
                <fld SHIFR="ai0" NAME="AI 0" NAME_ru="AI 0" NAME_uk="AI 0"
                        DESCR="" DESCR_ru="" DESCR_uk=""
                        EN="0" TYPE="0" CNL="0" GAIN="0"/>
        </tbl>
        <tbl id="AthenaDigPrm">
                <fld SHIFR="di0" NAME="DI 0" NAME_ru="DI 0" NAME_uk="DI 0"
                        DESCR="" DESCR_ru="" DESCR_uk=""
                        EN="0" TYPE="0" PORT="0" CNL="0"/>
        </tbl> -->
</node>

<node id="mod_BlockCalc">
        <!--
        <tbl id="DAQ">
                <fld ID="Model" NAME="Model" NAME_ru="Модель" NAME_uk="Модель"
                        DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                        PRM_BD="Model_prm" BLOCK_SH="Model_blcks"
                        PERIOD="1000" PRIOR="0" PER_DB="0" ITER="1"/>
        </tbl>
        <tbl id="Model_blcks">
                <fld ID="Klap" NAME="Klapan" NAME_ru="Клапан" NAME_uk="Клапан"
                        DESCR="" DESCR_ru="" DESCR_uk=""
                        FUNC="DAQ.JavaLikeCalc.lib_techApp.klap" EN="1" PROC="1"/>
        </tbl>
        <tbl id="Model_blcks_io">
                <fld BLK_ID="Klap" ID="l_kl1" TLNK="0" LNK="" VAL="50"/>
                <fld BLK_ID="Klap" ID="l_kl2" TLNK="0" LNK="" VAL="20"/>
```

```
        </tbl>
        <tbl id="Model_prm">
                <fld SHIFR="l_kl" NAME="Klap lev" NAME_ru="Полож. клапана"
                        NAME_uk="Полож. клапана" DESCR="" DESCR_ru="" DESCR_uk=""
                        EN="1" BLK="Klap" IO="l_kl1"/>
        </tbl> -->
</node>

<node id="mod_JavaLikeCalc">
        <!--
        <tbl id="DAQ">
                <fld ID="CalcTest" NAME="Calc Test" NAME_ru="Тест вычисл."
                        NAME_uk="Тест обчисл." DESCR="" DESCR_ru="" DESCR_uk=""
                        ENABLE="1" START="1" PRM_BD="Cal FUNC="TemplFunc.d_alarm"
                        PERIOD="1000" PRIOR="0" PER_DB="0" ITER="1"/>
        </tbl>
        <tbl id="CalcTest_val">
                <fld ID="in" VAL="0"/>
                <fld ID="alrm" VAL=""/>
                <fld ID="alrm_md" VAL="1"/>
                <fld ID="alrm_mess" VAL="Error present."/>
        </tbl>
        <tbl id="CalcTest_prm">
                <fld SHIFR="alrm" NAME="Alarm" NAME_ru="Авария" NAME_uk="Аварія"
                        DESCR="" DESCR_ru="" DESCR_uk="" EN="1" FLD="alrm"/>
        </tbl>
        <tbl id="lib">
                <fld ID="TemplFunc" NAME="" NAME_ru="" NAME_uk="" DESCR="" ESCR_ru=""
                        DESCR_uk="" DB="lib_TemplFunc"/>
        </tbl>
        <tbl id="lib_TemplFunc">
                <fld ID="d_alarm" NAME="Digit alarm" NAME_ru="Авария по дискр."
                        NAME_uk="Аварія за дискр" DESCR=""
                        FORMULA="alrm=(in==alrm_md)?&amp;quot;1:&amp;quot;
                                +alrm_mess:&amp;quot;0&amp;quot;;"/>
        </tbl>
        <tbl id="lib_TemplFunc_io">
                <fld F_ID="d_alarm" ID="in" NAME="Input" NAME_ru="Вход" NAME_uk="Вхід"
                        TYPE="3" MODE="0" DEF="" HIDE="0" POS="0"/>
                <fld F_ID="d_alarm" ID="alrm" NAME="Alarm" NAME_ru="Авария"
                        NAME_uk="Аварія" TYPE="0" MODE="1" DEF="" HIDE="0" POS="1"/>
                <fld F_ID="d_alarm" ID="alrm_md" NAME="Alarm mode"
                        NAME_ru="Режим аварии" NAME_uk="Режим аварії" TYPE="3"
                        MODE="0" DEF="" HIDE="0" POS="2"/>
                <fld F_ID="d_alarm" ID="alrm_mess" NAME="Alarm message"
                        NAME_ru="Сообщ. аварии" NAME_uk="Повід. аварії" TYPE="0"
                        MODE="0" DEF="" HIDE="0" POS="3"/>
        </tbl>-->
</node>

<node id="mod_Siemens">
        <!--
        <tbl id="DAQ">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                        PRM_BD="test2prm" PERIOD="1000" PRIOR="0" CIF_DEV="0" ADDR="5"
                        ASINC_WR="0"/>
        </tbl>
        <tbl id="test2prm">
                <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" EN="1" TMPL="S7.ai_man"/>
        </tbl>-->
</node>

<node id="mod_SNMP">
        <!--
        <tbl id="DAQ">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                        PRM_BD="test2prm" PERIOD="1000" PRIOR="0" ADDR="localhost"
                        COMM="public" PATTR_LIM="20"/>
        </tbl>
        <tbl id="test2prm">
                <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" EN="1" OID_LS="system"/>
        </tbl>-->
</node>

<node id="mod_ModBus">
        <!--
        <tbl id="DAQ">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
```

```
                                DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                                PRM_BD="test2prm" PERIOD="1000" PRIOR="0" TRANSP="Sockets"
                                ADDR="exlar.diya.org" NODE="1"/>
                        </tbl>
                        <tbl id="test2prm">
                                <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                DESCR="" DESCR_ru="" DESCR_uk=""
                                EN="1" ATTR_LS="321:0:tst:Test"/>
                        </tbl>-->
                </node>

                <node id="mod_Transporter">
                        <!--
                        <tbl id="DAQ">
                                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                                PRM_BD="test2prm" PERIOD="1000" PRIOR="0" SYNCPER="60"
                                STATIONS="loop" CNTRPRM="System.AutoDA"/>
                        </tbl>-->
                </node>
        </node>

        <node id="sub_Archive">
                <prm id="MessBufSize">1000</prm>
                <prm id="MessPeriod">5</prm>
                <prm id="ValPeriod">1000</prm>
                <prm id="ValPriority">10</prm>
                <!--
                <tbl id="Archive_mess_proc">
                        <fld
                                ID="StatErrors"
                                MODUL="FSArch"
                                NAME="Errors"
                                NAME_ru="Ошибки"
                                NAME_uk="Помилки"
                                DESCR="Local errors\' archive"
                                DESCR_ru="Архив локальных ошибок"
                                DESCR_uk="Архів локальних помилок"
                                START="1"
                                CATEG="/DemoStation*"
                                LEVEL="4"
                                ADDR="ARCHIVES/MESS/stError/"
                                FSArchMSize="300"
                                FSArchNFiles="10"
                                FSArchTmSize="30"
                                FSArchXML="1"
                                FSArchPackTm="10"
                                FSArchTm="60"/>
                        <fld
                                ID="NetRequsts"
                                MODUL="FSArch"
                                NAME="Net requests"
                                NAME_ru="Сетевые запросы"
                                NAME_uk="Мережеві запити"
                                DESCR="Requests to server through transport Sockets."
                                DESCR_ru="Запросы к серверу через транспорт Sockets."
                                DESCR_uk="Запити до сервера через транспорт Sockets."
                                START="1"
                                CATEG="/DemoStation/Transport/Sockets*"
                                LEVEL="1"
                                ADDR="ARCHIVES/MESS/Net/"
                                FSArchMSize="300"
                                FSArchNFiles="10"
                                FSArchTmSize="30"
                                FSArchXML="1"
                                FSArchPackTm="10"
                                FSArchTm="60"/>
                </tbl>
                <tbl id="Archive_val_proc">
                        <fld
                                ID="1h"
                                MODUL="FSArch"
                                NAME="1hour"
                                NAME_ru="1час"
                                NAME_uk="1год"
                                DESCR="Averaging for hour"
                                DESCR_ru="Усреднение за час"
                                DESCR_uk="Усереднення за годину"
                                START="1"
                                ADDR="ARCHIVES/VAL/1h/"
                                V_PER="360"
                                A_PER="60"
```

```
                        FSArchTmSize="8640"
                        FSArchNFiles="10"
                        FSArchRound="0.1"
                        FSArchPackTm="10"
                        FSArchTm="60"/>
        </tbl>
        <tbl id="Archive_val">
                <fld
                        ID="test1"
                        NAME="Test 1"
                        NAME_ru="Тест 1"
                        NAME_uk="Тест 1"
                        DESCR="Test 1"
                        DESCR_ru="Тест 1"
                        DESCR_uk="Тест 1"
                        START="1"
                        VTYPE="1"
                        BPER="1"
                        BSIZE="200"
                        BHGRD="1"
                        BHRES="0"
                        SrcMode="0"
                        Source=""
                        ArchS=""/>
        </tbl>-->
</node>

<node id="sub_Protocol">
</node>

<node id="sub_UI">
        <node id="mod_QTStarter">
                <prm id="StartMod">QTCfg</prm>
        </node>
        <node id="mod_WebCfg">
                <prm id="SessTimeLife">20</prm>
        </node>
        <node id="mod_VCAEngine">
                <!--
                <tbl id="LIB">
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                DESCR="" DESCR_ru="" DESCR_uk="" DB_TBL="wlib_test2" ICO=""
                                USER="root" GRP="UI" PERMIT="436"/>
                </tbl>
                <tbl id="wlib_test2">
                        <fld ID="test2" ICO="" PARENT="/wlb_originals/wdg_Box" PROC=""
                                PROC_ru="" PROC_uk="" PROC_PER="-1" USER="root" GRP="UI"
                                PERMIT="436"/>
                </tbl> <tbl id="wlib_test2_io">
                        <fld IDW="test2" ID="name" IO_VAL="Test 2" IO_VAL_ru="Тест 2"
                                IO_VAL_uk="Тест 2" SELF_FLG="" CFG_TMPL="" CFG_TMPL_ru=""
                                CFG_TMPL_uk="" CFG_VAL=""/>
                        <fld IDW="test2" ID="dscr" IO_VAL="Test module 2"
                                IO_VAL_ru="Тест модуля 2" IO_VAL_uk="Тест модуля 2"
                                SELF_FLG="" CFG_TMPL="" CFG_TMPL_ru="" CFG_TMPL_uk=""
                                CFG_VAL=""/>
                </tbl>
                <tbl id="PRJ">
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                DESCR="" DESCR_ru="" DESCR_uk="" DB_TBL="prj_test2" ICO=""
                                USER="root" GRP="UI" PER </tbl> <tbl id="prj_test2">
                        <fld OWNER="/test2" ID="pg1" ICO="" PARENT="/wlb_originals/wdg_Box"
                                PROC="" PROC_ru="" PROC_uk="" PROC_PER="-1" USER="root"
                                GRP="UI" PERMIT="436" FLGS="1"/>
                        <fld OWNER="/test2/pg1" ID="pg2" ICO=""
                                PARENT="/wlb_originals/wdg_Box" PROC="" PROC_ru="" PROC_uk=""
                                PROC_PER="-1" USER="root" GRP="UI" PERMIT="436" FLGS="0"/>
                </tbl>
                <tbl id="prj_test2_incl">
                        <fld IDW="/prj_test2/pg_pg1" ID="wdg1"
                                PARENT="/wlb_originals/wdg_Box"/>
                </tbl>-->
        </node>
</node>

<node id="sub_Special">
        <node id="mod_SystemTests">
                <prm id="PARAM" on="0" per="5" name="LogicLev.experiment.F3"/>
                <prm id="XML" on="0" per="10" file="/etc/oscada.xml"/> <prm id="MESS" on="0"
                        per="10" categ="" arhtor="DBArch.test3"/>
                <prm id="SOAttDet" on="0" per="20" name="../../lib/openscada/daq_LogicLev.so"
                        full="1"/>
```

```
                        <prm id="Val" on="0" per="1" name="LogicLev.experiment.F3.var" arch_len="5"
                          arch_per="1000000"/>
                        <prm id="Val" on="0" per="1" name="System.AutoDA.CPULoad.load" arch_len="10"
                          arch_per="1000000"/>
                        <prm id="BD" on="0" per="10" type="MySQL"
                          bd="server.diya.org;roman;123456;oscadaTest"
                          table="test" size="1000"/>
                        <prm id="BD" on="0" per="10" type="DBF" bd="./DATA/DBF" table="test.dbf"
                          size="1000"/>
                        <prm id="BD" on="0" per="10" type="SQLite" bd="./DATA/test.db" table="test"
                          size="1000"/>
                        <prm id="BD" on="0" per="10" type="FireBird"
                          bd="server.diya.org:/var/tmp/test.fdb;roman;123456"
                          table="test" size="1000"/>
                        <prm id="TrOut" on="0" per="1" addr="TCP:127.0.0.1:10001" type="Sockets"
                          req="time"/>
                        <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:10001" type="Sockets"
                          req="time"/>
                        <prm id="TrOut" on="0" per="1" addr="UNIX:./oscada" type="Sockets"
                          req="time"/>
                        <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:daytime" type="Sockets"
                          req="time"/>
                        <prm id="Func" on="0" per="10"/> <prm id="SysContrLang" on="0" per="10"
                          path="/Archive/FSArch/mess_StatErrors/%2fprm%2fst"/>
                        <prm id="ValBuf" on="0" per="5"/> <prm id="Archive" on="0" per="30"
                          arch="test1" period="1000000"/>
                        <prm id="Base64Code" on="0" per="10"/>
                    </node>
                </node>
            </station>
        </OpenSCADA>
```

Рассмотрим детальнее структуру конфигурационного файла. Один конфигурационный файл может содержать конфигурацию нескольких станций в секциях *<station id="DemoStation"/>*. Атрибутом указывается идентификатор станции. Использование той или иной секции станции, при вызове, указывается параметром командной строки *--Station=DemoStation*. Секция станции непосредственно содержит параметры станции и секции подсистем. Параметры конфигурации секции записываются в виде *<prm id="StName">Demo station</prm>*. Где в атрибуте *<id>* указывается идентификатор атрибута, а в теле тега указывается значение параметра "Demo station". Перечень доступных параметров и их описание для станции и всех остальных секций можно получить в консоли, посредством вызова OpenSCADA с параметром --help или во вкладках "Помощь" страниц компонентов конфигурационных файлов OpenSCADA (рис.4.10а).

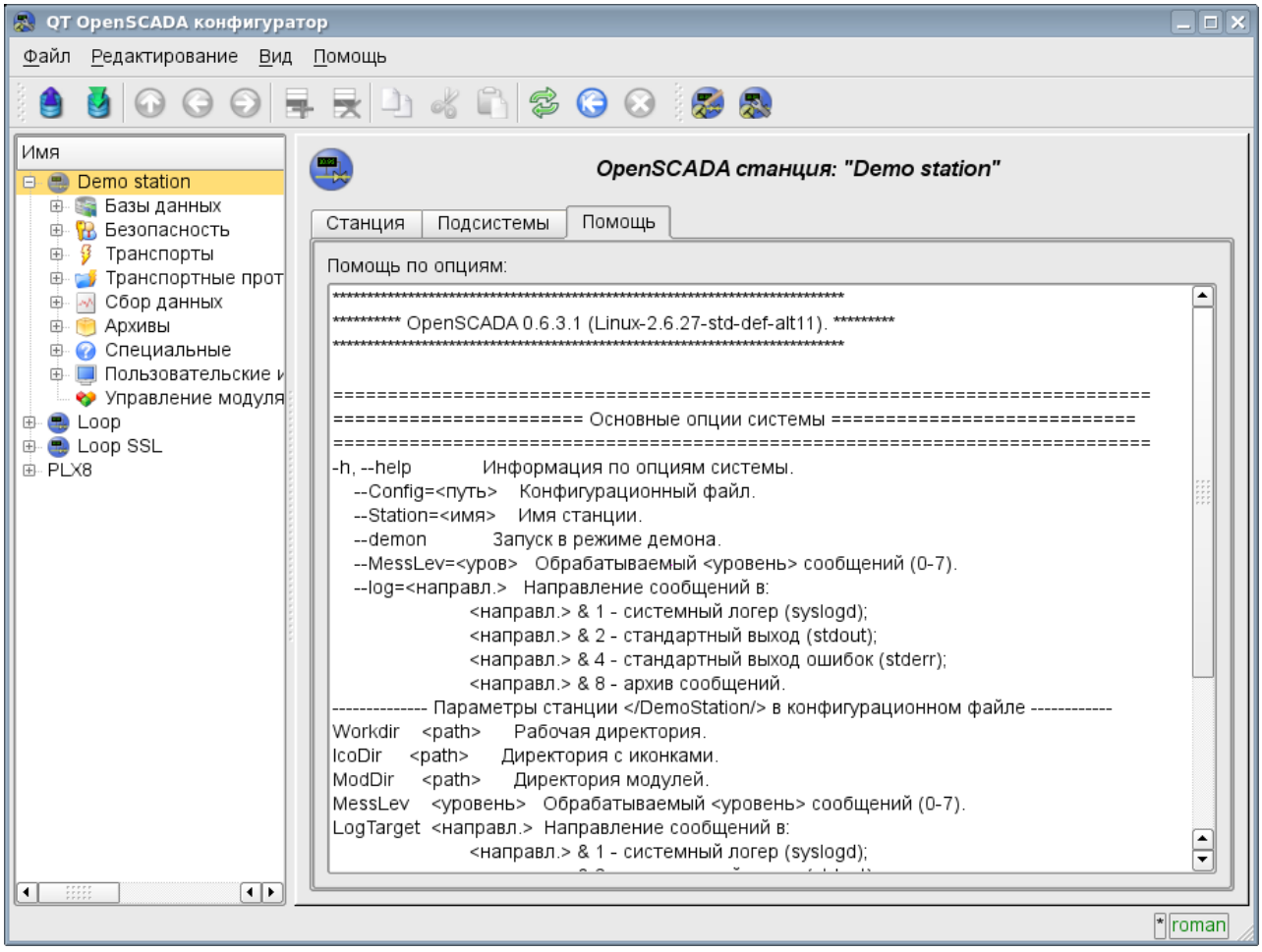


Рис. 4.10а. Вкладка "Помощь" компонента OpenSCADA.

Результат вызова команды: **# ./openscada_demo --help**

```
******************************************************************************
********** OpenSCADA 0.6.4.1 (Linux-2.6.30-std-def-alt15). *********
******************************************************************************
==============================================================================
====================== Основные опции системы ===============================
==============================================================================
-h, --help            Информация по опциям системы.
    --Config=<путь>   Конфигурационный файл.
    --Station=<имя>   Имя станции.
    --demon           Запуск в режиме демона.
    --MessLev=<уров>  Обрабатываемый <уровень> сообщений (0-7).
    --log=<направл.>  Направление сообщений в:
                      <направл.> & 1 - системный логер (syslogd);
                      <направл.> & 2 - стандартный выход (stdout);
                      <направл.> & 4 - стандартный выход ошибок (stderr);
                      <направл.> & 8 - архив сообщений.
-------------- Параметры станции </EmptySt/> в конфигурационном файле ------------
StName     <имя>       Имя станции.
WorkDB     <Тип.Имя>   Рабочая БД (тип и имя).
Workdir    <path>      Рабочая директория.
IcoDir     <path>      Директория с иконками.
ModDir     <path>      Директория модулей.
MessLev    <уровень>   Обрабатываемый <уровень> сообщений (0-7).
LogTarget <направл.>   Направление сообщений в:
                       <направл.> & 1 - системный логер (syslogd);
                       <направл.> & 2 - стандартный выход (stdout);
                       <направл.> & 4 - стандартный выход ошибок (stderr);
                       <направл.> & 8 - архив сообщений.
Lang2CodeBase <язык>   Базовый язык для перевода текстовых переменных, два символа.
SaveAtExit <true>      Сохранять систему при выходе.
SavePeriod <сек>       Период сохранения системы.

=================== Подсистема "Управление модулями" ====================
    --ModPath=<путь> Путь к модулям (/var/os/modules/).
--------- Параметры секции </DemoStation/sub_ModSched/> в конфигурационном файле ---------
ModPath    <путь>      Путь к разделяемым библиотекам (модулям).
ModAllow   <список>    Список разделяемых библиотек допустимых для автоматической загрузки, подключения и
                       запуска (bd_DBF.so;daq_JavaLikeCalc.so).
                       Использовать значение '*' для разрешения всех модулей.
ModDeny    <список>    Список разделяемых библиотек запрещённых для автоматической загрузки, подключения и
                       запуска (bd_DBF.so;daq_JavaLikeCalc.so).
ChkPer     <сек>       Период поиска новых разделяемых библиотек (модулей).

========================== Опции подсистемы "БД" ==========================
-------------- Параметры станции </DemoStation/sub_BD/> в конфигурационном файле ------------
SYSStPref    <1>   Использовать идентификатор станции в общей (SYS) таблице.

==================== Опции подсистемы "Безопасности" ====================

====================== Опции подсистемы "Транспорты" ====================

============== Опции подсистемы "Транспортные протоколы" ================

======================= Опции модуля <Protocol:HTTP> ============================
------- Параметры модульной секции </DemoStation/sub_Protocol/mod_HTTP/> в конфигурационном файле ---------
AuthTime  <мин>        Время жизни аутентификации, минут (по умолчанию 10).

=================== Опции подсистемы "Сбор данных" ======================
--------------- Параметры секции </DemoStation/sub_DAQ/> в конфигурационном файле --------------
RdStLevel  <уров>      Уровень текущей станции в схеме резервирования.
RdTaskPer  <с>         Периодичность вызова задачи обслуживания резервирования.
RdRestConnTm <с>       Интервал времени восстановления соединения с "мёртвой" резервной станцией.
RdRestDtTm <час>       Глубина восстановления данных архива из резервной станции, при включении.
RdStList   <список>    Список резервных станций, разделённых символом ';' (st1;st2).

==================== Опции подсистемы "Архивы" ======================
---------- Параметры секции </DemoStation/sub_Archive/> в конфигурационном файле --------
MessBufSize <ед.>      Размер буфера сообщений.
MessPeriod <сек>       Период архивирования сообщений.
ValPeriod <мсек>       Период архивирования значений.
ValPriority <уровень> Уровень приоритета задачи значений.
MaxReqMess <ед.>       Максимальное количество запрашиваемых сообщений.
MaxReqVals <ед.>       Максимальное количество запрашиваемых значений.

==================== Опции подсистемы "Специальные" ====================

========================= Опции модуля <Special:SystemTests> =========================
--------  Параметры  модульной  секции  </DemoStation/sub_Special/mod_SystemTests/>  в  конфигурационном  файле
--------
Общие опции всех тестов:
```

```
  id           идентификатор теста;
  on           флаг включения теста;
  per          период повторения (сек).
      *** Опции тестов ***
1) Param        Тест DAQ параметров. Вычитывает атрибуты и конфигурационные поля параметра.
  1:name        Адрес DAQ параметра
2) XML  Тест разбора файла XML. Парсит и отображает структуру указанного файла.
  1:file        XML файл
3) Mess Тест архива сообщений. Периодически вычитывает новые сообщения из архива, для указанного архиватора.
  1:arhtor      Архиватор
  2:categ       Шаблон категории сообщения
  3:depth       Глубина сообщения (с)
4) SOAttach     Тест подключения/отключения модулей.
  1:name        Путь к модулю
  2:mode        Режим (1-подключ.;-1-отключ.;0-изменение)
  3:full        Полное подключение(при старте)
5) Val  Тест значений атрибута параметра.
Выполняет периодический опрос последнего значения указанного атрибута, а также опрос архива на указанную
глубину.
  1:name        Путь к атрибуту параметра
  2:arch_len    Глубина запроса к архиву значений (с)
  3:arch_per    Период запроса к архиву значений (мкс)
6) DB   Полный тест БД. Выполняет:
  - создание/открытие БД;
  - создание/открытие таблицы;
  - создание множества записей (строк) предопределённой структуры;
  - модификация множества записей;
  - получение и проверка значений множества записей;
  - модификация структуры записи и таблицы;
  - удаление записей;
  - закрытие/удаление таблицы;
  - закрытие/удаление БД.
  1:type        Тип БД
  2:addr        Адрес БД
  3:table       Таблица БД
  4:size        Количество записей
7) TrOut        Тест выходных и/или входных транспортов.
Выполняет тестирование исходящего транспорта путём отправления запроса к указанному входящему транспорту.
  1:addr        Адрес
  2:type        Модуль транспорта
  3:req Текст запроса
8) SysContrLang Тест языка управления системой.
Производит запрос элементов языка посредством полного пути.
Полный путь к элементу языка имеет вид </Archive/%2fbd%2fm_per>.
Полный путь состоит из двух вложенных путей.
Первый </d_Archive/> это путь к узлу дерева контроля.
Второй </bd/m_per> это путь к конкретному элементу узла.
  1:path        Путь к элементу языка
9) ValBuf           Тесты буфера значений. Содержит 13 тестов всех аспектов буфера значений (подсистема
"Архивы").
10) Archive     Тесты размещения в архиве значений.
Содержит 7(8) тестов архиватора значений на проверку корректности функционирования последовательного
механизма упаковки.
  1:arch        Архив значений
  2:period      Период значений (мкс)
11) Base64Code  Тесты кодирования Mime Base64 алгоритмом.

============= Опции подсистемы "Пользовательские интерфейсы" ============

======================== Опции модуля <UI:Vision> ============================
------- Параметры модульной секции </DemoStation/sub_UI/mod_Vision/> в конфигурационном файле ---------
StartUser    <польз>  Стартовый, безпарольный, пользователь.
RunPrjs      <список> Перечень запускаемых при старте проектов.
RunTimeUpdt  <mode>   Режим обновления динамики в RunTime (0 - адаптивное периодическое обновление всех
                      виджетов, 1 - обновление только изменённых виджетов).
VCAstation   <id>     Станция с движком СВУ ('.' - локальная).

======================== Опции модуля <UI:VCAEngine> ============================
    --VCADBClearForce Принудительная очистка БД СВУ от данных API 1.

======================== Опции модуля <UI:QTCfg> ============================
------- Параметры модульной секции </DemoStation/sub_UI/mod_QTCfg/> в конфигурационном файле ---------
StartPath    <path>   Стартовый путь конфигуратора.
StartUser    <user>   Стартовый, безпарольный, пользователь.

======================== Опции модуля <UI:QTStarter> ============================
------- Параметры модульной секции </DemoStation/sub_UI/mod_QTStarter/> в конфигурационном файле ---------
StartMod     <модули> Список запускаемых модулей (разделитель - ';');

======================== Опции модуля <UI:WebVision> ============================
------- Параметры модульной секции </DemoStation/sub_UI/mod_WebVision/> в конфигурационном файле ---------
SessTimeLife <время>  Время жизни сессии, минуты (по умолчанию 10).
```

Секции подсистем (<*node id="sub_DAQ" />*) содержат параметры подсистемы, секции модулей и секции таблиц отражения данных баз данных в конфигурационном файле. Секции модулей (<*node id="mod_DiamondBoards" />*) содержат индивидуальные параметры модулей и секции таблиц отражения данных баз данных в конфигурационном файле.

Секции таблиц отражения данных баз данных предназначены для размещения в конфигурационном файле записей таблиц БД для компонентов OpenSCADA. Рассмотрим таблицу входящих транспортов "Transport_in" подсистемы транспорты (<*node id="sub_Transport"*>) из примера конфигурационного файла выше. Таблица содержит две записи с полями: ID, MODULE, NAME, DESCRIPT, ADDR, PROT, START. После загрузки с такой секцией и вообще без БД в подсистеме "Транспорты" модуля "Sockets" появятся два входных транспорта. Форматы структур таблиц основных компонентов включены в демонстрационные конфигурационные файлы. За деталями структуры БД нужно обращаться к документации соответствующих модулей.

5. Общесистемное API пользовательского программирования.

API пользовательского программирования представляет собой дерево объектов системы OpenSCADA, каждый объект которого может представлять собственный перечень свойств и функций. Свойства и функции объектов могут использоваться пользователем в процедурах на языках пользовательского программирования OpenSCADA. Точкой входа для доступа к объектам системы OpenSCADA из языка пользовательского программирования JavaLikeCalc является зарезервированное слово "SYS" корневого объекта OpenSCADA. Например, для доступа к функции исходящего транспорта нужно записать: **SYS.Transport.Serial.out_ModBus.messIO(mess);**.

API объектов, предоставляемых модулями, описывается в собственной документации модуля.

5.1. Общесистемные пользовательские объекты.

Абстрактный объект представляет собой ассоциативный контейнер свойств и функций. Свойства могут содержать как данные четырёх базовых типов, так и другие объекты. Доступ к свойствам объекта обычно осуществляется посредством записи имён свойств через точку к объекту *<obj.prop>*, а также посредством заключения имени свойства в квадратные скобки *<obj["prop"]>*. Очевидно, что первый механизм статичен, а второй позволяет указывать имя свойства через переменную. Базовое определение объекта не содержит функций. Операции копирования объекта на самом деле делают ссылку на исходный объект. При удалении объекта осуществляется уменьшения счётчика ссылок, а при достижении счётчика ссылок нуля объект удаляется физически.

Разные компоненты могут доопределять базовый объект особыми свойствами и функциями. Стандартным расширением объекта является массив "Array".

Объект Array

Особенностью массива является то, что он работает со свойствами, как с индексами, и полное их именование бессмысленно, а значит доступен механизм обращения только заключением индекса в квадратные скобки *<arr[1]>*. Массив хранит свойства в собственном контейнере одномерного массива.

Массив предоставляет специальное свойство "length" для получения размера массива *<var = arr.length;>*. Также массив предоставляет следующие функции:

- *string join(string sep = ",")*, *string toString(string sep = ",")*, *string valueOf(string sep = ",")* - Возвращает строку с элементами массива разделёнными *<sep>* или символом ','.
- *Array concat(Array arr);* - Добавляет к исходному массиву элементы массива *<arr>*. Возвращает исходный массив с изменениями.
- *int push(ElTp var, ...);* - Помещает элемент(ы) *<var>* в конец массива, как в стек. Возвращает новый размер массива.
- *ElTp pop( );* - Удаление последнего элемента массива и возврат его значения, как из стека.
- *Array reverse( );* - Изменение порядка расположения элементов массива. Возвращается исходный массив с изменениями.
- *ElTp shift( );* - Сдвиг массива в верх. При этом первый элемент массива удаляется, а его значение возвращается.
- *int unshift(ElTp var, ...);* - Задвигает элемент(ы) *<var>* в массив. Первый элемент в 0, второй в 1 и т.д.
- *Array slice(int beg, int end);* - Возвращает фрагмент массива от *<beg>* к *<end>*. Если значение начала или конца отрицательно, то отсчёт ведётся с конца массива. Если конец не указан, то концом является конец массива.
- *Array splice(int beg, int remN, ElTp val1, ElTp val2, ...);* - Вставляет, удаляет или заменяет елементы массива. Возвращает исходный массив с изменениями. В первую очередь осуществляется удаление элементов с позиции *<beg>* и количеством *<remN>*, а затем вставляются значения *<val1>* и т.д., начиная с позиции *<beg>*.
- *Array sort( );* - Сортировка элементов массива в лексикографическом порядке.

Объект XMLNodeObj

Функции:

- *string name( )* - Имя узла, XML-тега.
- *string text( )* - Текст узла, содержимое XML-тега.
- *string attr(string id)* - Значение атрибута узла <*id*>.
- *XMLNodeObj setName(string vl)* - Установка имени узла в <*vl*>. Возвращает текущий узел.
- *XMLNodeObj setText(string vl)* - Установка текста узла в <*vl*>. Возвращает текущий узел.
- *XMLNodeObj setAttr(string id, string vl)* - Установка атрибута <*id*> в значение <*vl*>. Возвращает текущий узел.
- *int childSize( )* - Количество вложенных узлов.
- *XMLNodeObj childAdd(ElTp no = XMLNodeObj)* - Добавление объекта <*no*> как вложенного. <*no*> может быть как непосредственно объектом-результатом функции *xmlNode()*, так и строкой с именем нового тега. Возвращается вложенный узел.
- *XMLNodeObj childIns(int id, ElTp no = XMLNodeObj)* - Вставка объекта <*no*> как вложенного в позицию <*id*>. <*no*> может быть как непосредственно объектом-результатом функции *xmlNode()*, так и строкой с именем нового тега. Возвращается вложенный узел.
- *XMLNodeObj childDel(int id)* - Удаление вложенного узла в позиции <*id*>. Возвращает текущий узел.
- *XMLNodeObj childGet(int id)* - Получение вложенного узла в позиции <*id*>.
- *string load(string str, bool file = false)* - Загрузка XML из строки <*str*> или из файла с путём в <*str*> если <*file*> "true".
- *string save(int opt = 0, string path = "")* - Сохранение дерева XML в строку или в файл <*path*> с параметрами форматирования <*opt*>. Возвращает текст XML или код ошибки. Предусмотрены следующие опции форматирования <*opt*>:
  - 0x01 - прерывать строку перед открывающим тегом;
  - 0x02 - прерывать строку после открывающего тега;
  - 0x04 - прерывать строку после закрывающего тега;
  - 0x08 - прерывать строку после текста;
  - 0x10 - прерывать строку после инструкции;
  - 0x1E - прерывать строку после всех.

5.2. Система (SYS)

Функции объекта:

- *string system(string cmd, bool noPipe = false);* - осуществляет вызов консольных команд <*cmd*> ОС с возвратом результата по каналу. Если <*noPipe*> установлен то возвращается код возврата вызова и возможен запуск программ в фоне ("sleep 5 &"). Функция открывает широкие возможности пользователю OpenSCADA путём вызова любых системных программ, утилит и скриптов, а также получения посредством них доступа к огромному объёму системных данных. Например команда "ls -l" вернёт детализированное содержимое рабочей директории.
- *int message(string cat, int level, string mess);* - формирование системного сообщения <*mess*> с категорией <*cat*>, уровнем <*level*>. Отрицательное значение уровня формирует нарушения (Alarm).
- *int messDebug(string cat, string mess); int messInfo(string cat, string mess); int messNote(string cat, string mess); int messWarning(string cat, string mess); int messErr(string cat, string mess); int messCrit(string cat, string mess); int messAlert(string cat, string mess); int messEmerg(string cat, string mess);* - формирование системного сообщения <*mess*> с категорией <*cat*> и соответствующим уровнем.
- *XMLNodeObj XMLNode(string name = "");* - создание объекта узла XML с именем <*name*>.
- *string cntrReq(XMLNodeObj req, string stat = "");* - запрос интерфейса управления к системе посредством XML. Обычный запрос записывается в виде <get path="/OPath/%2felem"/>. При указании станции осуществляется запрос к внешней станции.
- *int time(int usec);* - возвращает абсолютное время в секундах от эпохи 1.1.1970 и

микросекундах, если <*usec*> указан.

- *int localtime(int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int yday, int isdst);* - возвращает полную дату в секундах (sec), минутах (min), часах (hour), днях месяца (mday), месяце (month), годе (year), днях недели (wday), днях в году (yday) и признак летнего времени (isdst), исходя из абсолютного времени в секундах <*fullsec*> от эпохи 1.1.1970.
- *string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S");* - Преобразует абсолютное время <*sec*> в строку нужного формата <*form*>. Запись формата соответствует POSIX-функции strftime.
- *int strptime(int str, string form = "%Y-%m-%d %H:%M:%S");* - Возвращает время в секундах от эпохи 1.1.1970, исходя из строковой записи времени <*str*>, в соответствии с указанным шаблоном <*form*>. Например, шаблону "%Y-%m-%d %H:%M:%S" соответствует время "2006-08-08 11:21:55". Описание формата шаблона можно получить из документации на POSIX-функцию "strptime".
- *int cron(string cronreq, int base = 0);* - Возвращает время, спланированное в формате стандарта Cron <*cronreq*>, начиная от базового времени <*base*> или от текущего, если базовое не указано.
- *string strFromCharCode(int char1, int char2, int char3, ...);* - Создание строки из кодов символов char1, char2 ... charN.

5.3. Любой объект дерева OpenSCADA (SYS.*)

Функции объекта:

- *TArrayObj nodeList(string grp = "", string path = "");* - Получение списка дочерних узлов для группы <*grp*> и узла по пути <*path*>. Если <*grp*> пуста то возвращаются узлы всех групп.
- *TCntrNodeObj nodeAt(string path, string sep="");* - Подключение к узлу <*path*> в дереве объектов OpenSCADA. Если указывается разделитель в <*sep*> то путь обрабатывается как строка с разделителем.

5.4. Подсистема "БД" (SYS.BD)

Функции объекта БД (SYS.BD["TypeDB"]["DB"]):

- *Array SQLReq(string req);* - Формирование SQL-запроса к БД.

Пример:

```
DBTbl=SYS.BD.MySQL.GenDB.SQLReq("SELECT * from DB;");
for( var i_rw = 0; i_rw < DBTbl.length; i_rw++ )
{
      var rec = "";
      for( var i_fld = 0; i_fld < DBTbl[i_rw].length; i_fld++ )
            rec += DBTbl[i_rw][i_fld]+"\t";
      SYS.messDebug("TEST DB","Row "+i_rw+": "+rec);
}
```

5.5. Подсистема "Сбор данных" (SYS.DAQ)

Функции объекта атрибута параметра контроллера (SYS.DAQ["Modul"]["Controller"]["Parameter"]["Attribute"]):

- *ElTp get(int tm = 0, int utm = 0, bool sys = false);* - запрос значения атрибута на время <*tm:utm*> и признаком системного доступа <*sys*>.
- *bool set(ElTp val, int tm = 0, int utm = 0, bool sys = false);* - запись значения <*val*> в атрибут с меткой времени <*tm:utm*> и признаком системного доступа <*sys*>.

5.6. Подсистема "Архивы" (SYS.Archive)

Функции объекта подсистемы:

- *Area messGet(int btm, int etm, string cat = , int lev = 0, string arch =);* - запрос системных

сообщений за время от <*btm*> до <*etm*> для категории <*cat*>, уровня <*lev*> и архиватора <*arch*>.

5.7. Подсистема "Транспорты" (SYS.Transport)

Функции объекта исходящего транспорта (SYS.Transport["Modul"]["OutTransp"]):

- *string messIO(string mess, real timeOut = 1000);* - отправка сообщения <*mess*> через транспорт с таймаутом ожидания <*timeOut*>.
- *int messIO(XMLNodeObj req, string prt);* - отправка запроса <*req*> к протоколу <*prt*> для осуществления сеанса связи через транспорт посредством протокола.

Сбор данных в OpenSCADA.

Сбор данных SCADA(Supervisory Control and Data Acquisition)-системы является её неотъемлемой частью, которая занимается получением данных из источников различного происхождения. Природа данных, с которыми работает SCADA, характеризуется сигналами базовых типов значений (целое, вещественное, логическое и строка). Сигналы изменяются во времени и обладает историей, жизнью. В теории управления технологическими процессами (ТП) под сигналом понимается значение датчика установки ТП в коде АЦП, "сырой" сигнал или в реальном значении. Сигналы могут объединяться в группы по смысловой нагрузке, часто называемые параметрами. Например, развитые источники данных могут предоставлять структуры параметров с предопределённым набором связанных сигналов. Кроме непосредственного сбора данных в функции этого механизма также входит и передача воздействий на исполнительные устройства управления ТП; обычно это задвижки, насосы и регулирующие клапана. В совокупности этот процесс получил название Устройство Сопряжения с Объектом (УСО).

Источники данных характеризуются большим разнообразием, которое можно условно разделить на три группы.

- Источники "сырых" данных, предоставляющие код АЦП или уровни дискретных сигналов, а также включающие простейшую обработку. Обычно это модули рассредоточенного УСО или простейшие промышленные программируемые логические контроллеры (ПЛК).
- Мощные промышленные ПЛК, обладающие значительной вычислительной мощностью и возможностью формирования сложных параметров с различной структурой.
- Локальные или сопутствующие источники данных. Например, УСО в виде плат расширения, а также данные аппаратного и программного окружения, в котором функционирует система.

Разнообразие источников данных породило большой спектр механизмов доступа к ним. Локальные источники данных различаются интерфейсами программирования приложения (API), а сетевые источники, в свою очередь, транспортным и протокольным уровнями взаимодействия. В целом это привело к тому, что добавление поддержки нового источника данных требует создание модуля сопряжения или драйвера. Учитывая же большое разнообразие источников, это крайне накладно, и фактически нереально охватить весь спектр рынка таких устройств. Ситуация несколько упрощается с сетевыми источниками благодаря наличию ряда стандартных и свободных протоколов взаимодействия, однако многие источники всё же используют собственные протоколы: закрытые коммерческие или протоколы, завязанные на закрытые механизмы ограниченного круга коммерческих операционных систем (ОС).

В терминах системы OpenSCADA предоставляются следующие объекты для обслуживания механизма сбора данных:

- Атрибут — объект отражения данных сигнала, включает текущее значение с типом сигнала и историю изменения значений;
- Параметр — объект группы атрибутов (сигналов) со структурой, соответствующей особенностям отдельно взятого источника данных;
- Контроллер — объект отдельного устройства данных. Как правило, это отдельный модуль УСО или устройство промышленного ПЛК.

Для учёта особенностей различных устройств сбора данных, а также различных механизмов взаимодействия в OpenSCADA предусмотрена подсистема "Сбор данных", которая является модульной. В качестве модуля подсистемы выступает драйвер для сопряжения с источником данных отдельного типа. Каждый модуль может содержать конфигурацию нескольких устройств этого типа в виде объектов "Контроллер" системы OpenSCADA. Общая схема объектов подсистемы "Сбор данных" изображена на рисунке 1.

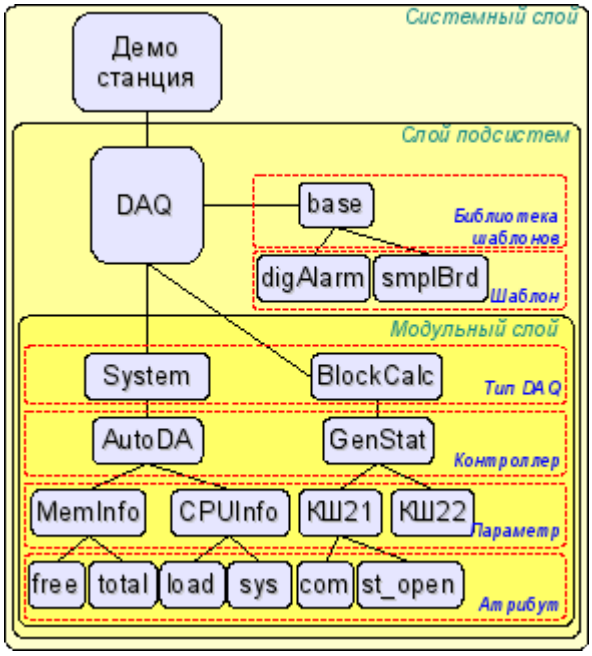


Рис. 1. Схема подсистемы "Сбор данных".

1. Методы сбора данных

Учитывая различие свойств источников данных, а также возможные варианты взаимодействия, методы сбора данных можно разделить на простой синхронный, простой асинхронный, пакетный и пассивный.

В рассмотрении механизмов ниже будут участвовать следующие объекты:

- ObjectSCADA — любой объект SCADA-системы, обращающийся за значением сигнала; например, архивы и визуализаторы;
- DAQParamAttribute — атрибут параметра подсистемы "Сбор данных", выступающий посредником в доступе к значению сигнала источника данных;
- DAQParamAttributeArch — объект архива атрибута;
- HardwarePLC — объект источника данных, например, модули рассосредоточенного УСО или промышленные ПЛК.

1.1. Простой синхронный механизм сбора

Механизм характеризуется запросами к источнику данных синхронно с запросом к атрибуту параметра (рис.2). Данный механизм обычно применяется при работе с локальными источниками данных, характеризующимися низкой латентностью, т.е. задержкой в ответе на запрос. С помощью этого метода можно получить актуальные данные непосредственно с запросом, однако время запроса объекта будет включать время транспортировки и обработки запроса источником данных.

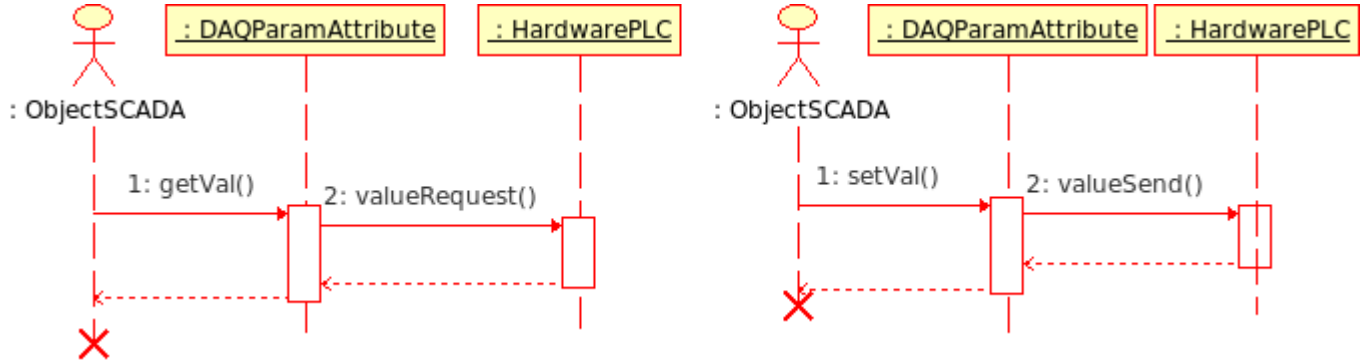


Рис. 2. Диаграмма последовательности взаимодействия при синхронных запросах.

В соответствии с диаграммой выше мы получаем следующую последовательность запросов получения данных и их передачи:

- объект SCADA-системы шлёт запрос значения к объекту атрибута параметра DAQParamAttribute::getVal();
- объект атрибута параметра, получив запрос, шлёт его источнику данных HardwarePLC::valueRequest();
- источник данных, обработав запрос, возвращает результат;
- объект атрибута параметра, получив результат, возвращает его объекту SCADA-системы.

В OpenSCADA такой механизм реализуют следующие модули подсистемы "Сбор данных".

- *ModBus* — модуль доступа к данным источников посредством семейства протоколов ModBus. В модуле реализован синхронный режим для записи данных.
- *DiamondBoards* — модуль доступа к данным PC/104 плат фирмы Diamond Systems. Платы PC/104 размещаются на ISA-шине, следовательно являются локальными и доступны сравнительно быстро. В режиме сбора данных не по прерыванию доступ к значениям АЦП осуществляется синхронно. Режим записи значения ЦАП всегда работает синхронно.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. В модуле реализован синхронный режим для записи данных.
- *BlockCalc* — вычислитель на языке блочных схем. В качестве источника данных в нём выступает пользовательская блочная схема. Атрибуты параметров модуля синхронно обращаются к входам/выходам блоков блочной схемы.
- *JavaLikeCalc* — вычислитель на Java-подобном языке высокого уровня. В качестве

источника данных в нём выступает пользовательская программа на Java-подобном языке. Атрибуты параметров модуля синхронно обращаются к входам/выходам пользовательской вычислительной функции.

• *LogicLev* — модуль логического уровня параметров сбора данных, детальнее о нём в разделе 2. В качестве источника данных этого модуля выступают другие параметры подсистемы "Сбор данных" и контекст исполнения шаблона параметров. Атрибуты параметров модуля синхронно обращаются к атрибутам других параметров в режиме отражения параметров подсистемы "Сбор данных" или к входам/выходам контекста исполнения шаблона в режиме работы по шаблону.

1.2. Простой асинхронный механизм сбора

Механизм характеризуется запросами к источнику данных независимо от запроса к атрибуту параметра (рис.3). Обычно запросы к источнику данных осуществляются периодически в собственной задаче опроса отдельно взятого контроллера и блоками по несколько сигналов. При этом запросом к атрибуту параметра возвращается значение, полученное последним сеансом связи с источником данных. Данный механизм обычно применяется при работе с удалёнными (сетевыми) источниками данных, характеризующимися высокой латентностью, то есть задержкой в ответе на запрос.

С помощью этого метода можно обеспечить оптимизацию временного ресурса, затраченного на один сигнал, и тем самым увеличить максимальное количество опрашиваемых сигналов за интервал времени опроса.

В качестве примера рассмотрим промышленный ПЛК Siemens S7–315 при опросе его по шине Profibus (1,5 Мбит/с). Среднее время обработки MPI-запроса этим контроллером составляет 30 мс. Если использовать синхронный механизм для каждого сигнала, т.е. один запрос на каждый сигнал, то в течении одной секунды мы сможем получить около 33 сигналов. А если применить асинхронный механизм, т.е. в одном MPI-пакете получать до 220 байт или 110 сигналов целочисленного типа на 16-разрядов, то мы сможем за одну секунду получить до 3630 сигналов. Как можно видеть, эффективность асинхронного механизма в данном случае составляет 110 раз, а именно значение максимальной ёмкости MPI-пакета.

Недостатком асинхронного механизма является то, что запрос значения атрибута параметра возвращает не актуальное на момент запроса значение, а значение последнего сеанса опроса контроллера. Впрочем, если учесть, что источник данных может обновляться с периодичностью аппаратных ограничений АЦП, да и сами датчики могут иметь определённые ограничения в скорости реакции, то применение асинхронного механизма сбора может иметь серьёзные основания.

Применение асинхронного механизма для записи значений в ПЛК является достаточно редким явлением, поскольку запись значений обычно подразумевает воздействие оператора на ТП. Оператор по факту достаточно редко вносит коррективы в процесс, следовательно, запись можно выполнять синхронно. Однако существуют ситуации, например, управление ТП регуляторами на SCADA-системе, выполняющей функции среды исполнения ПЛК.

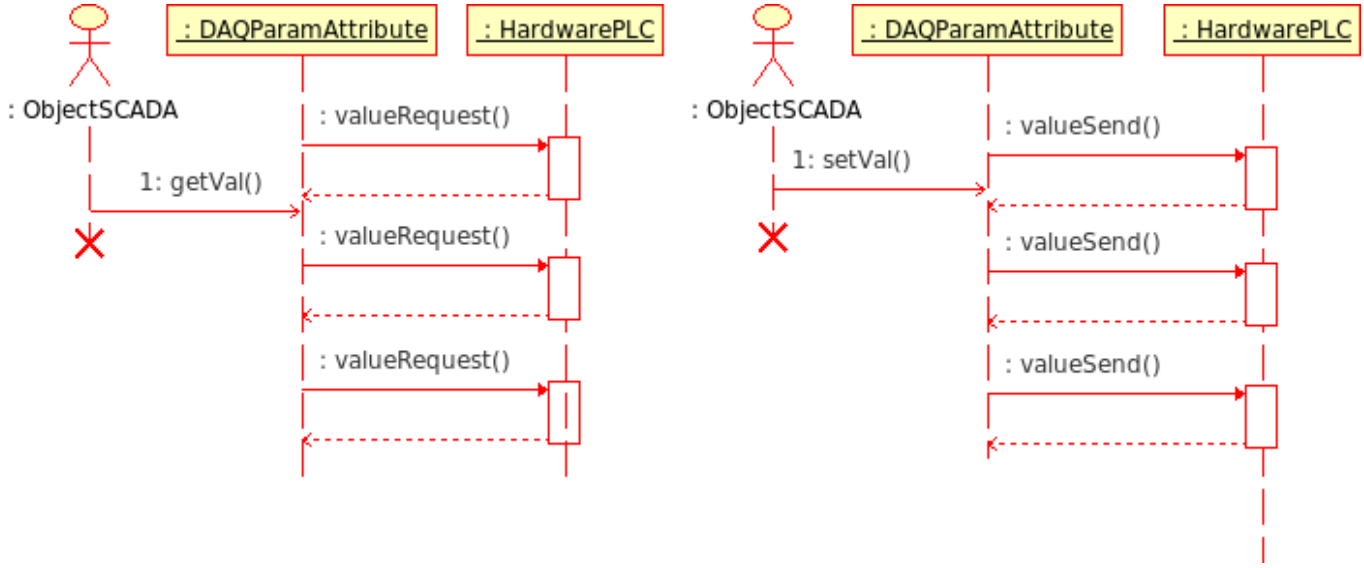


Рис. 3. Диаграмма последовательности взаимодействия при асинхронных запросах.

В соответствии с диаграммой выше мы получаем следующую картину:

- объект атрибута параметра (или вышестоящий объект контроллера) выполняет периодические запросы HardwarePLC::valueRequest() для получения значения сигнала или группы сигналов;
- полученные значения сигналов размещаются в объектах атрибутов параметров локально;
- объект SCADA-системы шлёт запрос значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса опроса источника данных.

В OpenSCADA такой механизм реализуют следующие модули подсистемы "Сбор данных".

- *Siemens* — модуль доступа к данным контроллеров фирмы Siemens серии S7. В данном модуле асинхронный режим реализован как для чтения данных, так и для их записи (опционально) на ПЛК.
- *ModBus* — модуль доступа к данным источников посредством семейства протоколов Mod Bus. В модуле реализован асинхронный режим чтения данных.
- *SNMP* — модуль доступа к данным устройств сети посредством Simple Network Management Protocol. В модуле реализован асинхронный режим чтения данных.
- *System* — модуль доступа к данным окружения исполнения OpenSCADA. В модуле реализован асинхронный режим чтения данных.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. В модуле реализован асинхронный режим чтения данных.

1.3. Пакетный механизм сбора

Пакетный механизм сбора данных характерен сбором данных каждого сигнала пакетом, включающим историю его изменения. Т.е. за один сеанс опроса данных получается несколько значений истории сигнала. Пакетный механизм работает совместно с синхронным и асинхронными механизмами.

В случае работы с синхронным механизмом выполняется фактический проброс архива источника данных для оперативной работы в системе (рис. 2). Как и простой синхронный механизм, его желательно применять только на низколатентных источниках данных или с источниками, работа которых является сеансовой, например, в сфере коммерческого учёта для чтения значений счётчиков.

При работе совместно с асинхронным механизмом история полученых сигналов обычно прямо помещается в архивы (рис. 4), а текущее значение атрибута параметра устанавливается в последнее значение пакета. Данная комбинация эффективна при сборе быстрых данных или при синхронизации архивов после потери связи с удалённым источником данных.

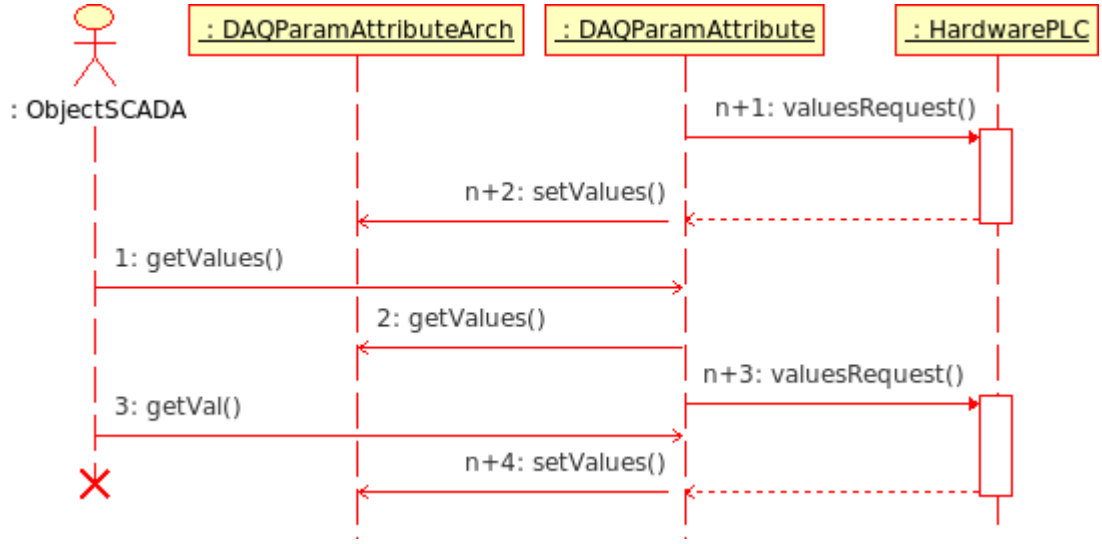


Рис. 4. Диаграмма последовательности взаимодействия при асинхронных запросах пакетного механизма.

В соответствии с диаграммой выше мы получаем следующее поведение пакетного механизма при асинхронных запросах:

- объект атрибута параметра или вышестоящий объект контроллера выполняет периодические запросы HardwarePLC::valuesRequest() для получения пакетов значений сигнала или группы сигналов;
- полученные пакеты значений сигналов помещаются в архив запросом DAQParamAttributeArch::setValues(), а последнее значение пакетов размещается в объектах атрибутов параметров;
- объект SCADA-системы шлёт запрос фрагмента архива к объекту атрибута параметра DAQParamAttribute::getValues(), а тот перенаправляет запрос к архиву DAQParamAttributeArch::getValues(). В результате возвращается фрагмент архива, доступный после предыдущего сеанса опроса источника данных;
- объект SCADA-системы шлёт запрос последнего значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса опроса источника данных.

В OpenSCADA такой механизм реализуют следующие модули подсистемы "Сбор данных".

- *DiamondBoards* — модуль доступа к данным PC/104 плат фирмы Diamond Systems. Платы PC/104 размещаются на ISA-шине, следовательно, являются локальными и доступны сравнительно быстро. В режиме сбора данных по прерыванию осуществляется ожидание пакетов быстрых значений (до 200 кГц) за одну секунду (до 200000 значений в пакете) и последующее размещение данных пакетов в архивах атрибутов параметров DAQ.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. Реализует синхронный и асинхронный пакетный режимы отражения архивов удалённых OpenSCADA-станций.

1.4. Пассивный механизм сбора

Пассивный механизм сбора данных характерен инициативой предоставления данных в SCADA-систему со стороны источника данных. Этот механизм является достаточно редким явлением, однако может иметь место в случае определённых условий или ограничений в возможности использования прямых механизмов сбора данных, рис. 5. Примером такой ситуации могут служить географически рассредоточенные системы сбора данных посредством мобильных сетей GPRS/EDGE. В таких сетях наделение клиентских узлов отдельными/реальными IP-адресами или формирование корпоративной мобильной сети может оказаться дорогим удовольствием, поэтому доступнее оказывается инициатива сеанса передачи данных с клиентских динамических IP-адресов на один реальный IP-адрес сервера SCADA-системы. Хотя возможна работа и через сетевую СУБД-посредника.

Воздействия на модификацию передаются источнику данных в момент сеанса передачи данных источником.

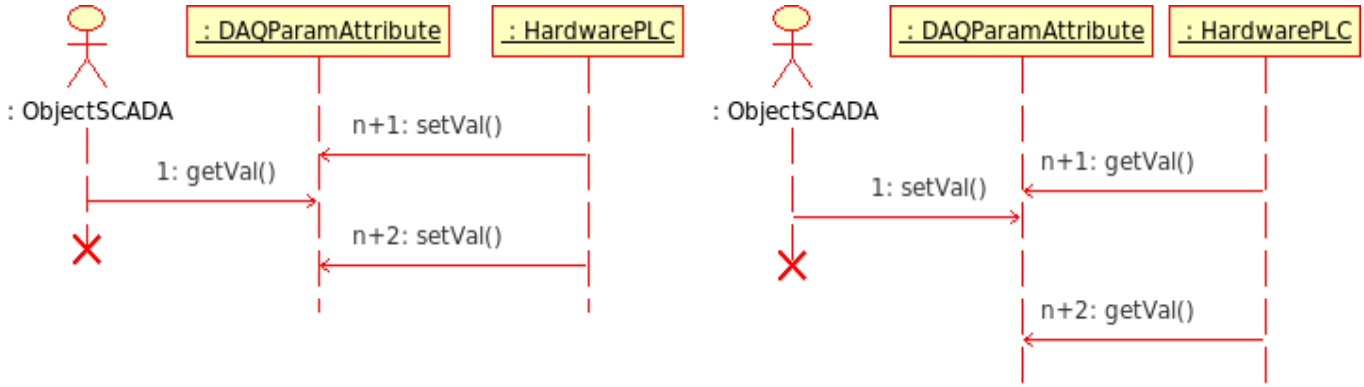


Рис. 5. Диаграмма последовательности взаимодействия при пассивном механизме работы.

В соответствии с диаграммой выше мы получаем следующее поведение пассивного механизма:

- объект источника данных осуществляет периодические сеансы связи с объектом атрибута параметра DAQParamAttributeArch::setVal() для передачи своих данных и получения команд воздействия;
- объект SCADA-системы шлёт запрос последнего значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса связи источника данных.

В OpenSCADA такой механизм ещё не использован, однако принципиальная возможность его реализации в системе есть.

2. Виртуальные источники данных

Кроме сбора физических данных актуальной является функция виртуального сбора данных. Виртуальные данные представляют собой данные, полученные внутри системы как независимо, так и на основе физических данных. Практически механизмы формирования виртуальных данных реализуются совместно с механизмом пользовательских вычислений. В среде промышленных контроллеров и SCADA-систем используются различные языки программирования. В случае с контроллерами в качестве таких языков часто используются языки низкого уровня (ассемблеры), однако в последнее время всё чаще используются языки высокого уровня (C, Pascal и другие), а также формальные языки МЭК 61131–3 (схемы потоков состояний SFC, блочные схемы FBD, релейные схемы LD и текстовые ST, IL). В случае со SCADA-системами вычисления чаще обеспечиваются языками программирования высокого уровня и формальными языками.

В системе OpenSCADA могут быть реализованы интерфейсы программирования и виртуальных источников данных на основе различных языков в отдельных модулях подсистемы «Сбор данных». На момент версии 0.6.3.2 доступны модули виртуальных вычислителей:

- Вычислитель на Java-подобном языке: /Doc / Java Like Calc,
- Блочный вычислитель: /Doc / Block Calc.

В ядро OpenSCADA интегрирован механизм пользовательских функций или API пользовательского программирования. Пользовательские функции могут предоставляться любым объектом системы, в том числе и модулями в соответствии со своей функциональностью, тем самым предоставляя пользователю некий набор функций для контроля за тем или иным объектом. Функции пользовательского API могут быть как статическими, т.е. реализующими фиксированную функциональность отдельного объекта, так и динамическими, т.е. формируемые пользователем под нужную ему задачу на языке пользовательского программирования высокого уровня.

Модуль /Doc / Java Like Calc предоставляет в систему механизм создания динамических пользовательских 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 с ассоциированной вычислительной функцией. Модулем предоставляется механизм прекомпиляции контекстно-зависимых функций, что используется для встраивания пользовательских алгоритмов непосредственно в контекст различных компонентов OpenSCADA. Например, это механизм шаблонов параметров подсистемы «Сбор данных» и движок среды визуализации и управления (СВУ).

Модуль /Doc / Block Calc предоставляет в систему OpenSCADA механизм создания пользовательских вычислений. Механизм вычислений основывается на формальном языке блочных схем (функциональных блоков). Языки блочного программирования основываются на понятии блочных схем (функциональных блоков). Причём в зависимости от сущности блока блочные схемы могут быть: логическими схемами, схемами релейной логики, моделью технологического процесса и другое. Суть блочной схемы состоит в том, что она содержит список блоков и связи между ними. С формальной точки зрения блок – это элемент (функция), который имеет входы, выходы и алгоритм вычисления. Исходя из концепции среды программирования, блок – это кадр значений, ассоциированный с объектом функции. Входы и выходы блоков нужно соединять для получения цельной блочной схемы.

С целью наполнения API пользовательского программирования пользовательскими функциями созданы следующие специализированные модули статических функций API пользовательского программирования:

- Библиотека функций совместимости со SCADA Complex1: /Doc / F Lib Complex 1,
- Библиотека стандартных математических функций: /Doc / F Lib Math,
- Библиотека функций системного API: /Doc / F Lib SYS.

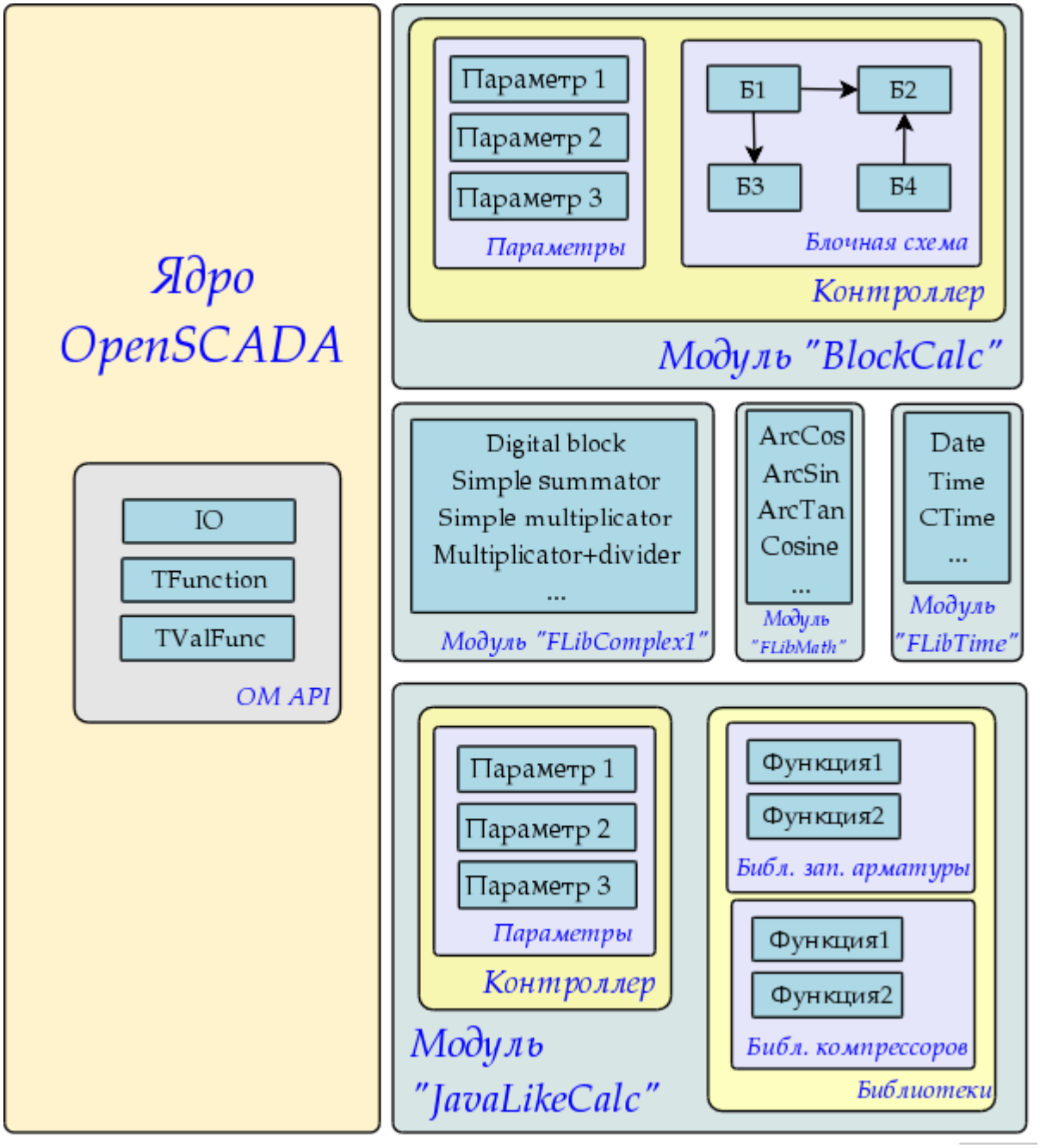


Рис. 6. Общая структура компонентов среды программирования

3. Логический уровень обработки данных

Выше мы говорили о том, что тип источника данных может колебаться от «сырого» до комплексного. Под «сырым» подразумевается источник, который предоставляет только элементарные сигналы (целое, вещественное, логическое, строка, ...), причём отдельно. Под комплексным подразумевается источник, который группирует сигналы и в параметре подсистемы "Сбор данных" предоставляет атрибуты дополнительного назначения, покрывающие практически все диагностические задачи, т.е. параметр является законченным объектом, нетребующим дополнения.

Учитывая такой разброс, может возникнуть ситуация, когда информации в объекте параметра контроллера источника данных недостаточно для описания реального объекта ТП в целом и нужен производный объект более высокого уровня абстракции. Решением такой ситуации может быть формирование дополняющих параметров, что является ненаглядным и вносит путаницу. Более правильным решением является использование прослойки, так называемого «Логического уровня», выполняющего функции гибкого формирования параметров, контейнеров сигналов, необходимой структуры и включающего пост-обработку.

Функционально «Логический уровень» предназначен для предоставления в системе OpenSCADA механизма свободного формирования объектов параметров, контейнеров сигналов, нужной структуры.

Эксплуатационным назначением «Логического уровня» является:

- расширение сферы применения системы OpenSCADA за счёт увеличения гибкости описания объектов параметров подсистемы "Сбор данных";
- сокращение затрат труда на создание сложных автоматизированных систем.

Концепция «Логического уровня» основана на шаблонах параметров, для которых в подсистеме "Сбор данных" предусмотрен контейнер библиотек шаблонов (рис. 1). Каждая библиотека содержит шаблоны параметров, которые могут использоваться модулями подсистемы "Сбор данных" для реализации параметров на основе шаблонов. Модулями системы OpenSCADA, которые используют шаблоны в своей работе, являются:

- LogicLev — модуль реализации классической концепции "Логического уровня".
- Siemens — модуль сбора данных контроллеров фирмы Siemens серии S7. В виду высокой гибкости и функциональности контролеров фирмы этой серии, которая позволяет формировать комплексные типы данных различной структуры, все параметры этого модуля работают по шаблонам.

Общий механизм работы "Логического уровня" на примере модуля LogicLev изображён на рис. 7.

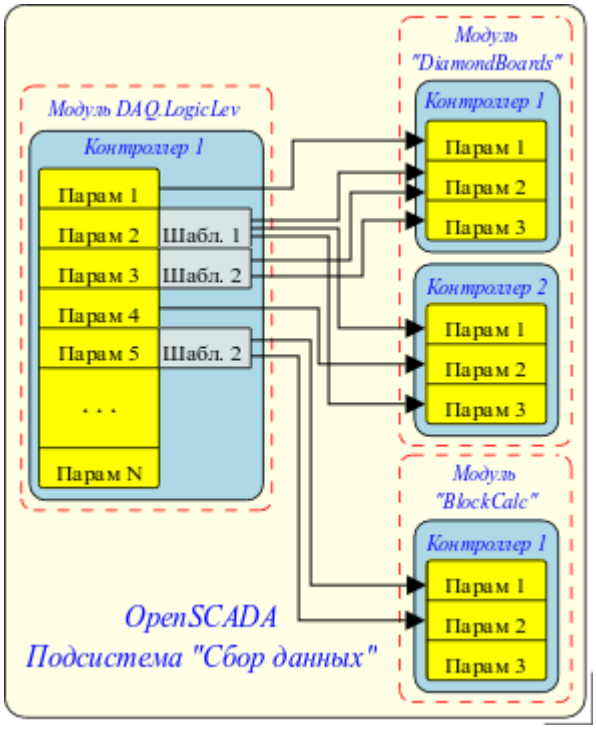


Рис. 7. Механизм работы "Логического уровня" на примере модуля LogicLev.

Исходя из изображения видно, что параметры контроллера логического уровня выполняют функцию отражения других параметров подсистемы "Сбор данных" (на примере параметров 1 и 4) и произвольное формирование параметров на основе шаблонов 1, 2 и других параметров подсистемы "Сбор данных" (на примере параметров 2, 3 и 5).

Структура параметров с шаблоном в основе имеет структуру, изображённую на рис. 8.

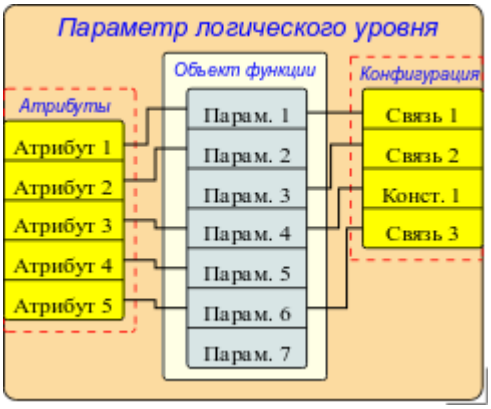


Рис. 8. Структура параметров, с шаблоном в основе.

Как можно видеть из структуры, параметр логического уровня состоит из объекта функции, атрибутов и конфигурации шаблона. Объект функции это экземпляр исполнения функции шаблона с набором входов/выходов и программой вычисления шаблона на одном из языков пользовательского программирования, обычно это Java-подобный язык пользовательского программирования модуля DAQ.JavaLikeCalc. Впрочем шаблон может быть вообще без программы, предоставляя только структуру проброса входов/выходов. Атрибуты в структуре изображают перечень атрибутов результирующего параметра в соответствии с шаблоном. Конфигурация в структуре предоставляет конфигурацию свойств шаблона и его внешних связей.

Логику работы логического уровня параметров можно записать следующим образом:

- Параметр связывается с шаблоном из которого получается структура атрибутов, в соответствии с функцией шаблона.
- В момент связывания параметра с функцией выполняется связывание объекта экземпляра функции параметра с функцией из шаблона.
- Далее, в соответствии с шаблоном функции, формируется структура связей. Исходя из структуры связей формируется форма связывания параметра и пользователем

устанавливаются связи.

- При доступе к атрибутам полученного параметра производится проверка на наличие прямой связи. В случае наличия прямой связи запрос перенаправляется по этой связи, в противном случае значение берётся из объекта экземпляра функции параметра.
- В этот момент работает вычисление функции шаблона, по объекту функции параметров. При этом, перед вычислением производится чтение значений по связям, а после вычисления запись результатов по этим связям.

Шаблон параметров в целом предоставляет следующее:

- структуру входов/выходов функции шаблона;
- признаки конфигурации и связывания шаблона (константа, связь);
- предварительные значения конфигурации постоянных и шаблонов конфигурации связей;
- признаки атрибутов результирующего параметра логического уровня типов: не атрибут, атрибут с полным доступом, атрибут с доступом только на чтение;
- механизм вычисления входов/выходов функции шаблонов с использованием языка пользовательского программирования OpenSCADA.

На рис. 9 представлено изображение вкладки конфигурации шаблона параметров подсистемы "Сбор данных" в виде таблицы с конфигурацией входов/выходов и текста программы пользовательского программирования.

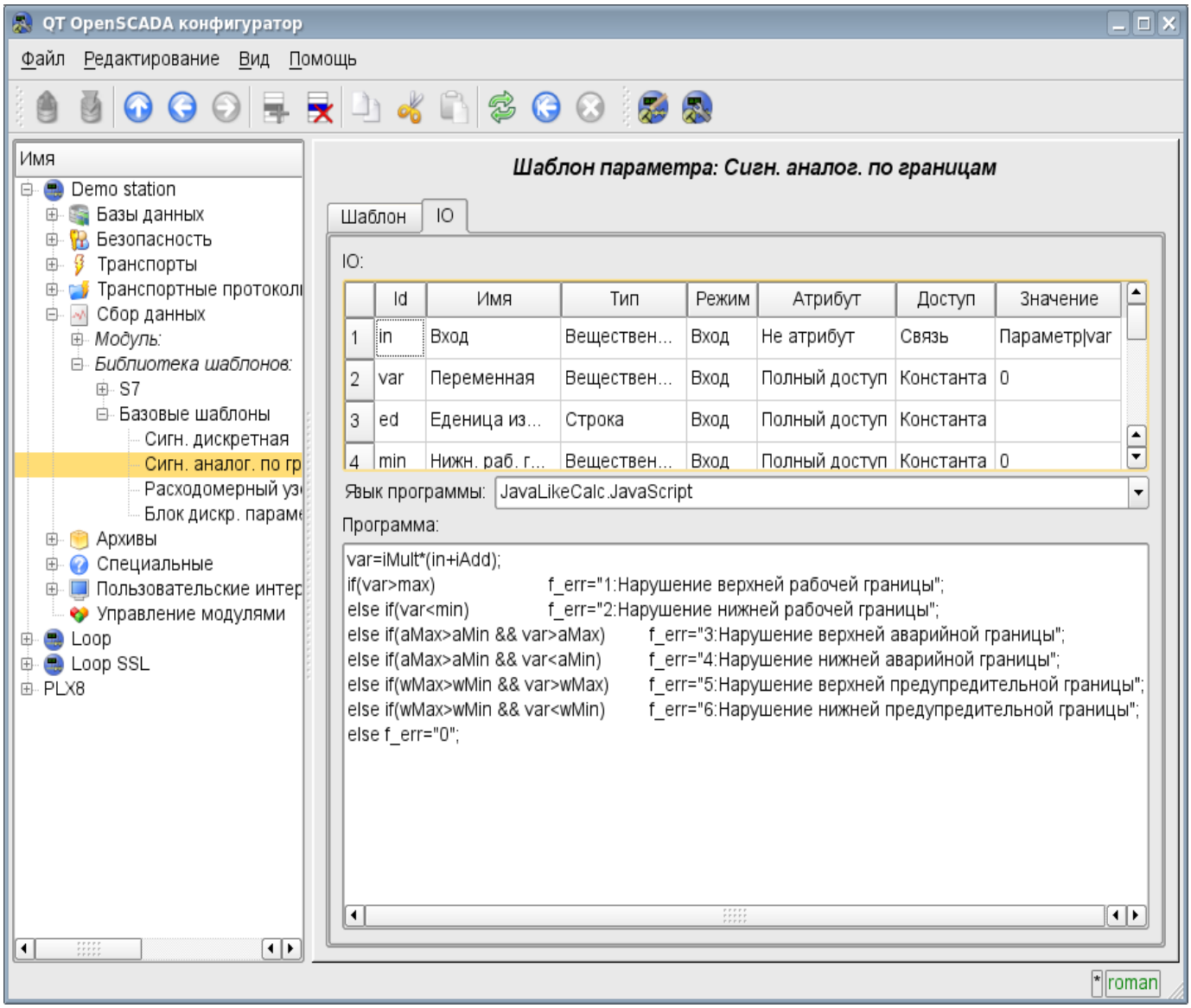


Рис. 9. Вкладка конфигурации шаблона параметров подсистемы "Сбор данных".

Полем входа/выхода шаблона параметра предусмотрены следующие свойства специализированного назначения: «Атрибут», «Доступ» и «Значение».

Свойство «Атрибут» выступает признаком отражения входа/выхода шаблона на результирующий атрибут параметра. Предусмотрены следующие варианты этого свойства:

- *не атрибут* — вход/выход функции шаблона не отражается на атрибут;
- *только чтение* — вход/выход функции шаблона отражается на атрибут с доступом только на чтение;
- *полный доступ* — вход/выход функции шаблона отражается на атрибут с полным доступом.

Свойство «Доступ» выступает признаком, указывающим на использование входа/выхода функции шаблона в результирующей конфигурации шаблона на логическом уровне. Предусмотрены следующие варианты этого свойства:

- *константа* — доступен для установки только на уровне конфигурации шаблона параметра в виде постоянной;
- *публичная константа* — доступен для установки на уровне параметра логического уровня в разделе конфигурации шаблона в виде постоянной;
- *Связь* — доступен для установки на уровне параметра логического уровня в разделе конфигурации шаблона в виде связи.

Поле «Значение» описывает предустановленное значение для постоянных и шаблон конфигурации внешних связей. Шаблон конфигурации внешних связей используется в целях описания механизма группировки и автоматического распределения внешних связей. Структура шаблона конфигурации внешних связей специфична для каждого модуля подсистемы «Сбор данных», который использует механизм шаблонов. В случае с модулем логического уровня распределение производится над внешними атрибутами параметров с шаблоном конфигурации внешней связи вида: <Параметр>|<атрибут>. Где «Параметр» используется для объединения параметров и помещения на форму конфигурации, а атрибут для ассоциативного связывания атрибутов при назначении параметра.

В качестве примера использования шаблона на рис.10 приведём изображения параметра модуля логического уровня "F3". На рис.10 представлена вкладка "Конфигурация шаблона" для конфигурации, включая связывание, шаблона параметра. На рис.11 представлена вкладка "Атрибуты" с перечнем атрибутов и их значений, созданных посредством шаблона.

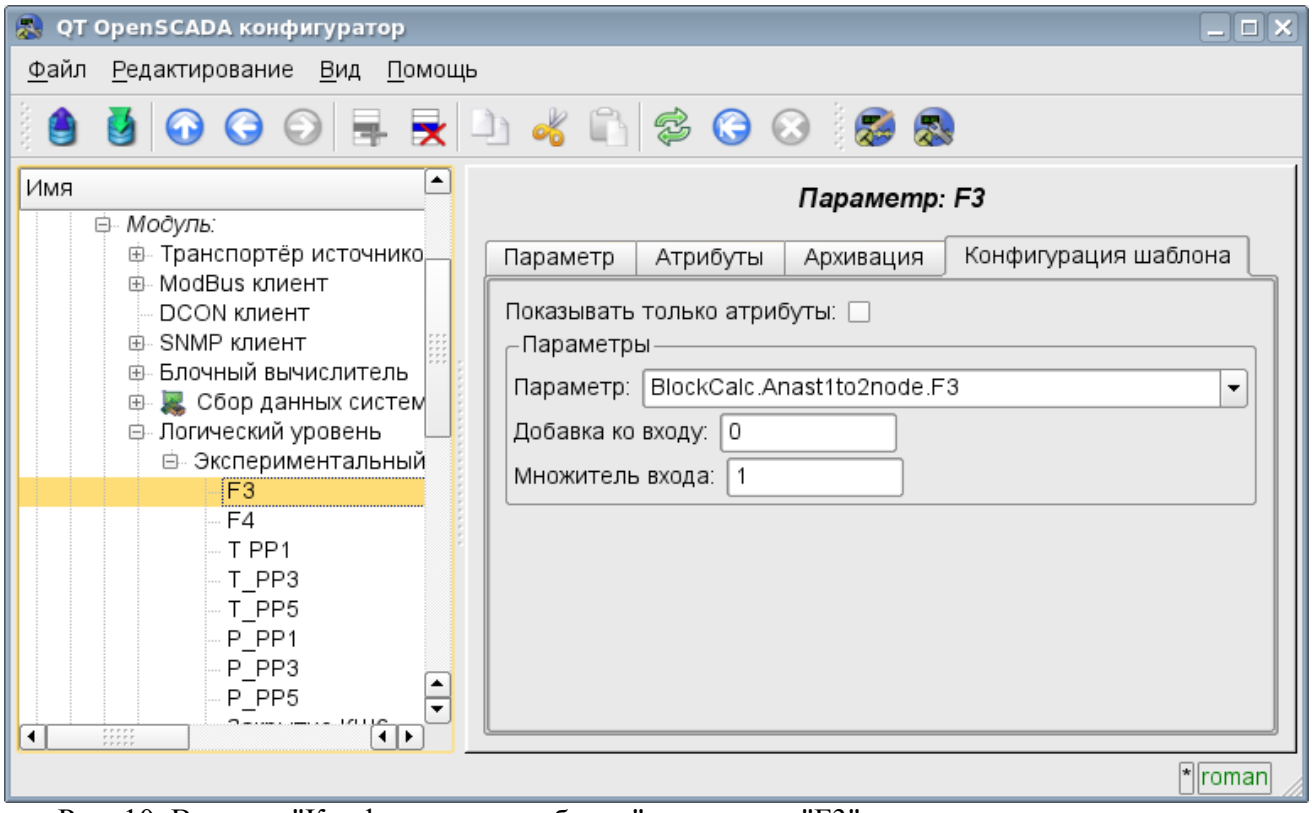


Рис. 10. Вкладка "Конфигурация шаблона" параметра "F3" модуля логического уровня.

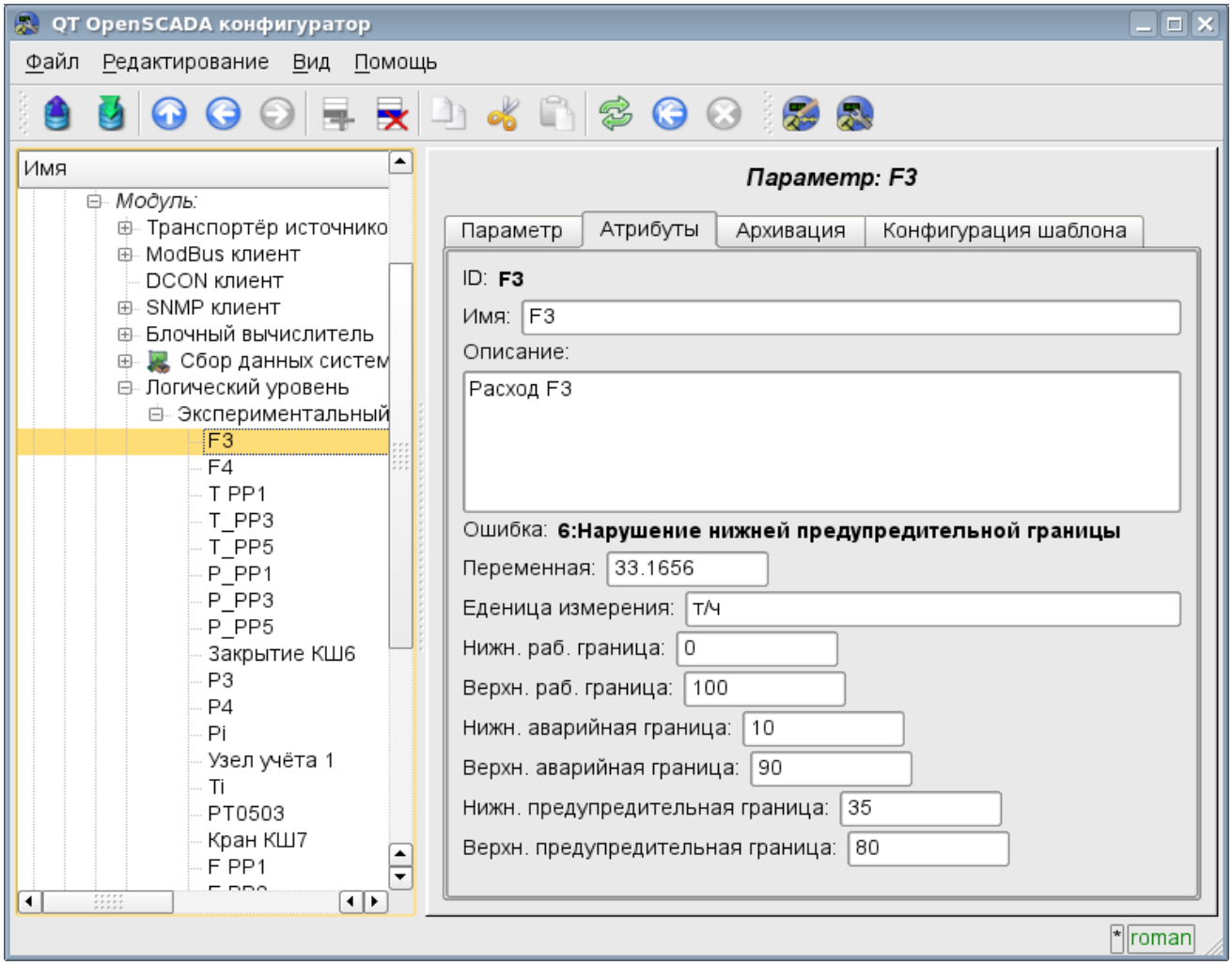


Рис. 11. Вкладка "Атрибуты" параметра "F3" модуля логического уровня.

4. Резервирование источников данных

Резервирование вообще и источников данных в частности служит для повышения общего уровня отказоустойчивости решения путём включения дублирующих узлов в совместную работу с основным узлом. В случае сбоя основного узла происходит подхват функций основного узла резервным. Часто схема резервирования может работать и в режиме распределения нагрузки между совместно работающими узлами.

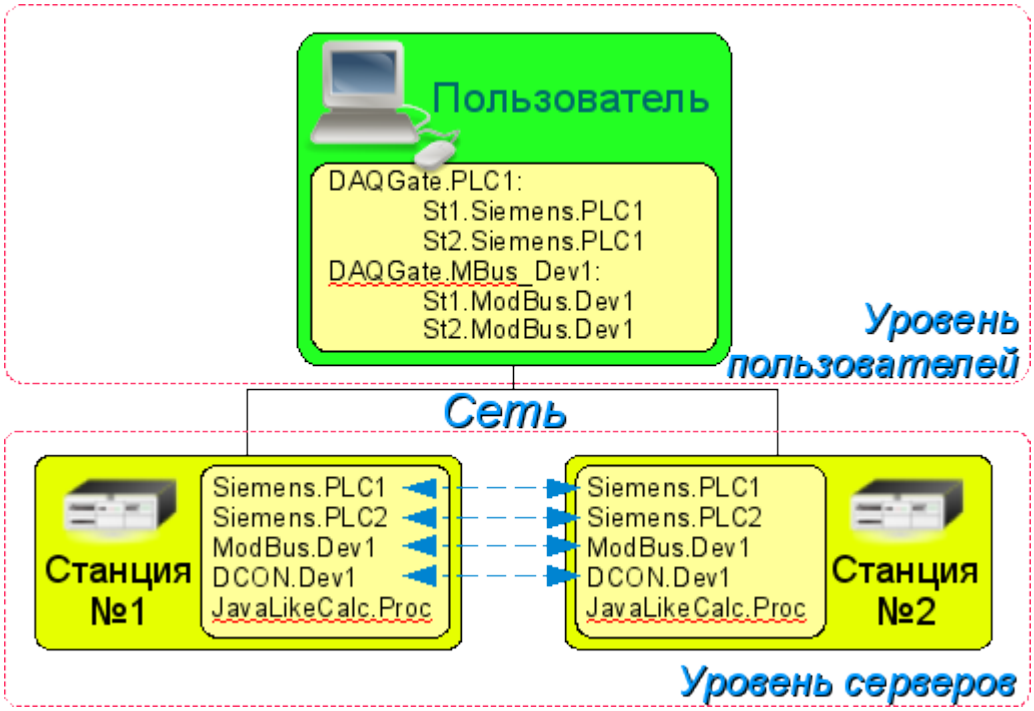


Рис. 12. Горизонтальное и вертикальное резервирование.

В случае с подсистемой «Сбор данных» системы OpenSCADA резервирование данных (рис.12) выполняет функции:

- Резервирование механизма сбора данных. Обычно эта функция реализуется без особых механизмов путём простого запуска параллельных резервных станций с одинаковой конфигурацией и работающих независимо. Однако в случае выполнения станцией функции ПЛК такое поведение недопустимо по причине одновременной выдачи управляющих воздействий и отсутствия синхронизации данных вычислителей.
- Компенсация потери данных на время простоя узла за счёт архива резервного узла. Предусмотрены два механизма компенсации. Первый и основной механизм осуществляет загрузку участков архива из резервной станции в момент запуска станции в целом или отдельных контроллеров подсистемы "DAQ". Участок архива запрашивается с момента последней записи в локальном архиве и по текущее время. Глубина запроса при этом ограничивается указанием предельного времени в конфигурации резервирования. Второй, дополняющий механизм, осуществляет заполнение дыр в архиве в момент фактического запроса пользователя к этим данным. Такой подход с одной стороны позволяет осуществить прогнозируемую по времени синхронизацию при старте, а с другой стороны фактически исключает потерю данных при условии работы хотя бы одной станции в течение всего рабочего времени.
- Распределение нагрузки по сбору данных между узлами. При создании сложных распределённых систем может оказаться важным вопрос прогнозирования и оптимизации общей производительности системы. С учётом таких задач механизм резервирования предусматривает исполнение задач сбора данных отдельных источников (контроллеров OpenSCADA) только на одной станции. При этом задачи остальных станций переходят в режим синхронизации данных с исполняющей станцией. В случае потери связи с исполняющей станцией запускается задача локального сбора данных. Предусмотрена также возможность оптимального распределение нагрузки исполнения задач сбора данных группы

контроллеров между станциями.

- Оптимизация нагрузки на внешние источники данных за счёт запроса данных у внешнего источника только одним узлом. В практике часто встречаются высоконагруженные источники данных или интерфейсы доступа к источникам данных, для которых даже сбор данных одной станцией может быть проблемой и потребует снижения периодичности сбора, т.е. качества данных. Механизм резервирования, кроме распределения нагрузки между станциями по описанной выше схеме позволяет снять дополнительную нагрузку на источник данных и его интерфейсы, тем самым повысив качество данных.
- Предотвращение некоторого отличия данных на разных узлах, связанное с несовпадением моментов времени при независимом сборе данных отдельными узлами путём получения данных у станции с активным контроллером. В системах высокой отчётности с резервированием должно быть исключено или сведено к минимуму расхождение в данных на разных станциях, что подразумевает реальный сбор данных одной станцией и синхронизацию с этими данными остальных станций.

Настройка резервирования начинается с добавления резервных станций в список системных станций OpenSCADA на вкладке "Подсистема" подсистемы "Транспорты" (рис.13). Далее вся конфигурация резервирования производится во вкладке "Резервирование" подсистемы "Сбор данных" (рис.14).

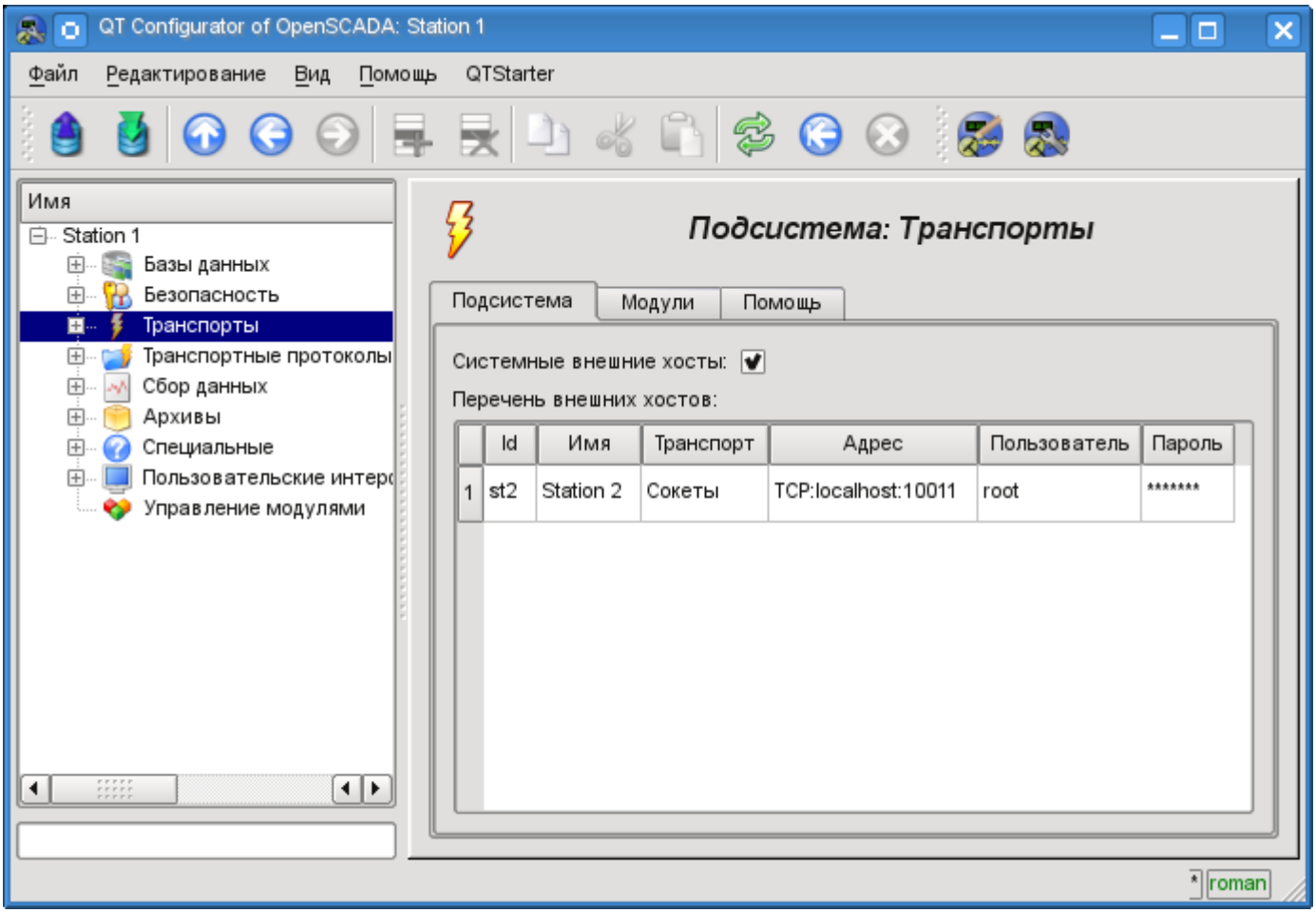


Рис. 13. Вкладка "Подсистема" подсистемы "Транспорты".

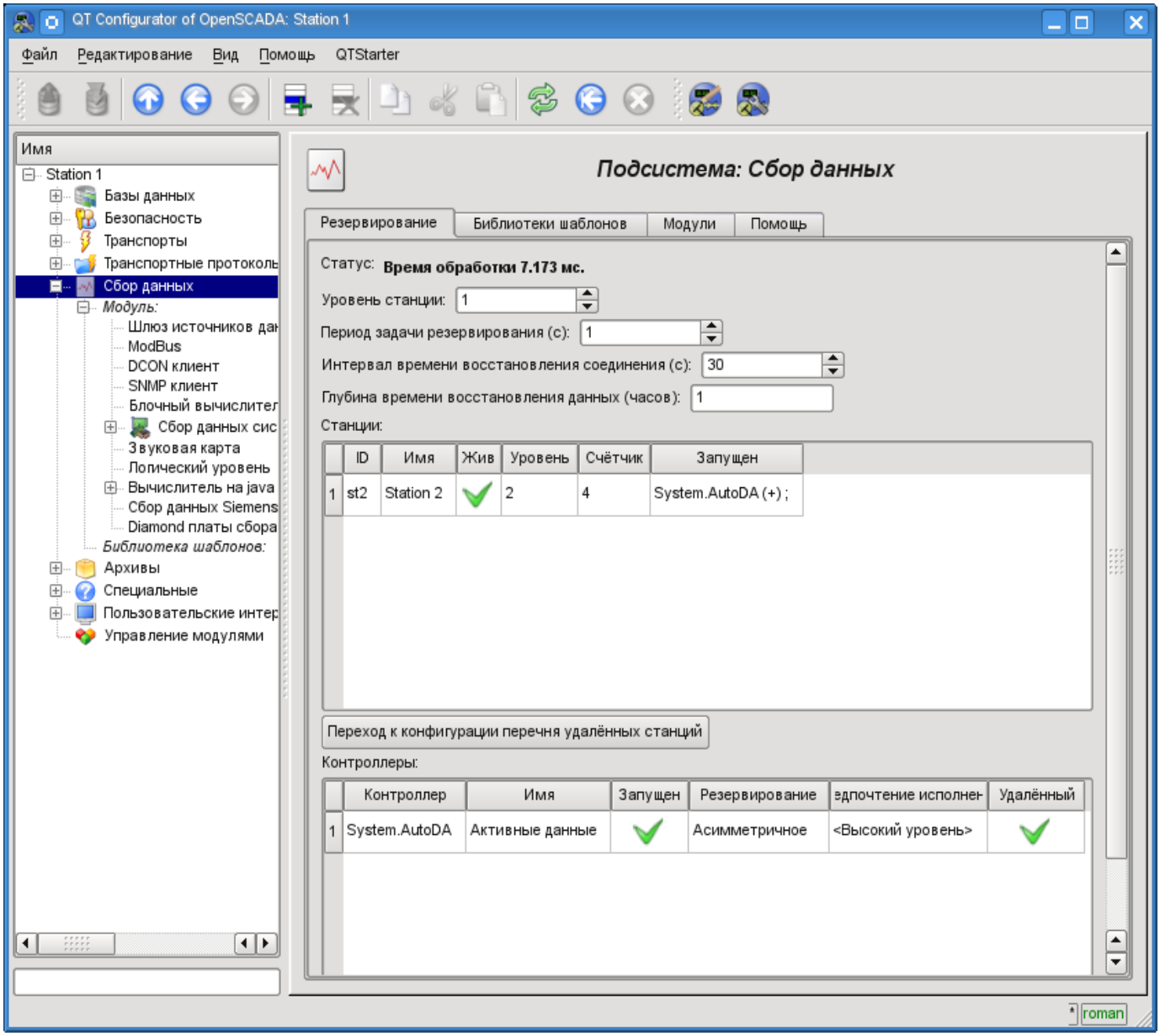


Рис. 14. Вкладка "Резервирование" подсистемы "Сбор данных".

Задача обслуживания механизма резервирования запускается всегда и исполняется с периодичностью, установленной в соответствующем конфигурационном поле. Реальная работа по осуществлению резервирования осуществляется при наличии хотя бы одной резервной станции в списке станций и предполагает:

- Контроль за соединением с внешними станциями. В процессе контроля осуществляются запросы к удалённым станциям за обновлением информации о них и проверке связи. В случае потери связи со станцией повтор подключения к ней осуществляется через промежуток времени, указанный в конфигурационном поле интервала времени восстановления соединения. В поле "Жив" станции отображается текущее состояние связи. В поле "Счётчик" представлено количество запросов, осуществлённых к удалённой станции, или же время, оставшееся для осуществления следующей попытки соединения с потерянной станции. В поле "Запущен" приводится перечень активных контроллеров на удалённой станции с признаком локального исполнения.
- Локальное планирование исполнения контроллеров в резерве. Планирование осуществляется в соответствии с уровнями станций и предпочтениями исполнения контроллеров.
- Вызов функции синхронизации данных для локальных контроллеров, работающих в режиме синхронизации данных из внешних станций. В процессе вызова осуществляется подготовка запроса к данным удалённой станции для параметров в контроллере и отталкиваясь от времени прошлого запроса. По запросу возвращаются только значения

модифицированных атрибутов и последовательность значений из архива в случае потери нескольких циклов значений.

Для контроля за временем, затраченным на выполнение цикла задачи обслуживания резервирования предусмотрено поле статуса. При приближении реального времени выполнения к циклу задачи обслуживания резервирования рекомендуется увеличить периодичность исполнения этой задачи!

Для контроллера подсистемы "Сбор данных" предусмотрены режимы асимметричного и симметричного резервирования. Асимметричное резервирование работает с той конфигурацией контроллера удалённой станции, какая есть и не пытается её обобщать. Симметричный режим подразумевает синхронизацию конфигурации контроллеров станций с конфигурацией станции наивысшего уровня, а также предполагает внесение изменений в конфигурацию всех контроллеров станций при изменении её на одной из станций. На данный момент этот режим не реализован!

Быстрый старт OpenSCADA

Открытая система OpenSCADA является предельно модульной, гибкой и многофункциональной SCADA-системой. Как следствие этого первое знакомство с OpenSCADA может быть достаточно сложным по причине малых шансов совпадения предыдущего опыта пользователя или полного его отсутствия, с подходами работы в OpenSCADA. Однако, в значительной степени это только первое впечатление, поскольку вся мощь OpenSCADA оказывается на ладони у пользователя, от изобилия которой пользователь может растеряться, и ему могут понадобиться значительные усилия для отбора функций, нужных в решении его задачи.

По этой причине и для наглядного представления общей концепции работы в OpenSCADA создан данный документ. Документ в достаточно краткой и наглядной форме представляет путь от запуска OpenSCADA до создания элементов пользовательского интерфейса на реальных примерах. Кроме того, документ содержит раздел с рецептами по конфигурации, реализации и решению типовых задач пользователя.

Документ не содержит детального описания концепции и глубокого погружения в детали OpenSCADA, а предоставляет ссылки на документы OpenSCADA, содержащие такую информацию.

Описание документа ведётся синхронно с реализацией примеров на демонстрационной базе данных (БД) OpenSCADA. Следовательно, пользователь должен получить дистрибутив OpenSCADA с этой БД, для наглядного изучения и опробования примеров.

1. Термины, определения и аббревиатуры

АРМ (аббревиатура) - Автоматизированное Рабочее Место. Обычно представляет из себя системный блок вычислительной системы, дисплей, манипулятор «мышь» иногда с клавиатурой, и другое периферийное оборудование, которое служит для визуального представления данных технологического процесса, а также выдачи управляющих воздействий на ТП.

Блокировка (термин) – условная граница технологического параметра, по преодолению которой выполняются предустановленные алгоритмом действия по предотвращению аварии. В некоторых режимах ТП (пуск) в соответствии с регламентом может быть необходимо отключение блокировки (деблокирование).

Деблокирование (термин) – процесс отключения блокировки на время работы ТП в режимах, для которых в регламенте предусмотрена данная операция. Внимание, деблокирование технологических параметров является строго отчётной операцией и должно производится оперативным персоналом в соответствующем порядке.

Квитация (термин) – процесс подтверждения факта того, что оперативный персонал обратил внимание на нарушение в работе ТП. Обычно этот процесс подразумевает принятие мер оператором для устранения нарушения и нажатие соответствующей кнопки для прекращения сигнализации.

ПЛК (аббревиатура) – Промышленный Логический Контроллер. Микропроцессорное электронное устройство, на которое через устройство сопряжения с объектом (УСО) собираются сигналы технологических параметров. ПЛК выполняет функцию непосредственного сбора данных, их обработку и выдачу управляющих воздействий посредством алгоритмов автоматического регулирования. Кроме того ПЛК предоставляет данные для визуализации ТП, а также получает данные ручного вмешательства оператора от системы верхнего уровня.

Сигнализация (термин) – процесс уведомления оперативного персонала о нарушении технологического процесса или работы оборудования автоматизации. Способы сигнализации могут быть различных типов воздействия на органы чувств человека с целью привлечения внимания. Часто предусматриваются следующие типы сигнализации:

- *Световая сигнализация* – обычно осуществляется посредством смены цвета графического объекта (миганием) для возникающих событий и установкой статичных аварийных цветов (красный и жёлтый) для сквитированных событий.

- *Звуковая* – осуществляется выдачей звукового сигнала в момент возникновения события. Тип звукового сигнала может быть как монотонным, так и синтезированным речевым сообщением с информацией о нарушении.

ТП (аббревиатура) - Технологический Процесс. Весь комплекс технологического оборудования производственного процесса.

УСО (аббревиатура) – Устройство Сопряжения с Объектом. Ряд устройств или модулей ПЛК, к которым непосредственно подключаются сигналы с датчиков ТП для последующего преобразования из аналогового в цифровой вид и обратно. Преобразование осуществляется с целю последующей обработки значений технологических параметров в ПЛК.

Уставка сигнализации (термин) - условная граница значения технологического параметра, преодоление которой считается нештатной ситуацией. Обычно предусматриваются следующие границы:

- *Верхняя и нижняя аварийные границы* – границы аварийных значений технологического параметра.
- *Верхняя и нижняя предупредительные границы* – границы предупреждения, регламентные границы, о выходе технологического параметра за рабочий диапазон.
- *Отказ* - признак выхода значения параметра за аппаратные границы технологического оборудования. Обычно характеризует отказ датчика, обрыв канала связи с датчиком или ПЛК.

SCADA (аббревиатура-ан.) - Supervisory Control And Data Acquisition (Диспетчерская система управления и сбора данных). Программное обеспечение, выполняющее комплекс задач по сбору данных ТП, их архивирование и представление, а также выдачу управляющих воздействий оператором в ручном режиме.

2. Установка и запуск

Установку дистрибутива OpenSCADA можно осуществить двумя способами. Первый и простой способ - это получить готовые пакеты для используемого дистрибутива ОС Linux. Второй - собрать систему OpenSCADA из исходных текстов. В целом процедура установки сильно зависит от используемого дистрибутива Linux и исчерпывающе её описать в данном руководстве не представляется возможным! Поэтому может понадобиться глубокое знакомство с механизмами установки ПО выбранного дистрибутива Linux в его документации.

В случае отсутствия у пользователя достаточно глубоких знаний и умений в выбранном дистрибутиве Linux, настоятельно рекомендуется выбирать дистрибутив Linux по критерию наличия для него пакетов системы OpenSCADA в репозиториях дистрибутива, что позволит и гарантирует простейшую и безпроблемную установку!

Если у пользователя вызывает затруднение установка не только OpenSCADA, но и дистрибутива Linux, то на первое время он может воспользоваться "живым" дистрибутивом Linux, с установленной и готовой для работы и изучения демонстрацией OpenSCADA. На данный момент доступны "живые" сборки на основе дистрибутива ALTLinux в виде CD и Flash-образов по ссылке: ftp://ftp.oscada.org/OpenSCADA/Live .

2.1. Установка OpenSCADA из готовых пакетов

Установка OpenSCADA из готовых пакетов, в свою очередь, может осуществляться двумя методами. Первый - простейший, когда пакеты OpenSCADA уже присутствуют в официальных или дополнительных репозиториях используемого дистрибутива ОС Linux, и установка их - вопрос запуска типичной программы управления пакетами дистрибутива с последующим выбором пакетов OpenSCADA. Второй подразумевает получение пакетов OpenSCADA и установку их вручную.

На данный момент пакеты системы OpenSCADA можно встретить в репозиториях таких дистрибутивов ОС Linux: ALTLinux и дистрибутивах, основанных на пакетной базе Fedora.

Проверить на наличие пакетов OpenSCADA в репозиториях используемого дистрибутива Linux, а

также загрузить пакеты OpenSCADA для ручной установки можно на странице загрузки официального сайта OpenSCADA (http://oscada.org/ru/zagruzka).

Описание установки из репозитория выбранного дистрибутива Linux пропустим и отошлём читателя к документации соответствующего дистрибутива.

Для ручной установки пакетов OpenSCADA загрузим их из официального сайта или другого источника. Загрузить можно набор пакетов двух типов.

Первый тип представлен набором из четырёх пакетов:

- **openscada** - пакет со всеми файлами, нужными для запуска OpenSCADA в полном объёме, включая все модули;
- **openscada-demo** - файлы демонстрационной БД с конфигурацией для запуска демонстрации;
- **openscada-doc** - вся документация на систему OpenSCADA;
- **openscada-devel** - пакеты разработки, для отдельного создания модулей к системе OpenSCADA.

Второй тип представлен набором из порядка сорока пакетов с выделением модулей OpenSCADA в отдельные пакеты:

- **openscada** - содержит ядро OpenSCADA, базовую конфигурацию и запускающие файлы;
- **openscada-DB.*** - модули подсистемы "БД";
- **openscada-DAQ.*** - модули подсистемы "Сбор данных";
- **openscada-Archive.*** - модули подсистемы "Архивы";
- **openscada-Transport.*** - модули подсистемы "Транспорты";
- **openscada-Protocol.*** - модули подсистемы "Транспортные протоколы";
- **openscada-UI.*** - модули подсистемы "Пользовательские интерфейсы";
- **openscada-Special.*** - модули подсистемы "Специальные";
- **openscada-demo** - файлы демонстрационной БД с конфигурацией для запуска демонстрации;
- **openscada-doc** - вся документация на систему OpenSCADA;
- **openscada-devel** - пакеты разработки, для отдельного создания модулей к системе OpenSCADA;
- **openscada-plc** - виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как ПЛК;
- **openscada-server** - виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как SCADA-сервер;
- **openscada-visStation** - виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как визуальная SCADA-станция.

Первый тип набора пакетов предназначен для простой-ручной установки, поскольку содержит только четыре пакета. Второй тип предназначен для помещения в репозиторий дистрибутива Linux и последующей установки их с помощью пакетного менеджера, осуществляющего автоматическое разрешение зависимостей. Второй тип набора пакетов позволяет установить только нужные компоненты OpenSCADA, тем самым оптимизируя рабочее окружение, чего не позволяет осуществить набор пакетов первого типа.

При установке из репозитория выбираем только пакет "openscada-demo". Всё остальное, в соответствии с зависимостями, будет выбрано и установлено автоматически.

Ручную установку RPM-пакетов первого типа можно осуществить командой, предварительно сменив рабочую директорию на директорию с пакетами:

```
# rpm -i openscada-demo-0.6.4.1-alt2.i586.rpm openscada-0.6.4.1-alt2.i586.rpm
```

Ручную установку DEB-пакетов первого типа можно осуществить командой, предварительно сменив рабочую директорию на директорию с пакетами:

```
# dpkg -i openscada-demo_0.6.4.1-2_all.deb openscada_0.6.4.1-2_i386.deb
```

В процессе выполнения могут возникнуть ошибки, связанные с неудовлетворёнными зависимостями. При ручной установке из пакетов удовлетворять их прийдётся в ручную, подобно установке пакетов OpenSCADA, или через менеджер пакетов дистрибутива Linux. Детальнее ознакомиться с процессом установки ПО в RPM-пакетах можно по ссылке: http://skif.bas-net.by/bsuir/admin/node51.html.

2.2. Установка из исходных текстов

Если нет возможности получить готовые пакеты OpenSCADA для выбранного дистрибутива, то остаётся только вариант сборки OpenSCADA из исходных текстов. Процесс сборки OpenSCADA детально описан в руководстве по ссылке http://wiki.oscada.org/Doc/SborkaIzIsxodnikov. Однако нужно иметь в виду, что если Вам удалось собрать OpenSCADA из исходных текстов, то это руководство не для Вас и Вы, скорее всего, можете легко освоить базовую документацию OpenSCADA (http://wiki.oscada.org/Doc/OpisanieProgrammy).

Данный раздел приведён здесь только для полноты и целостности рассмотрения вопроса, поскольку требование квалификационного уровня пользователя для этого раздела значительно выше уровня документа в целом!

3. Первичная конфигурация и запуск

После корректной установки OpenSCADA с демонстрационной БД никакой предварительной настройки не требуется. Если нужно выполнить особую настройку, которая отличается от базовой, то воспользуйтесь документом описания программы OpenSCADA по ссылке: http://wiki.oscada.org/Doc/OpisanieProgrammy?v=14os#h827-1.

Внимание! Демонстрация OpenSCADA на основе демонстрационной БД это не то же, что обычно предоставляют коммерческие производители ПО с целью продемонстрировать возможности, но исключить или усложнить нормальную работу путём ограничения функций. Демонстрация OpenSCADA это полно-функциональная система, предоставляющая примеры реализации и настройки различных компонентов. На основе демонстрационной БД OpenSCADA можно легко создавать собственные проекты, используя предоставленные наработки.

Запустить на исполнение OpenSCADA с демонстрационной БД можно из меню окружения рабочего стола в разделе "Графика", пункт "Демонстрация открытой SCADA системы" с характерной иконкой (рис.3.1).

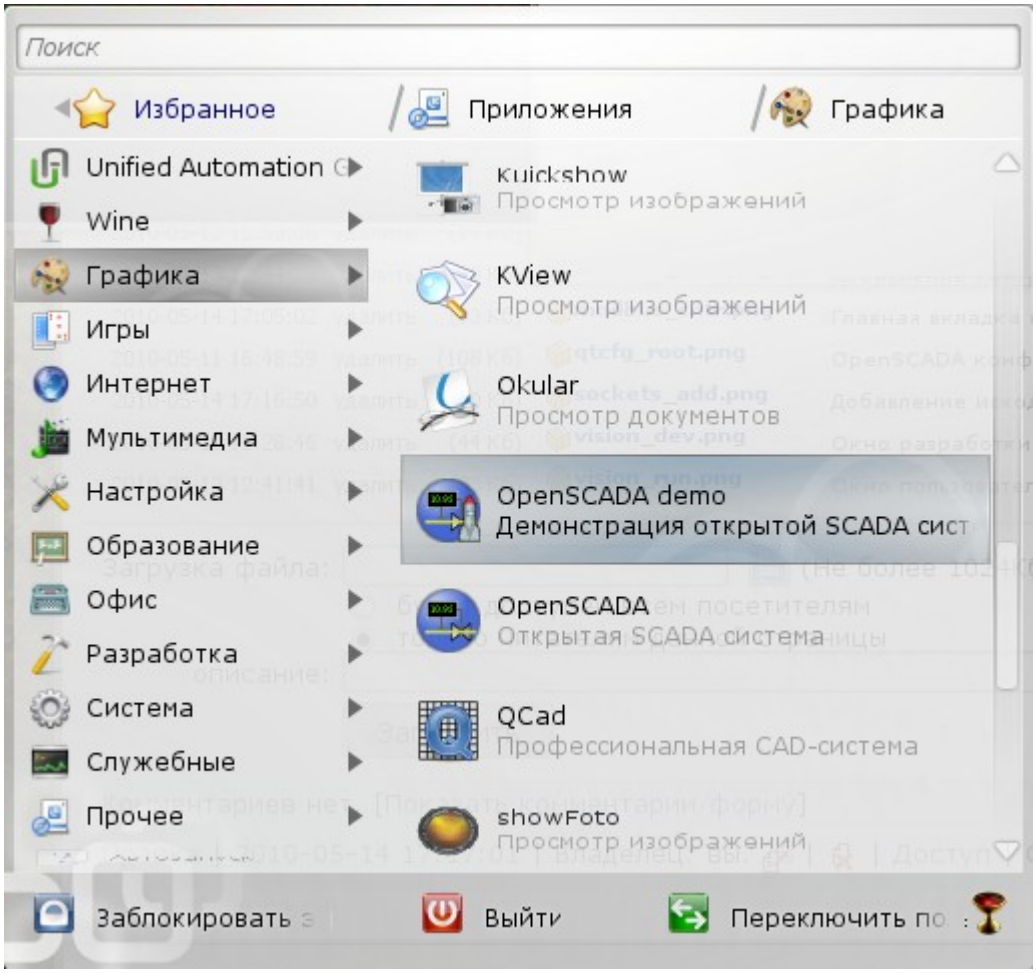


Рис. 3.1. Пункт меню окружения рабочего стола для запуска демонстрации OpenSCADA.

Запуск также можно осуществить из консоли командой:

```
# openscada_demo
```

После запуска получим окно графического конфигуратора системы OpenSCADA - QTCfg (рис.3.2) с открытой корневой страницей. Демонстрационная БД специально настроена так, что бы первым при запуске появлялось окно конфигуратора. В дальнейшем можно открыть окно разработки графических интерфейсов пользователя, а также запустить проект пользовательского интерфейса на исполнение.

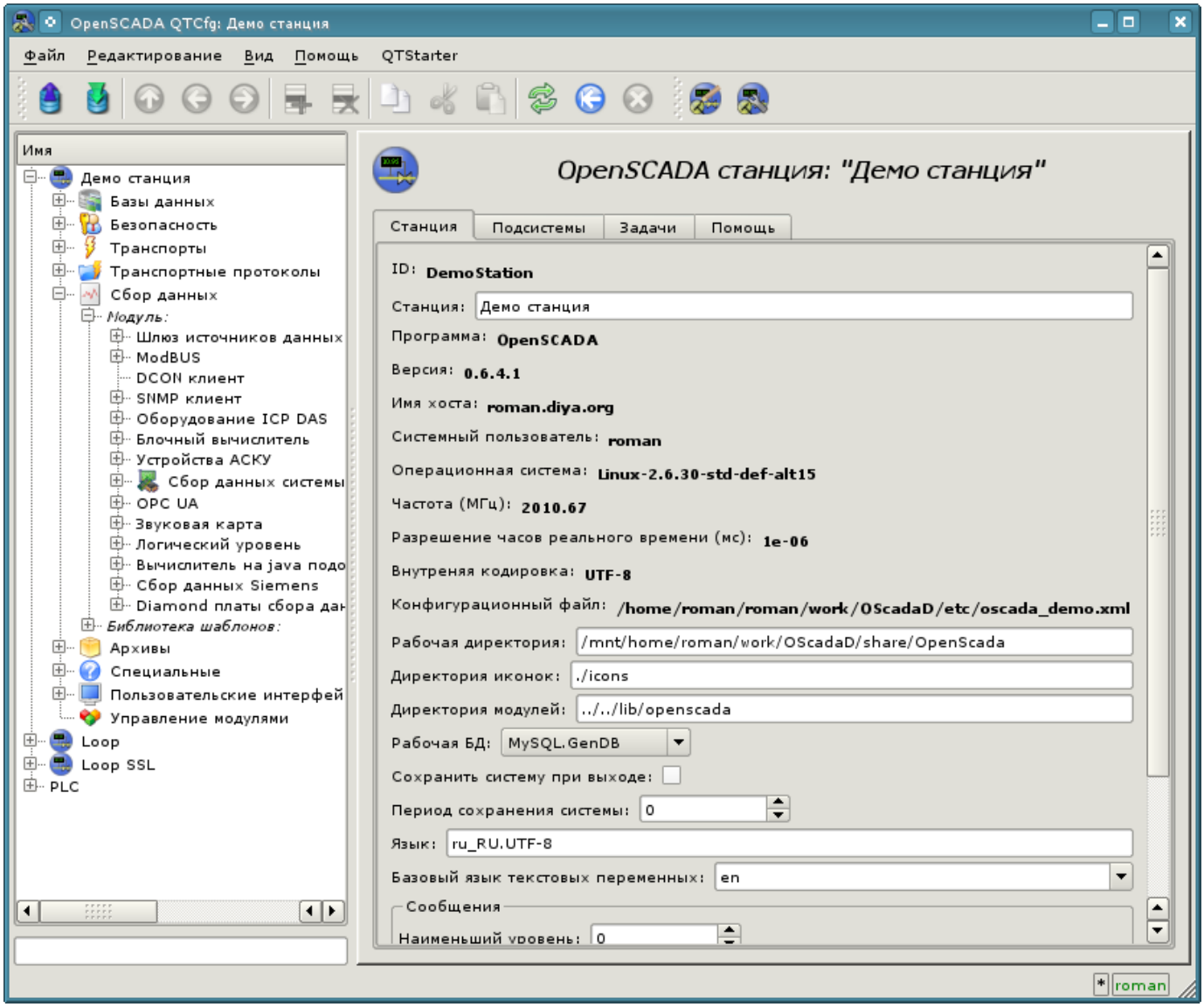


Рис. 3.2. OpenSCADA конфигуратор - QTCfg, корневая страница.

Конфигуратор OpenSCADA является основным и достаточным средством для конфигурации любого компонента системы. Как и многие компоненты OpenSCADA, конфигуратор реализован в виде модуля. Кроме конфигуратора QTCfg могут быть доступны другие конфигураторы, выполняющие те же функции, но реализованные на основе других технологиях. Например, таковыми являются Web-конфигураторы: WebCfg и WebCfgD.

Все действия в дальнейшем мы будем рассматривать только в конфигураторе QTCfg, хотя все их можно будет выполнить и в других конфигураторах.

Структуру интерфейса окна конфигуратора можно детально рассмотреть по ссылке http://wiki.oscada.org/Doc/QTCfg. Для нас же сейчас более важно рассмотреть все доступные интерфейсы OpenSCADA, поэтому нажмём предпоследнюю, с верху, иконку на панели инструментов. После нажатия на эту иконку откроется окно разработки пользовательского интерфейса (рис.3.3).

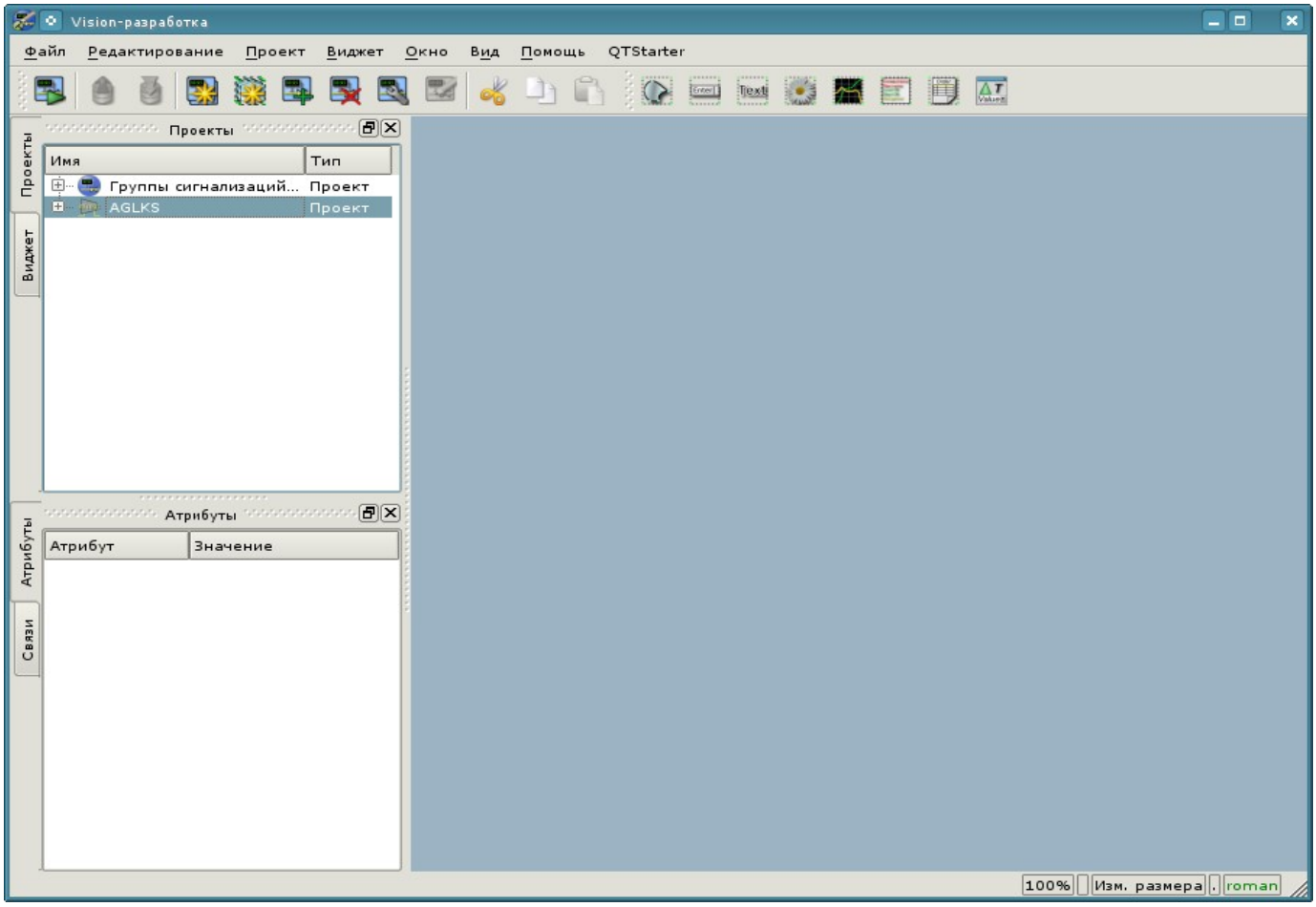


Рис. 3.3. Окно разработки пользовательского интерфейса.

Далее можем запустить проект "AGLKS" на исполнение. Для этого выбираем его в перечне проектов и запускаем путём нажатия на первую слева иконку на панели инструментов или в контекстном меню. В результате получим окно пользовательского интерфейса (рис.3.4).

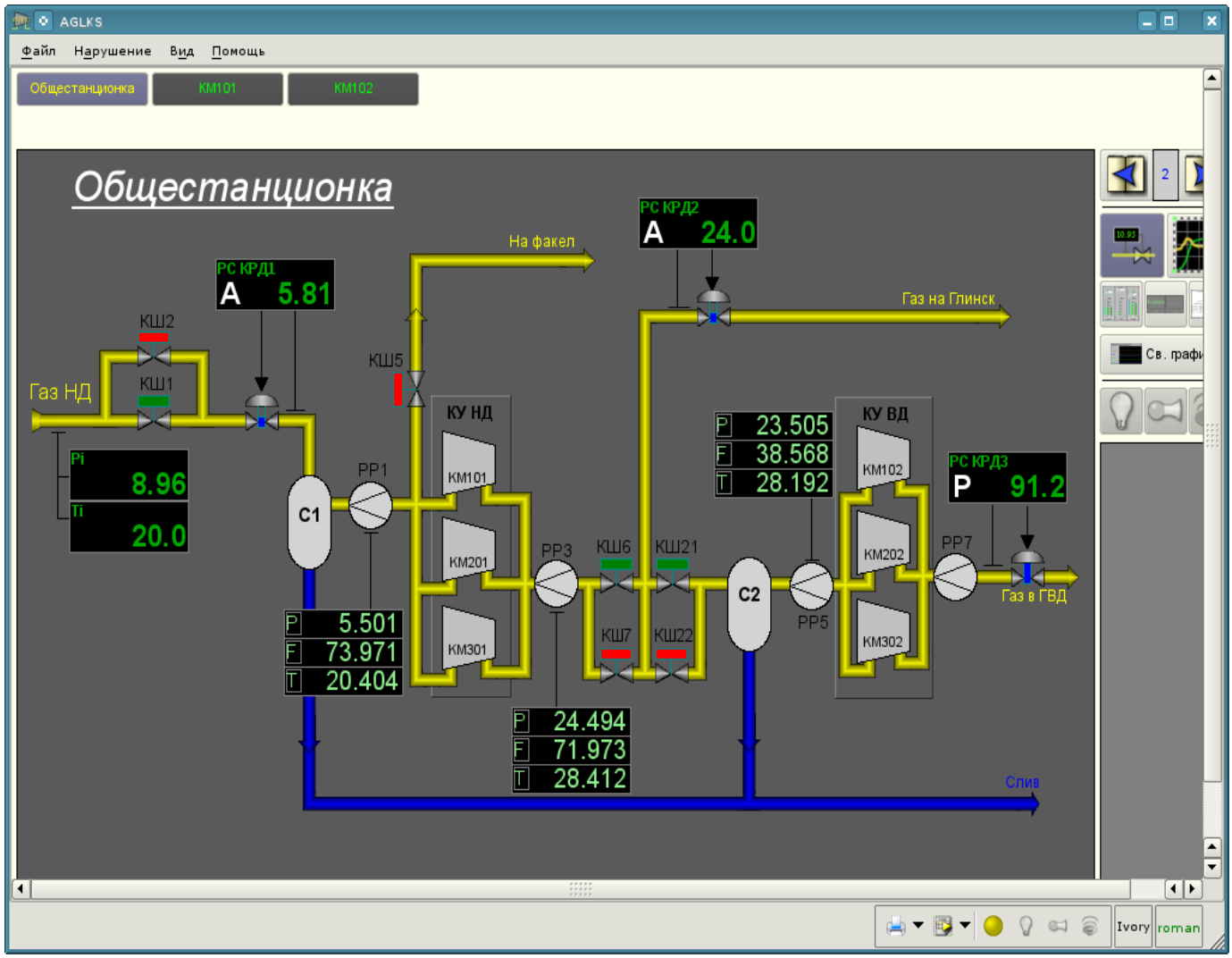


Рис. 3.4. Окно пользовательского интерфейса проекта "AGLKS".

Построение и исполнение пользовательских интерфейсов осуществляется модулем "Vision" подсистемы "Пользовательские интерфейсы". Кроме этого модуля могут быть доступны иные модули визуализации. Например, OpenSCADA предоставляет модуль WebVision, который позволяет исполнять проекты, ранее разработанные в интерфейсе модуля "Vision", посредством Web-технологий и стандартного Web-браузера. Все действия в дальнейшем мы будем рассматривать только в интерфейсе модуля "Vision".

Таким образом мы запустили демонстрацию OpenSCADA и познакомились с основным набором инструментов. В дальнейшем будем их использовать для конфигурации OpenSCADA, создания задач сбора данных, обвязки собранных данных, с целью их обработки и выдачи воздействий, а также для создания пользовательского интерфейса визуализации полученных данных и выдачи управляющих воздействий.

Закроем окно исполнения проекта "AGLKS" и окно разработки пользовательского интерфейса для подготовки к изучению следующих разделов.

Весь процесс конфигурации SCADA-системы для выполнения функций верхнего уровня можно условно разделить на два этапа:

- Конфигурация источников данных и создание базы данных (БД) параметров этих источников.
- Формирование визуального представления данных ТП путём создания интерфейса оператора в виде мнемосхем, групп графиков, групп контуров, документов и т.д.

4. Работа с источниками данных

Основной функцией любой SCADA-системы является работа с источниками данных, а именно опрос программируемых логических контроллеров (ПЛК) и простых модулей УСО. Детальнее ознакомиться с этим вопросом можно в документе "Сбор данных в OpenSCADA" по ссылке: http://wiki.oscada.org/Doc/DAQ.

Поддержка того или иного источника данных зависит от протокола или API, по которому источник свои данные предоставляет, и наличия для протокола/API модуля подсистемы "Сбор данных" в OpenSCADA. Общий перечень модулей подсистемы "Сбор данных" и документацию по ним можно получить по ссылке http://wiki.oscada.org/Doc?v=rrl#h79-4 в соответствующем разделе.

Полученные из источников данные в последствии архивируются, обрабатываются и используются для визуального представления оператору ТП.

4.1. Опрос данных аппарата ТП

В качестве примера рассмотрим и создадим опрос данных для аппарата воздушного холодильника. Демонстрационная БД содержит модель реального времени ТП компрессорной станции из шести компрессоров. Данные для двух аппаратов воздушных холодильников "AT101_1" и "AT101_2" компрессорной станции "KM101" доступны по протоколу ModBus/TCP на порту 10502.

Мы создадим контроллер опроса по протоколу ModBUS/TCP и получим эти данные, тем самым фактически реализовав задачу опроса реальных данных, поскольку от настоящего внешнего устройства наша конфигурация будет отличаться адресом этого устройства и адресами регистров ModBUS.

Для опроса данных по протоколу ModBUS/TCP в OpenSCADA присутствует модуль "ModBUS" подсистемы "Сбор данных". Для добавления нового контроллера откроем в конфигураторе страницу модуля "ModBUS" ("Демо станция"->"Сбор данных"->"Модуль"->"ModBUS") и в контекстном меню пункта "ModBUS" нажмём "Добавить" (рис.4.1.1).

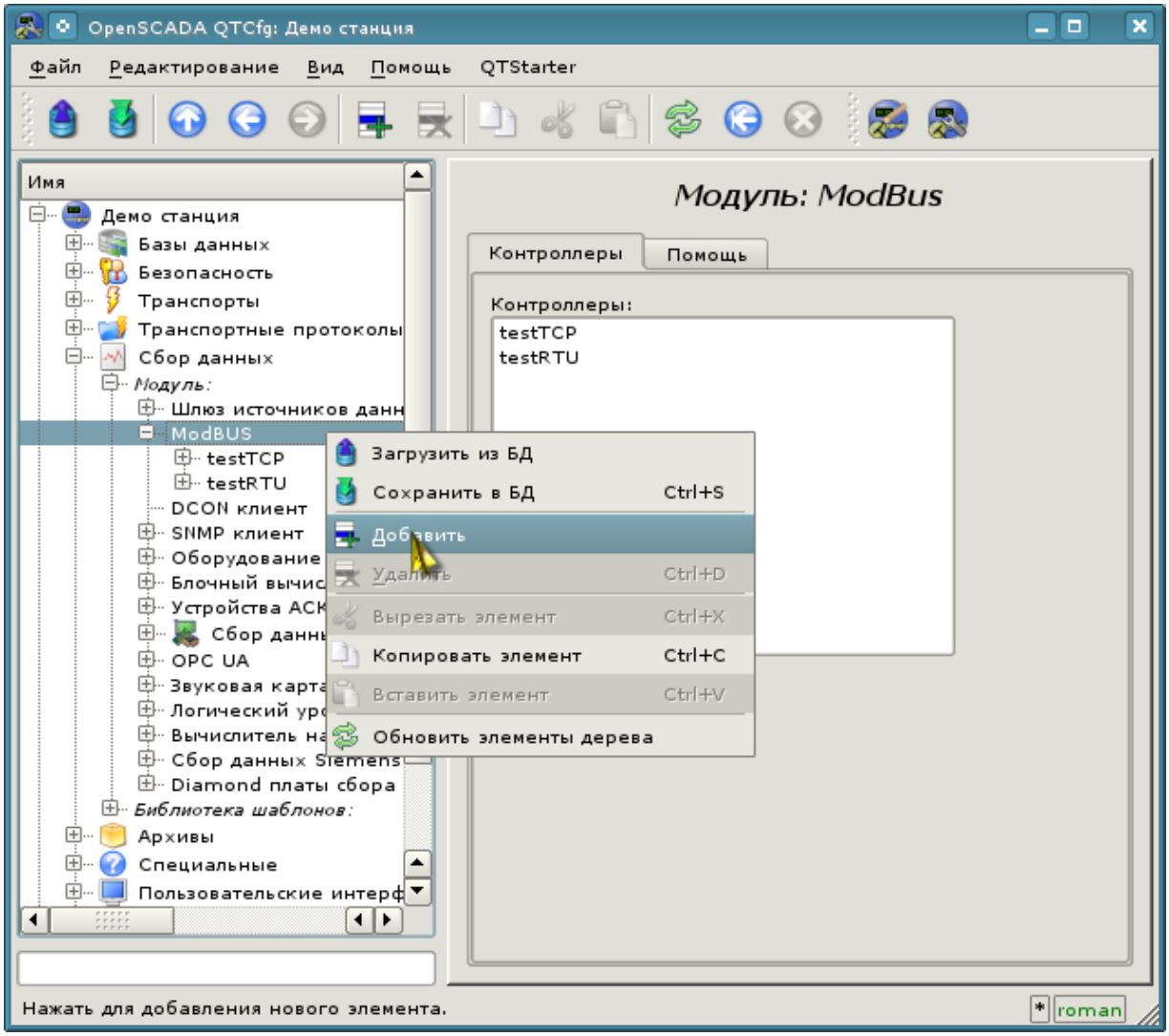


Рис. 4.1.1. Добавление контроллера в модуле "ModBUS" подсистемы "Сбор данных".

В результате появится окно диалога (рис.4.1.2) с предложением ввести идентификатор и имя нового контроллера. Идентификаторы любых объектов в OpenSCADA ограничены размером в 20 символов и их рекомендуется вводить символами английского алфавита и цифрами. Кроме этого, начинать идентификатор желательно с буквы. Это связано с тем, что идентификатор в последствии может использоваться в скриптах. Имена объектов OpenSCADA ограничены размером в 50 символов и могут вводиться любыми символами. Имена обычно используются для отображения. Если поле имени оставить пустым, то вместо него для отображения будет использоваться идентификатор. Введём идентификатор "KM101" и имя "KM 101".

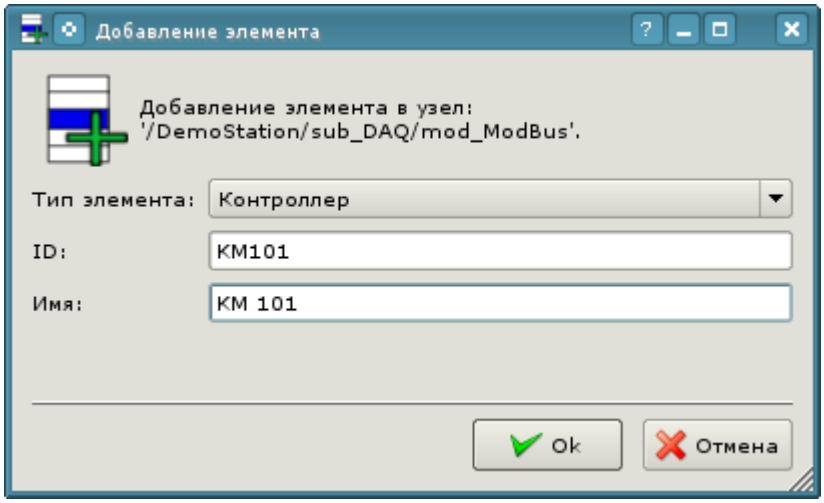


Рис. 4.1.2. Диалог для указания идентификатора и имени нового объекта.

После подтверждения у нас появится объект нового контроллера. Выберем его в конфигураторе и познакомимся с настройками (рис.4.1.3).

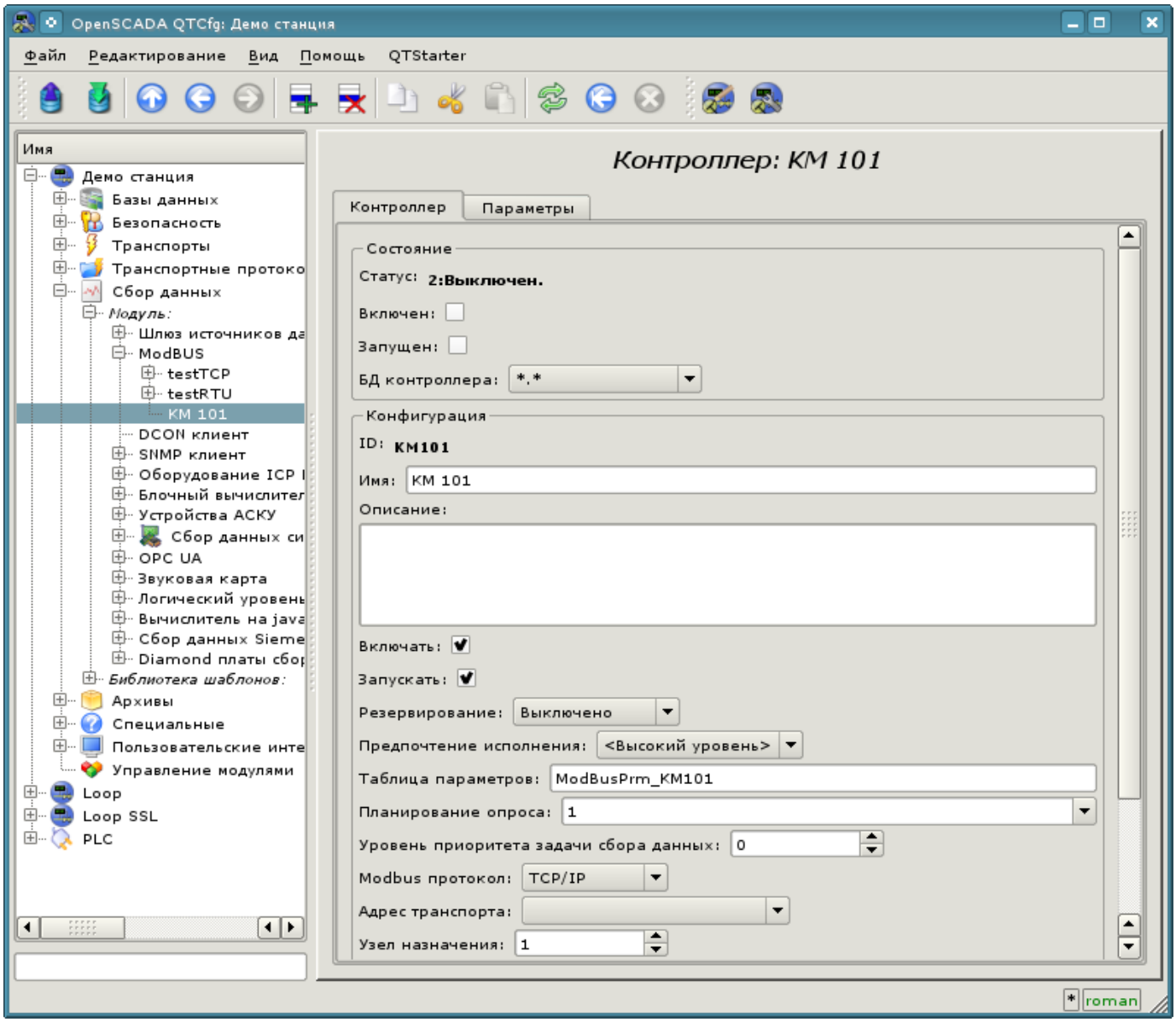


Рис. 4.1.3. Главная вкладка настройки объекта контроллера модуля ModBUS.

Настройки объекта контроллера, как правило, специфичны для разных типов источников данных и протоколов. Детально ознакомиться с настройками объекта контроллера модуля "ModBUS" можно по ссылке http://wiki.oscada.org/Doc/ModBus?v=16m6#h592-14. Мы же рассмотрим общие настройки объекта контроллера и ключевые настройки для модуля "ModBUS".

С помощью страницы объекта контроллера в разделе "Состояние" можно в первую очередь оценить текущее состояние объекта контроллера и реальное состояние связи с физическим контроллером, а также оперативно его менять. Так, поле "Статус" содержит код ошибки и текстовое описание текущего состояния связи с контроллером, в нашем случае объект контроллера выключен. Мы его можем включить и запустить, установив флажки напротив соответствующих полей. Включенный объект контролера инициализирует объекты параметров, запущенный же запускает задачу опроса и предоставляет возможность передавать данные в контроллер через атрибуты параметров. Поле БД указывает на то, в какой БД хранится конфигурация данного объекта. Нас устроит хранение в главной БД, т.е. оставим по умолчанию.

В разделе "Конфигурация" непосредственно содержится конфигурация объекта контроллера:

- "Идентификатор" и "Имя" содержат названия, которые мы вводили при создании объекта. Имя мы можем поменять прямо здесь, а вот идентификатор невозможно изменять прямо, однако объект можно вырезать (Ctrl+X) и затем вставить (Ctrl+V), тем самым переименовав

его.

- "Описание" может содержать развёрнутую характеристику и назначение объекта контроллера. В нашем случае значение этого поля не принципиально.
- "Включать" и "Запускать" указывают на то, в какое состояние переводить объект контроллера при запуске OpenSCADA. Установим оба поля.
- Два следующих поля "Резервирование" и "Предпочтение исполнения" служат для конфигурации горизонтального резервирования источников данных на разных станциях (http://wiki.oscada.org/Doc/DAQ?v=1dv7#h831-10). Оставим их как есть.
- "Таблица параметров" - содержит имя таблицы БД, в которой будет храниться конфигурация параметров данного контроллера. Оставим по умолчанию.
- "Планирование опроса" - содержит конфигурацию планировщика для запуска задачи опроса. Получить описание формата конфигурации данного поля можно из всплывающей подсказки. Одиночная цифра указывает на периодичность запуска в секундах. Оставим одну секунду.
- "Уровень приоритета задачи сбора данных" - указывает насколько приоритетна данная задача (от -1 до 99). Приоритеты выше нуля имеют смысл только при запуске OpenSCADA от привилегированного пользователя. Оставим это поле без изменений.
- "ModBUS протокол" - указывает на вариант протокола ModBUS. Нас интересует вариант "TCP/IP", поэтому оставим как есть.
- "Адрес транспорта" - указывает на исходящий транспорт подсистемы "Транспорты", который используется для соединения с контроллером. В случае с вариантом "TCP/IP" нам нужен транспорт в модуле "Sockets". На создании исходящего транспорта в "Sockets" детальнее остановимся ниже.
- "Узел назначения" - указывает узел ModBUS. В нашем случае это должно быть "1".
- "Объединять фрагменты данных" - включает объединение не смежных фрагментов регистров в один блок запроса, до 100 регистров, вместо генерации отдельных запросов. Позволяет уменьшить общее время опроса. Установим эту опцию.
- "Время ожидания соединения" - указывает, в течение какого времени ожидать ответа от контроллера и по истечению которого сообщать об ошибке связи. Ноль указывает на использование времени транспорта. Оставим без изменений.
- "Время восстановления" - Указывает на время в секундах, через которое в случае отсутствия связи повторять попытку восстановить соединение.

Сохраним наши изменения в БД, нажав вторую слева иконку на панели инструментов.

Теперь таким же образом, как и объект контроллера, создадим исходящий транспорт в модуле "Sockets" ("Демо станция"->"Транспорты"->"Сокеты") посредством контекстного меню (рис.4.1.4). И назовём транспорт так же, как контроллер: "KM101" и имя "KM 101". Обратите внимание, что в поле "Тип элемента" диалога ввода идентификатора и имени (рис.4.1.2) нужно выбрать "Выходной транспорт".

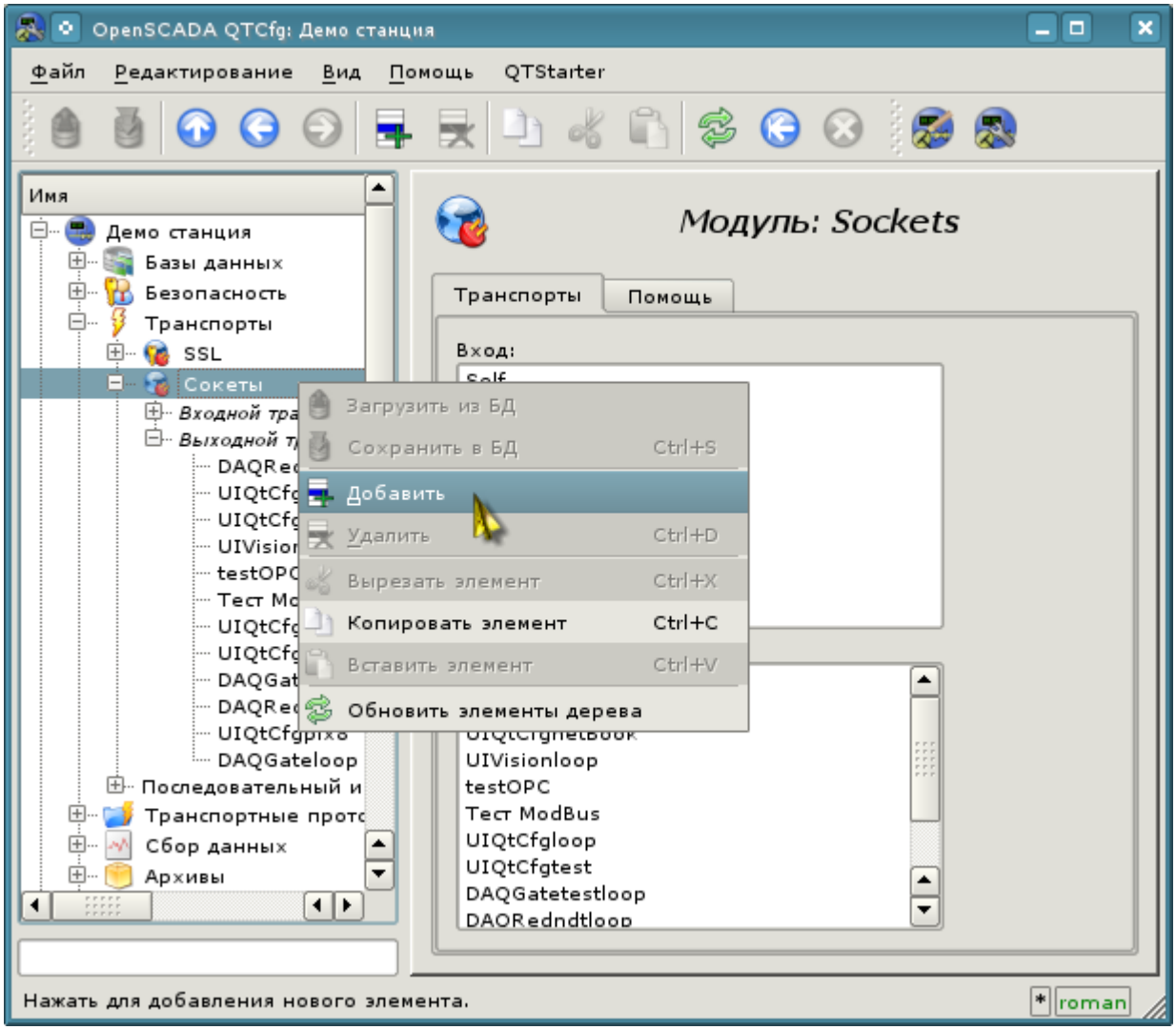


Рис. 4.1.4. Добавление исходящего транспорта в модуле "Sockets" подсистемы "Транспорты".

Страница конфигурации полученного исходящего транспорта приведена на рис.4.1.5. Эта страница также содержит раздел состояния и оперативного управления. В поле "Статус" содержится текстовое описание текущего состояния транспорта. Мы его можем запустить на исполнение, установив флажок напротив соответствующего поля. Выполняющийся объект транспорта инициирует соединение с внешним узлом. Поле БД указывает на то, в какой БД хранится конфигурация данного объекта. Нас устроит хранение в главной БД.

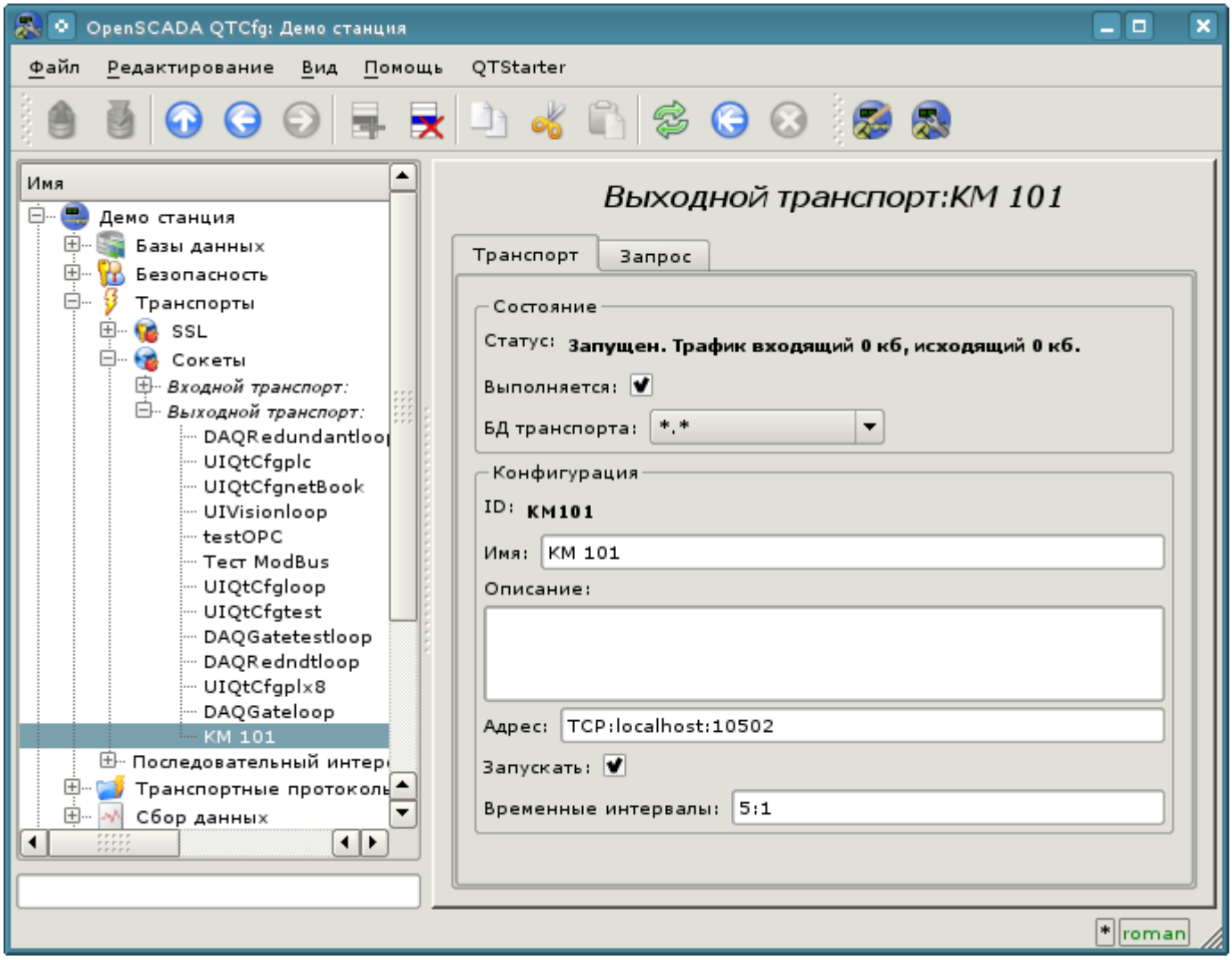


Рис. 4.1.5. Страница конфигурации исходящего транспорта модуля "Sockets" подсистемы "Транспорты".

В разделе "Конфигурация" непосредственно содержится конфигурация объекта транспорта:

- "Идентификатор" и "Имя" содержат названия, которые мы вводили при создании объекта.
- "Описание" может содержать развёрнутую характеристику и назначение объекта.
- "Адрес" указывает тип, адрес и режим соединения с удалённой станцией. Ознакомиться с форматом записи можно из всплывающей подсказки. Установим это поле в значение "TCP:localhost:10502".
- "Запускать" указывает на то, в какое состояние переводить объект при запуска OpenSCADA. Установим поля.
- "Временные интервалы" указывают продолжительность ожидания ответа от удалённой станции. Ознакомиться с форматом записи можно из всплывающей подсказки. Оставим значение неизменным.

Сохраним объект транспорта и вернёмся к конфигурационному полю "Адрес транспорта" объекта контроллера, где выберем адрес "Sockets.KM101". На этом настройка объекта контроллера закончена. Следующим этапом является конфигурация и выбор тех данных, которые нужно опрашивать из контроллера. Эта настройка производится путём создания объекта "Параметр" контроллера. Объект "Параметр" позволяет описать перечень данных, получаемых у контролера и

передать их в окружение OpenSCADA.

Для добавления нового объекта параметра откроем в конфигураторе страницу нашего объекта контроллера и в контекстном меню пункта "KM101" нажмём "Добавить". Объект параметра назовём: "AT101_1" и имя "AT 101_1".

Страница конфигурации полученного параметра приведена на рис.4.1.6. Эта страница содержит раздел состояния и оперативного управления. В поле "Тип" содержится идентификатор типа параметра, в нашем случае возможен только тип "Стандартный" (std). Параметр мы можем включить, установив флажок напротив соответствующего поля. Включенный параметр участвует в процессе обмена с контроллером.

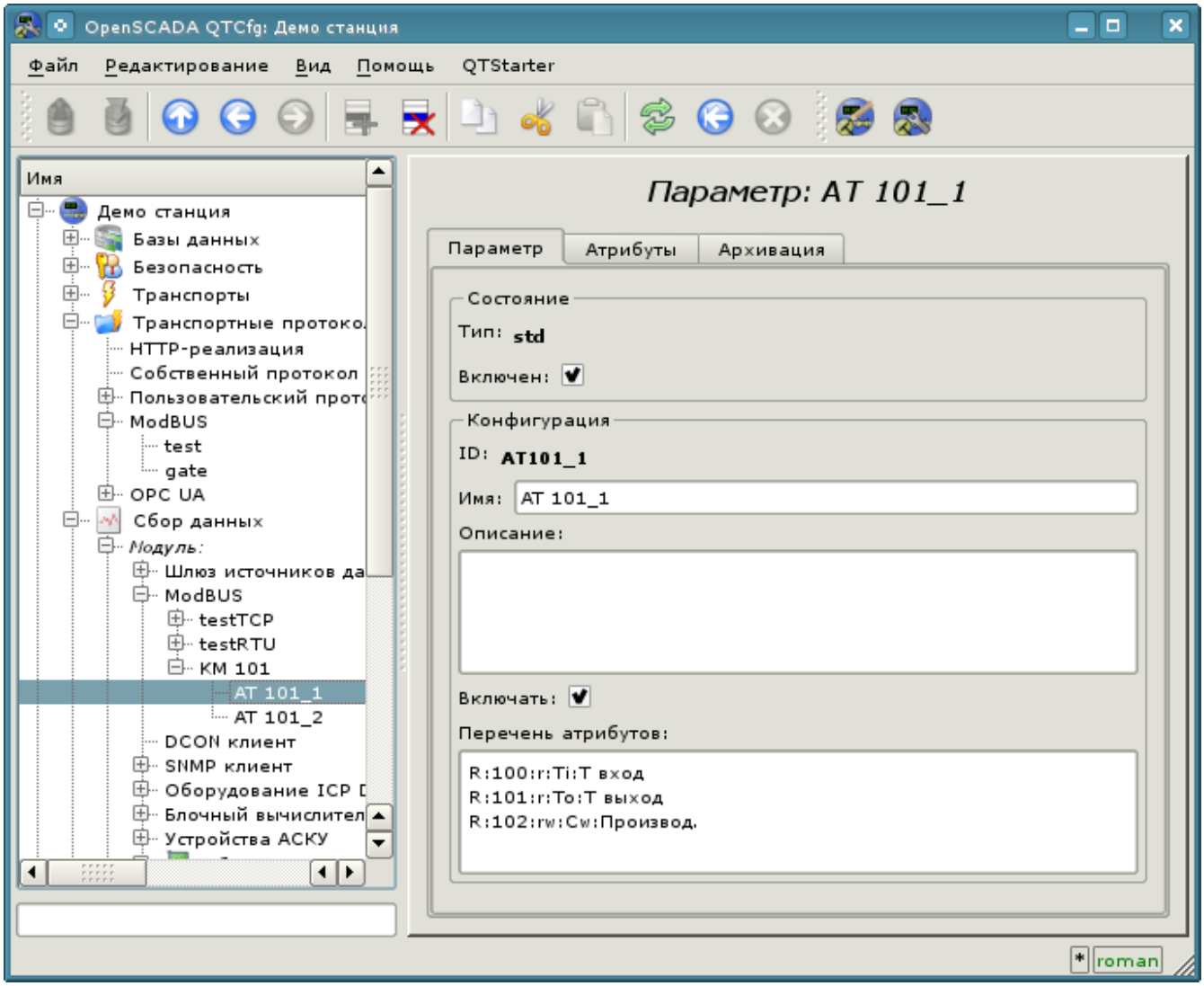


Рис. 4.1.6. Страница конфигурации параметра контроллера "ModBUS".

В разделе "Конфигурация" непосредственно содержится конфигурация объекта параметра:

- "Идентификатор" и "Имя" содержат названия, которые мы вводили при создании объекта.
- "Описание" может содержать развёрнутую характеристику и назначение объекта.
- "Включать" указывает на то, в какое состояние переводить объект при запуска OpenSCADA. Установим поле.
- "Перечень атрибутов" содержит конфигурацию атрибутов параметров в соотношении их с регистрами и битами ModBUS. Ознакомиться с форматом записи можно из всплывающей подсказки. Установим содержимое этого текстового поля в:

```
R:100:r:Ti:T вход
R:101:r:To:T выход
R:102:rw:Cw:Производ.
```

Таким же образом создадим второй параметр: "AT101_2" с именем "AT 101_2". Перечень атрибутов для него установим в:

```
R:103:r:Ti:T вход
R:104:r:To:T выход
R:105:rw:Cw:Производ.
```

Сохраним оба объекта параметра. Теперь мы можем включить и запустить наш контроллер для инициации обмена. Для этого вернёмся на страницу нашего объекта контроллера и в разделе "Состояние" установим флажёк "Запущен". Если мы ничего не пропустили, то обмен успешно запустится и в поле "Статус" мы получим подобное представленному на рис.4.1.7.

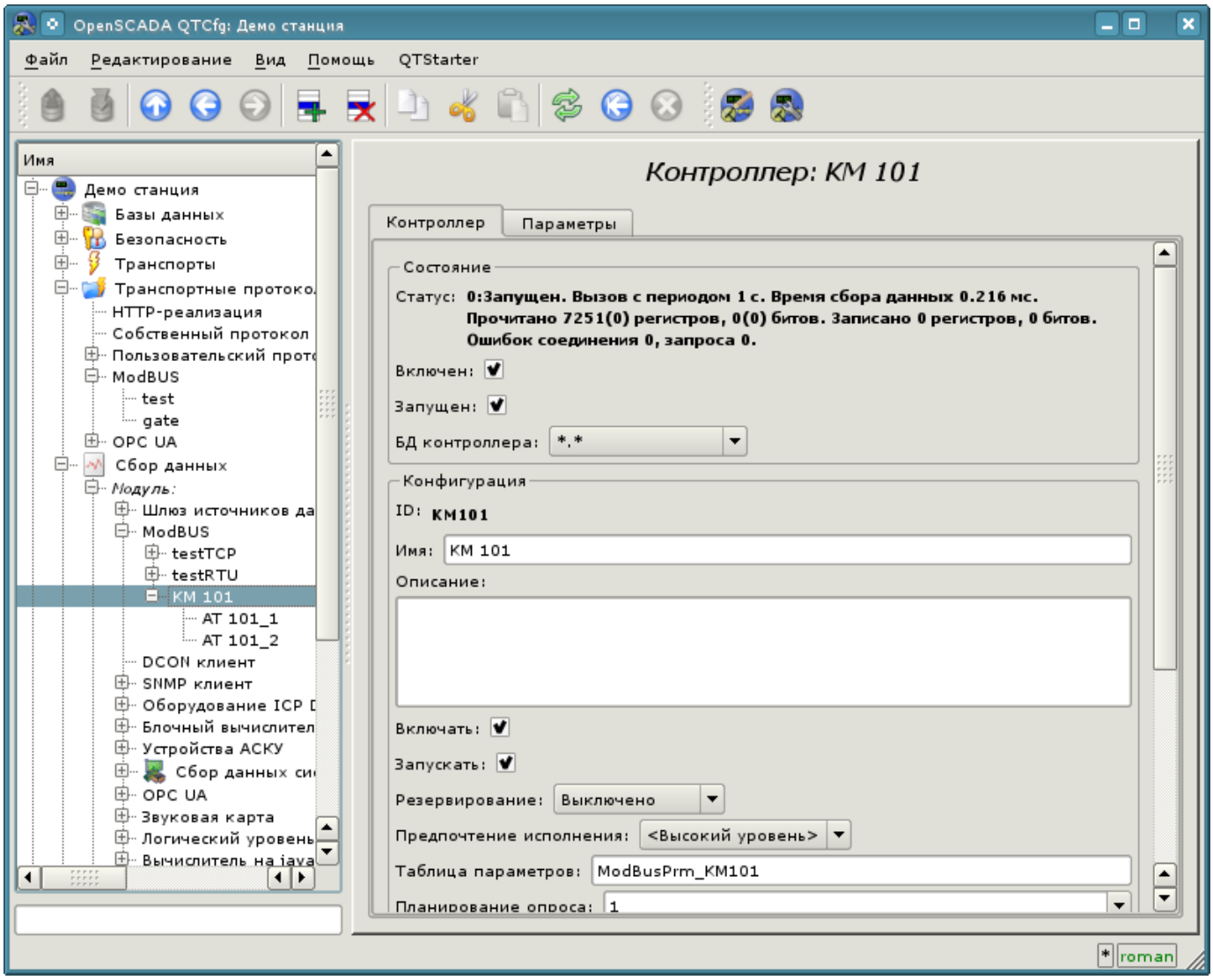


Рис. 4.1.7. Страница объекта контроллера при успешном обмене с физическим контроллером.

В случае успешного обмена с физическим контроллером мы получим описанные данные контроллера в инфраструктуре OpenSCADA. Увидеть эти данные можно на вкладке "Атрибуты" наших параметров AT101_1 (рис.4.1.8) и AT101_2. Поскольку опрос производится регулярно и с периодичностью в секунду, то мы можем наблюдать их изменение, нажимая кнопку "Обновить текущую страницу" на панели инструментов.

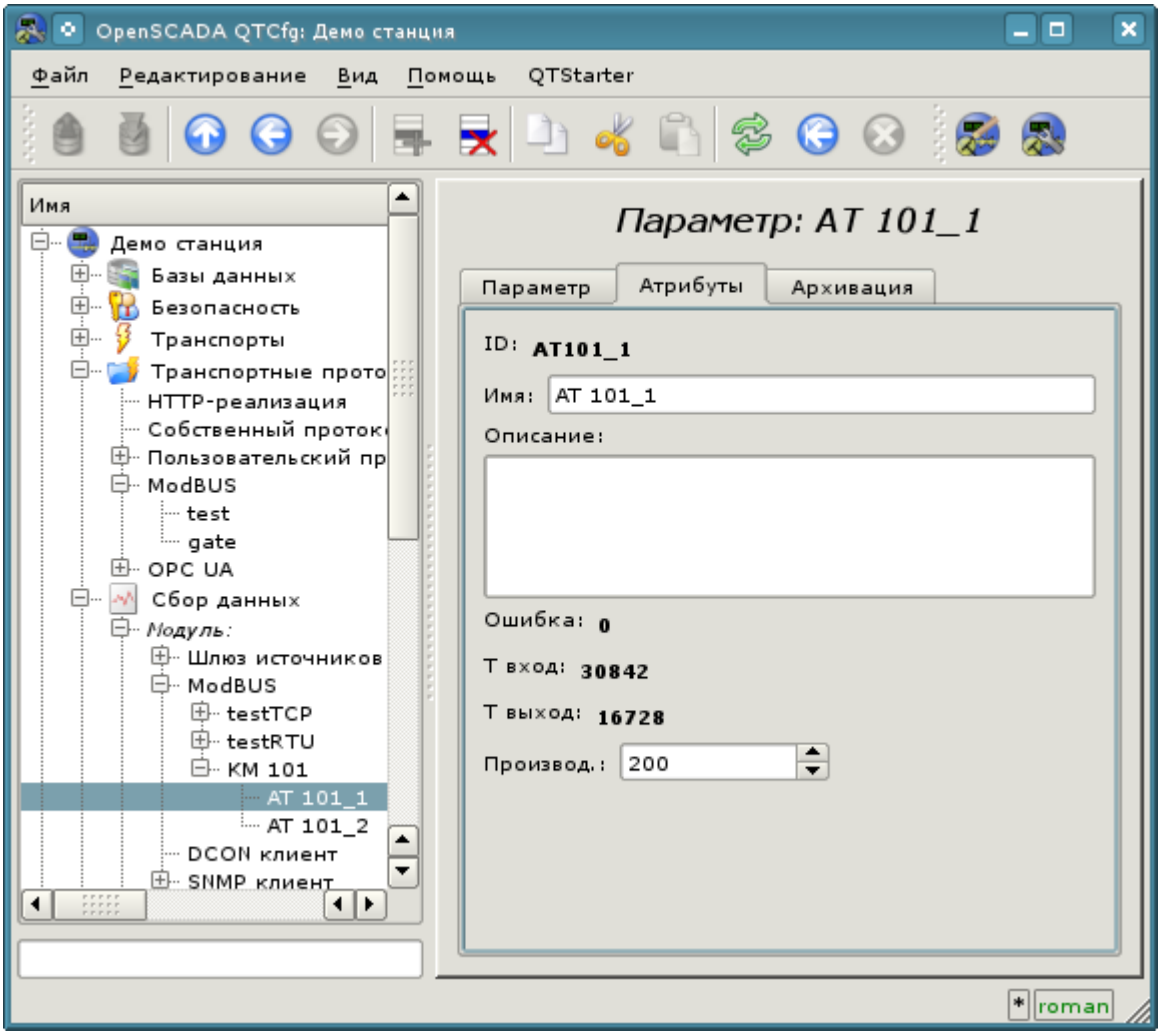


Рис. 4.1.8. Страница описанных атрибутов параметра AT101_1.

На этом конфигурация сбора данных считается законченной.

4.2. Обработка полученных данных ТП

Часто встречается ситуация, когда исходные данные, полученные из источника данных, являются "сырыми", т.е. неготовыми или неудобными для визуального представления, поэтому необходимо выполнить эту подготовку. В нашем примере мы получили данные, которые поступают в коде от шкалы внутри контроллера. Наша задача - выполнить вычисление реальных значений из полученных данных. Обработку данных в OpenSCADA можно делать, как непосредственно при визуализации, так и в подсистеме "Сбор данных". Однако, смешивание процесса визуализации и обработки исходных данных вносит путаницу в конфигурацию, а также делает полученные образы визуализации мало пригодными для повторного использования. По этой причине выполним подготовку данных в подсистеме "Сбор данных".

Вычисления в подсистеме "Сбор данных" выполняются посредством модуля логического уровня "LogicLev" и шаблонов параметров подсистемы "Сбор данных". Детальнее ознакомиться с концепцией "Логического уровня" можно по ссылке http://wiki.oscada.org/Doc/DAQ?v=1dv7#h831-9 .

Для выполнения вычислений в модуле логического уровня нужно предварительно создать шаблон параметра подсистемы "Сбор данных". Для этого откроем страницу библиотеки шаблонов "Базовые шаблоны" ("Демо станция"->"Сбор данных"->"Библиотека шаблонов"->"Базовые

шаблоны") и посредством контекстного меню создадим объект шаблона "airCooler" с именем "Воздушный холодильник". Страница конфигурации полученного объекта изображена на рис.4.2.1. Эта страница содержит раздел состояния и оперативного управления. Мы можем сделать шаблон доступным, установив флажок напротив соответствующего поля. Доступные шаблоны могут подключаться к параметрам сбора данных, а параметры будут выполнять вычисления по этому шаблону. В поле "Использовано" указывается число объектов, которые используют данный шаблон для вычисления образа параметра. В разделе "Конфигурация" содержаться только уже знакомые нам поля конфигурации.

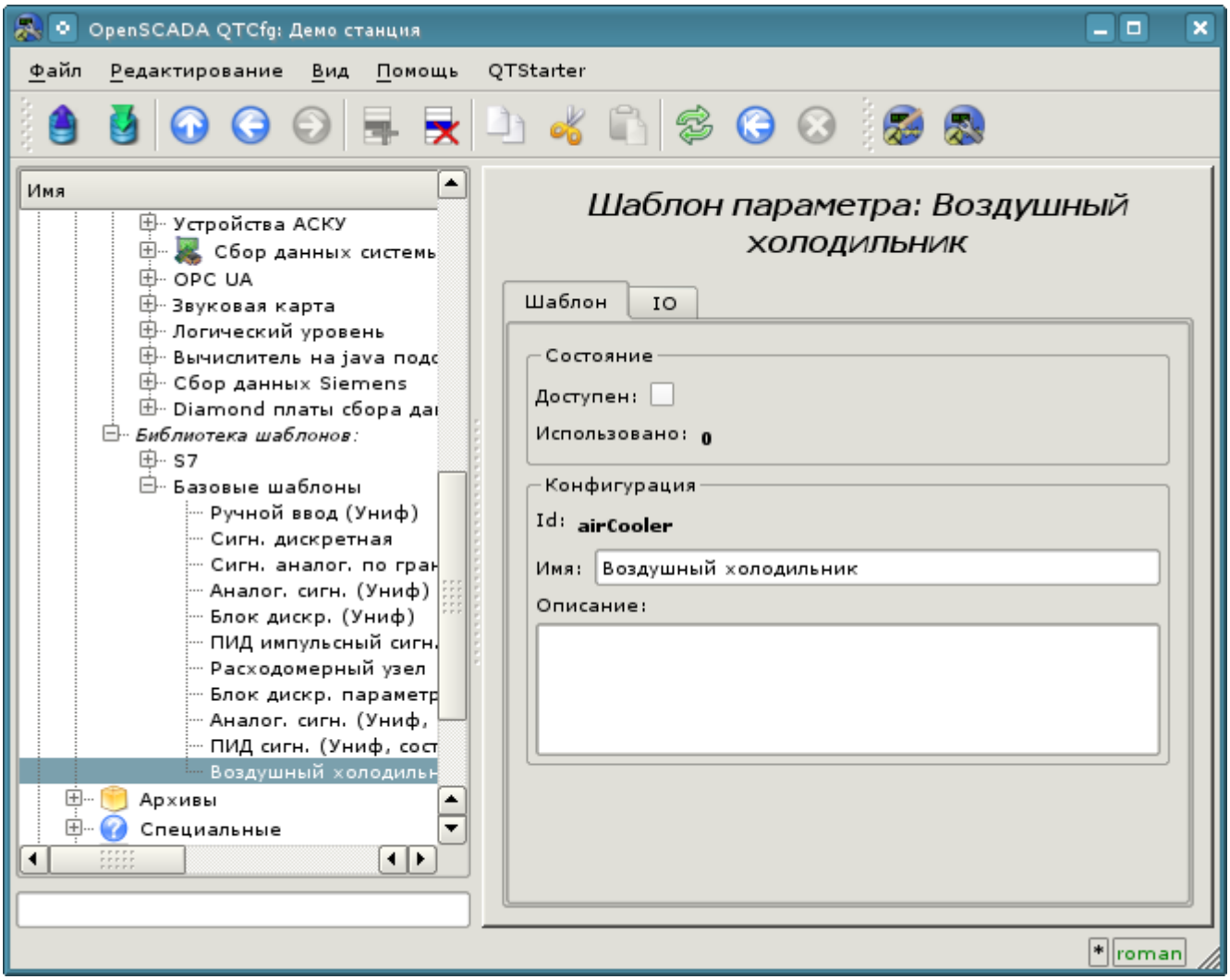


Рис. 4.2.1. Страница конфигурации объекта шаблона.

Основная конфигурация и формирование шаблона параметра сбора данных осуществляется во вкладке "IO" (рис.4.2.2) шаблона. Детальное описание процесса формирования шаблона можно получить по ссылке: http://wiki.oscada.org/Doc/OpisanieProgrammy?v=14os#h827-6 .

Создадим в шаблоне два свойства для входов ("TiCod", "ToCod"), два для выходов ("Ti","To") и один прозрачный ("Cw"). Свойствам "TiCod", "ToCod" и "Cw" установим флаг "Конфигурация" в "Связь", что позволит к ним подвязывать сырой источник. Параметрам "Ti" и "To" установим флаг "Атрибут" в "Только чтение", а "Cw" в "Полный доступ" для формирования трёх атрибутов: два только на чтение и один на полный доступ у результирующего параметра сбора данных.

Язык программы установим в "JavaLikeCalc.JavaScript", а программу в:

```
Ti=150*TiCod/65536;
To=100*ToCod/65536;
```

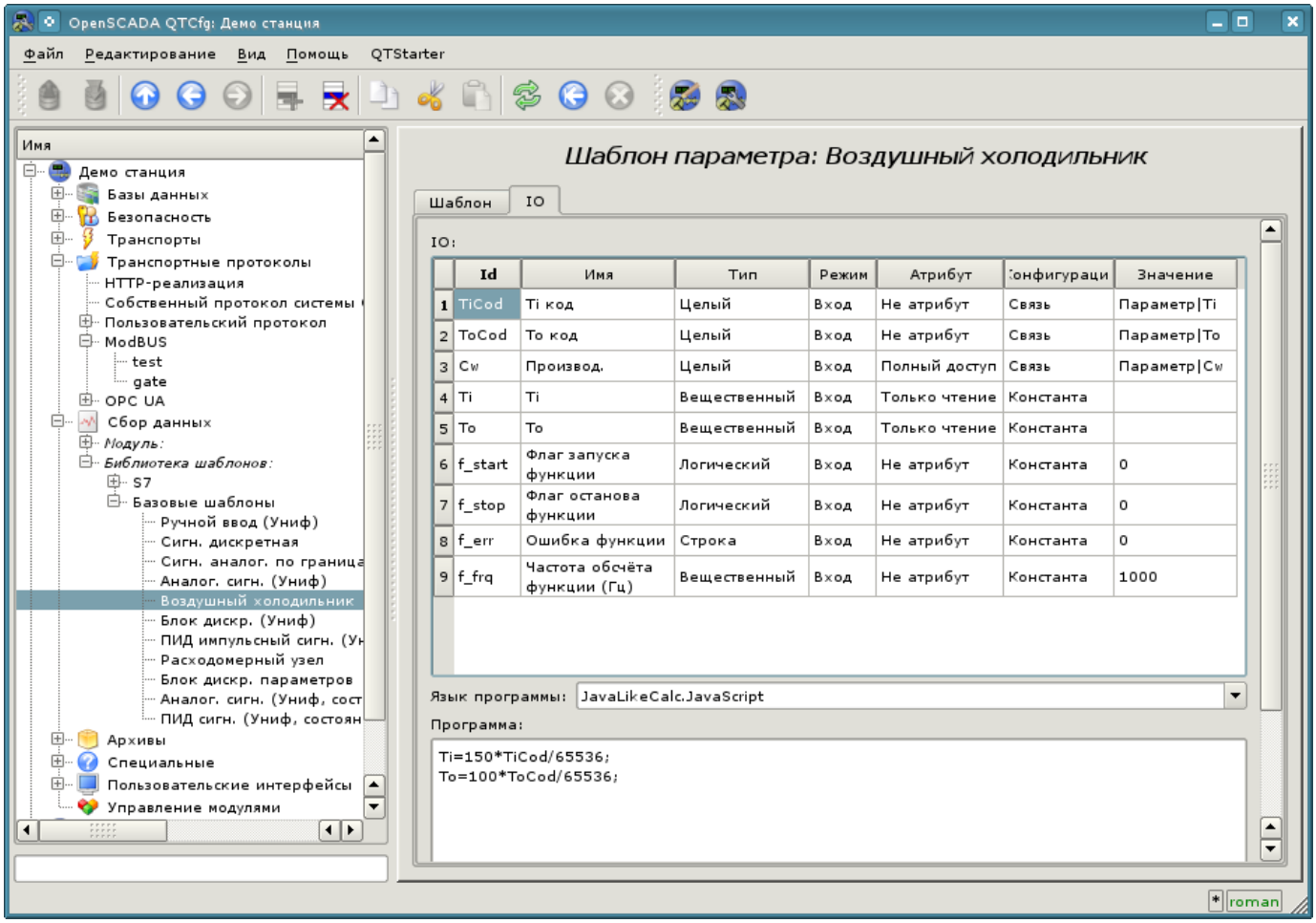


Рис. 4.2.2. Вкладка "IO" страницы конфигурации объекта шаблона.

Полученный шаблон сохраним и установим флаг доступности.

Теперь создадим объекты контроллера и параметров в модуле "LogicLev" подсистемы "Сбор данных". Контроллер и его параметры в модуле "LogicLev" создаются идентично ранее созданным в модуле "ModBUS" на странице: "Демо станция"->"Сбор данных"->"Модуль"->"Логический уровень". Объект контроллера и параметров назовём идентично объектам в модуле "ModBUS".

Объект контроллера модуля "LogicLev" (рис.4.2.3) не имеет специфических настроек и установленные по умолчанию можно не трогать.

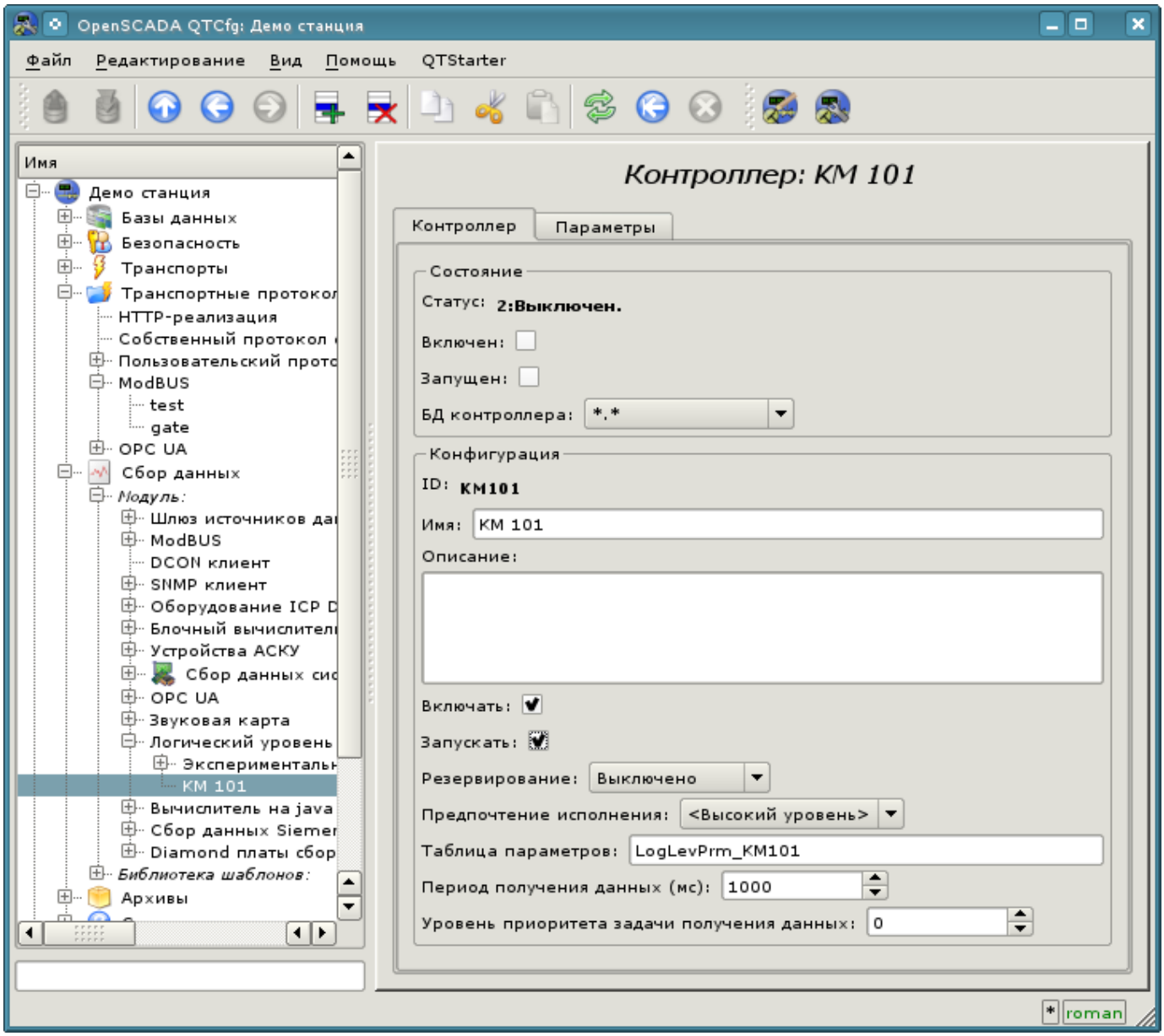


Рис. 4.2.3. Главная вкладка настройки объекта контроллера модуля LogicLev.

Объект параметра контроллера модуля "LogicLev" (рис.4.2.4) из специфических настроек имеет только "Режим", где нужно установить "Шаблон", а справа выбрать адрес созданного нами шаблона.

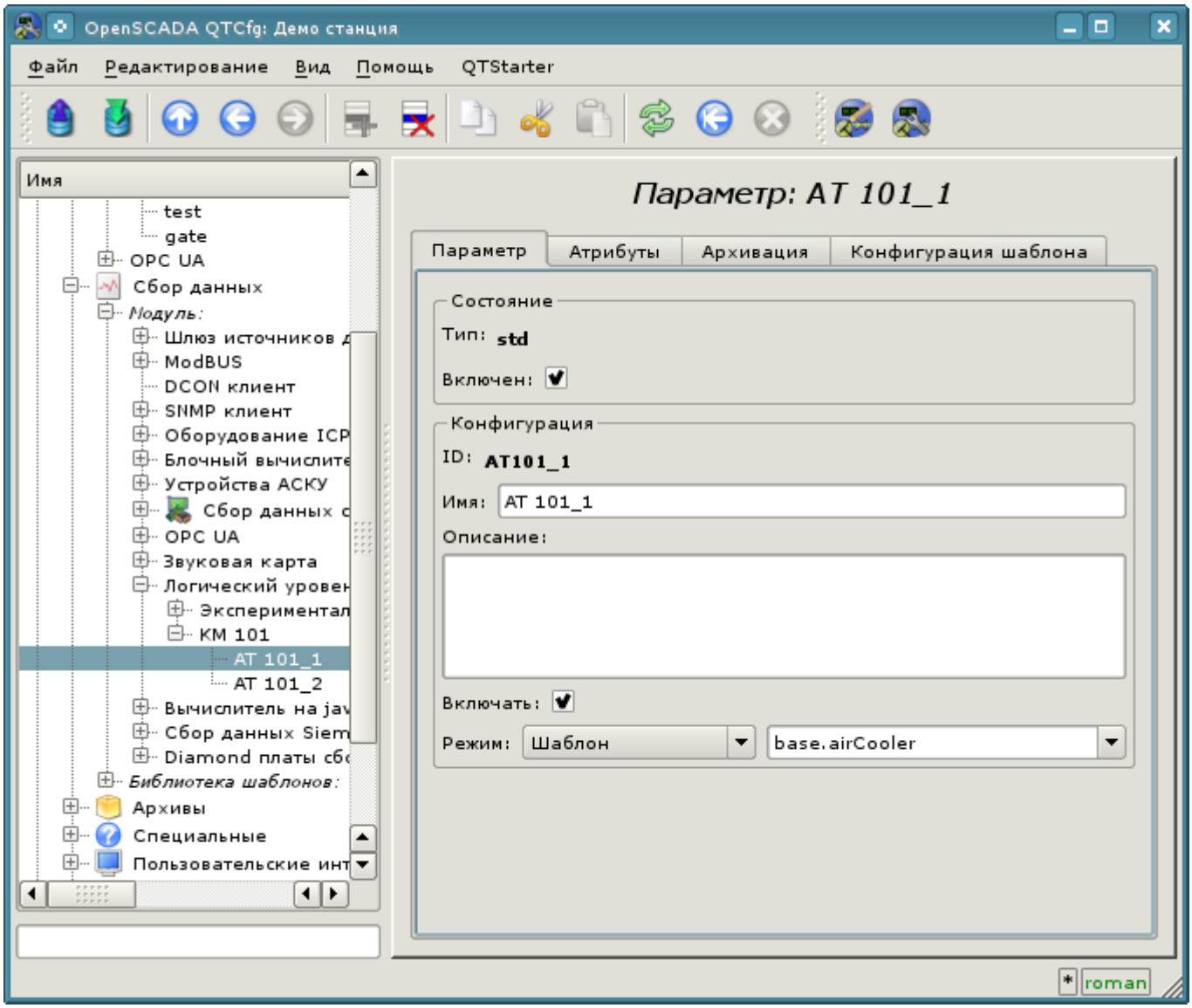


Рис. 4.2.4. Страница конфигурации параметра контроллера "LogicLev".

Кроме базовой конфигурации параметра нужно сконфигурировать подключенный шаблон (рис. 4.2.5). Вкладка конфигурации шаблона появляется в режиме параметра "Включен". Включить параметр можно, включив предварительно контроллер. Флаг "Показывать только атрибуты" позволяет устанавливать отдельно каждую связь (рис.4.2.6). Поскольку мы в шаблоне предусмотрительно описали формат связей в виде "Параметр|Ti", то все три связи мы можем установить просто указав адрес к параметру в контроллере "ModBus". Укажем адреса "ModBus.KM101.AT101_1" и "ModBus.KM101.AT101_2" в соответствующих параметрах.

Нужно отметить, что все поля ввода адресов объектов в OpenSCADA снабжены механизмом набора адреса. Данный механизм подразумевает поэлементный выбор, в процессе которого происходит движение в глубь. Например, при наборе адреса "ModBus.KM101.AT101_1" вначале мы будем иметь возможность выбора типа источника данных, в числе которых будет "ModBus". Выбрав пункт "ModBus" в перечень доступных элементов для выбора добавятся контроллеры модуля "ModBus", в числе которого будет "ModBus.KM101". Выбор элемента "ModBus.KM101" добавит в список параметры контроллера и т.д. до конечного элемента в соответствии с иерархией объектов (http://wiki.oscada.org/Doc/OpisanieProgrammy?v=gax#h827-6). Для возможность возврата на уровни выше в список выбора вставляются элементы всех вышестоящих уровней от текущего значения адреса.

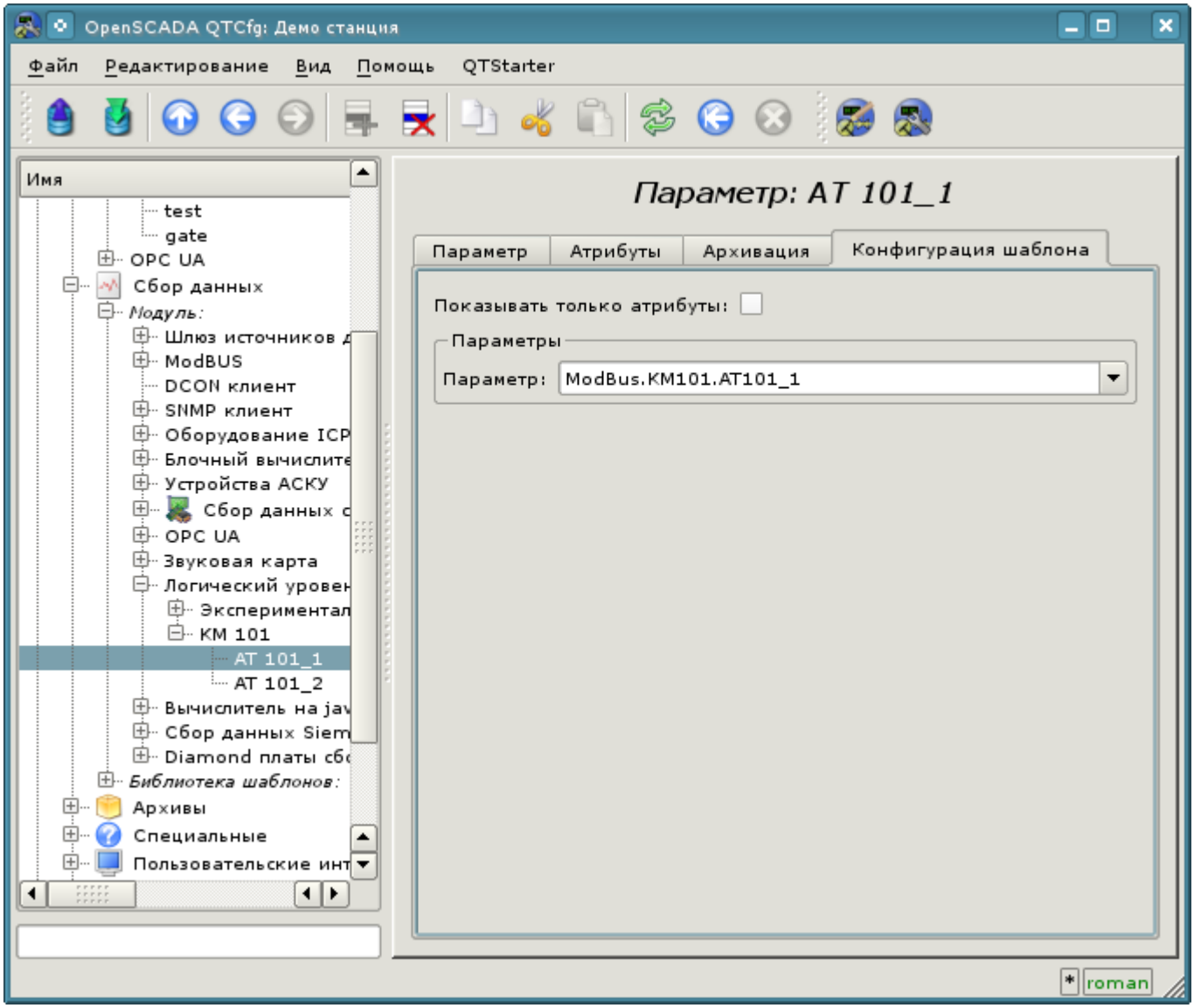


Рис. 4.2.5. Вкладка "Конфигурация шаблона" страницы параметра контроллера "LogicLev".

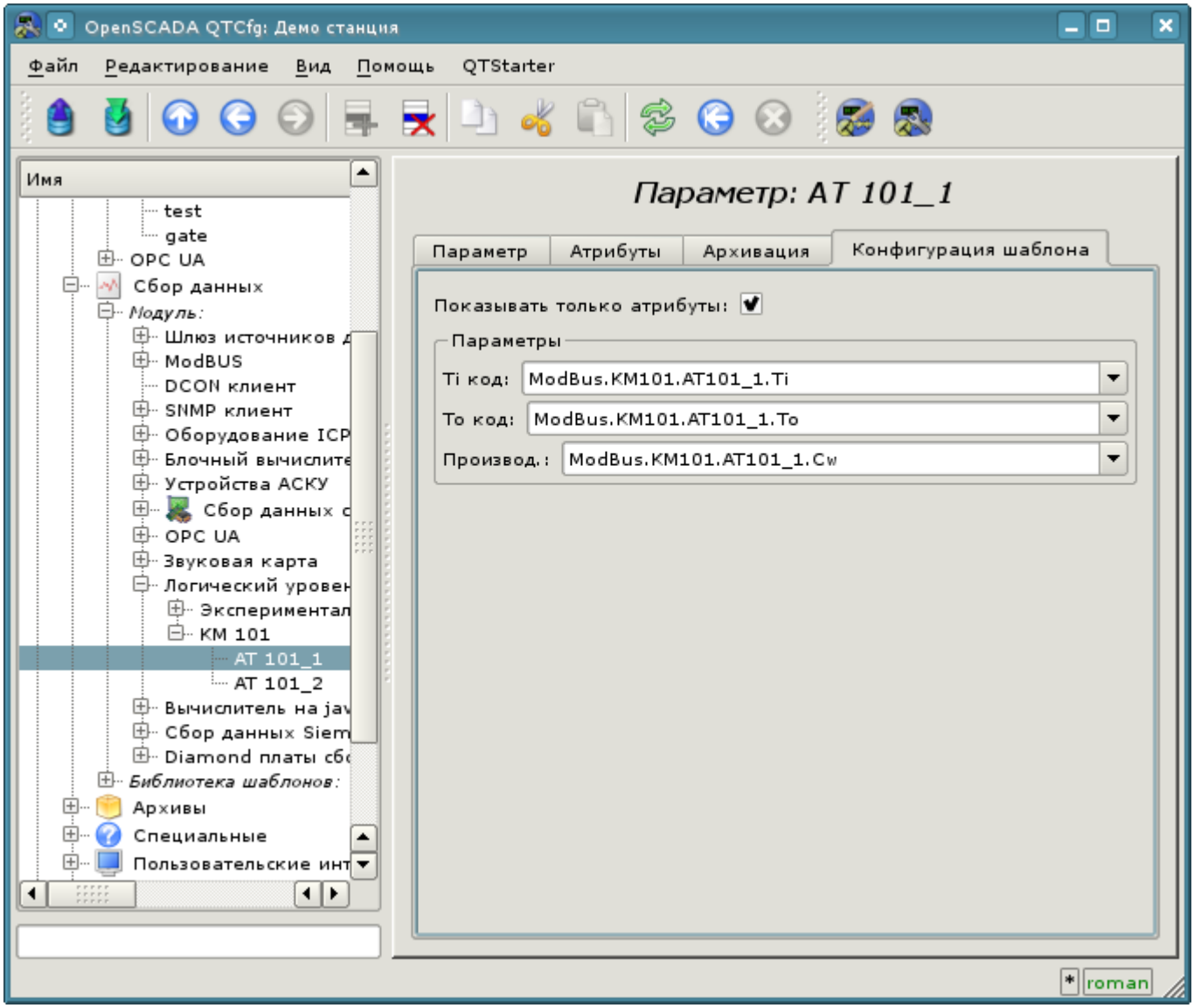


Рис. 4.2.6. Вкладка "Конфигурация шаблона" страницы параметра контроллера "LogicLev" с разворотом связей.

Сохраним созданные объекты контроллера и параметров. После этого запустим контроллер на исполнение, установив флаг контроллера "Запущен" в разделе "Состояние". Если мы ничего не пропустили, то вычисление успешно запустится и в поле "Статус" мы получим подобное представленному на рис.4.2.7.

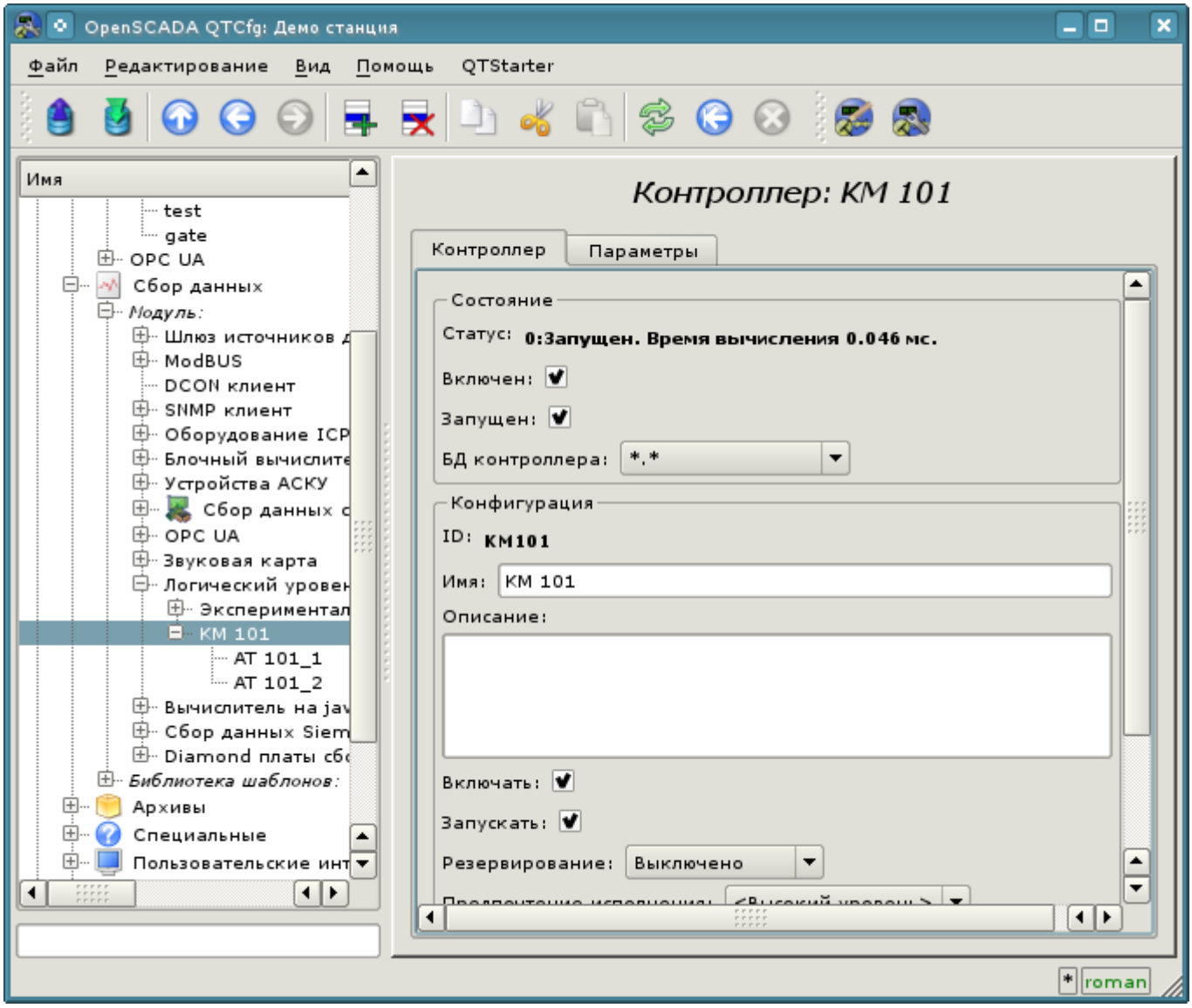


Рис. 4.2.7. Страница объекта контроллера при успешном вычислении контроллера в модуле "LogicLev".

В случае успешной обработки кода шаблона в параметрах мы получим обработанные данные в инфраструктуре OpenSCADA. Увидеть эти данные можно на вкладке "Атрибуты" наших параметров AT101_1 (рис.4.2.8) и AT101_2.

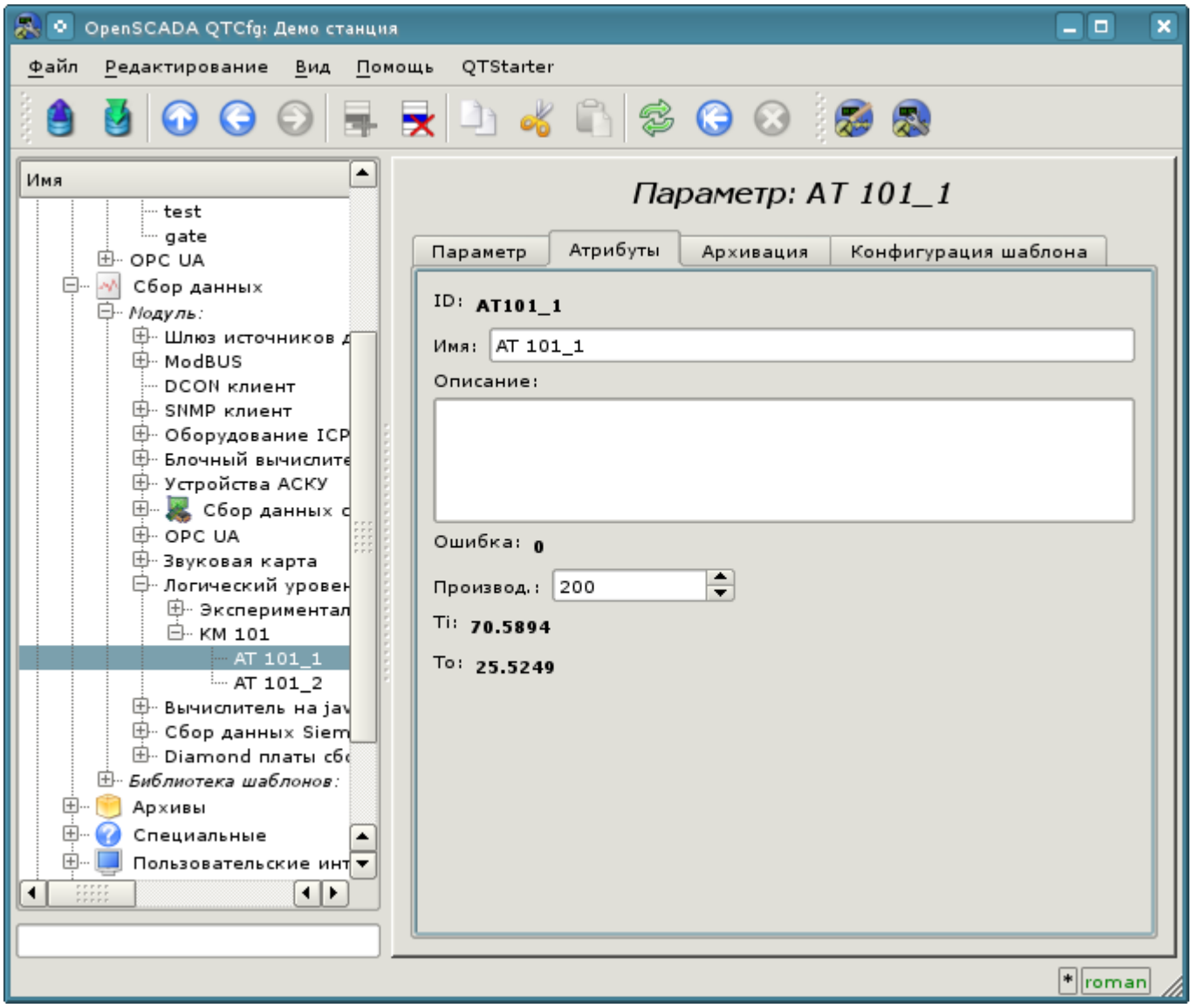


Рис. 4.2.8. Страница атрибутов параметра AT101_1 модуля "LogicLev".

На этом конфигурация обработки данных считается законченной.

4.3. Включение архивирования данных ТП

Для многих задач требуется ведение истории параметров ТП. Для включения архивирования атрибутов "Ti" и "To" параметров AT101_1 и AT101_2 в ранее созданном контроллере модуля "LogicLev" достаточно на вкладке "Архивация" страницы параметров выбрать, какие атрибуты архивировать и на каких архиваторах (рис.4.3.1). Выберем архивацию атрибутов "Ti" и "To" в архиваторе "FSArch.1s".

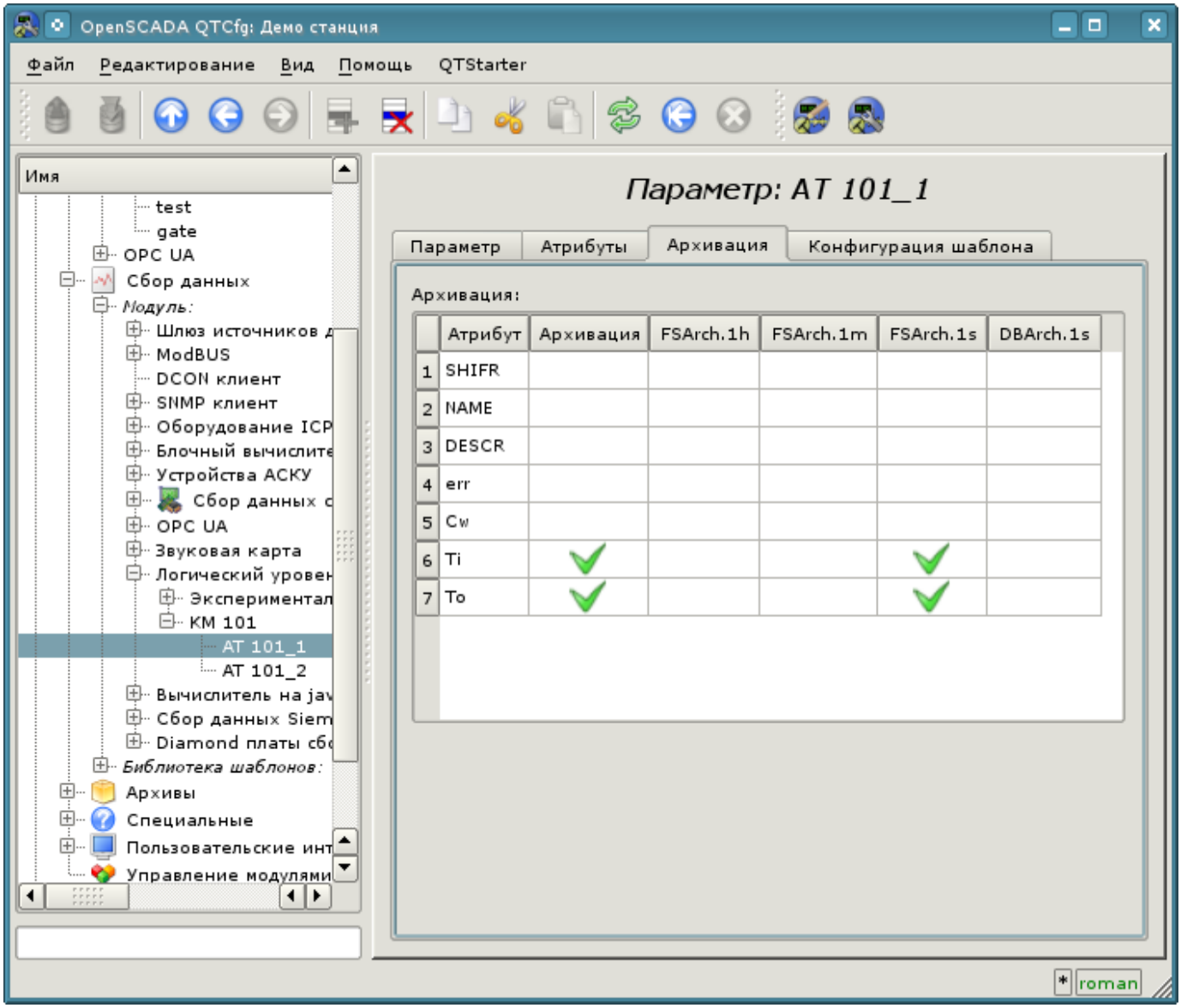


Рис. 4.3.1. Вкладка "Архивация" параметра AT101_1 модуля "LogicLev".

В результате этой операции будут автоматически созданы объекты архивов для выбранных атрибутов. Например, объект архива для атрибута "Ti" параметра AT101_1 представлен на рис.4.3.2.

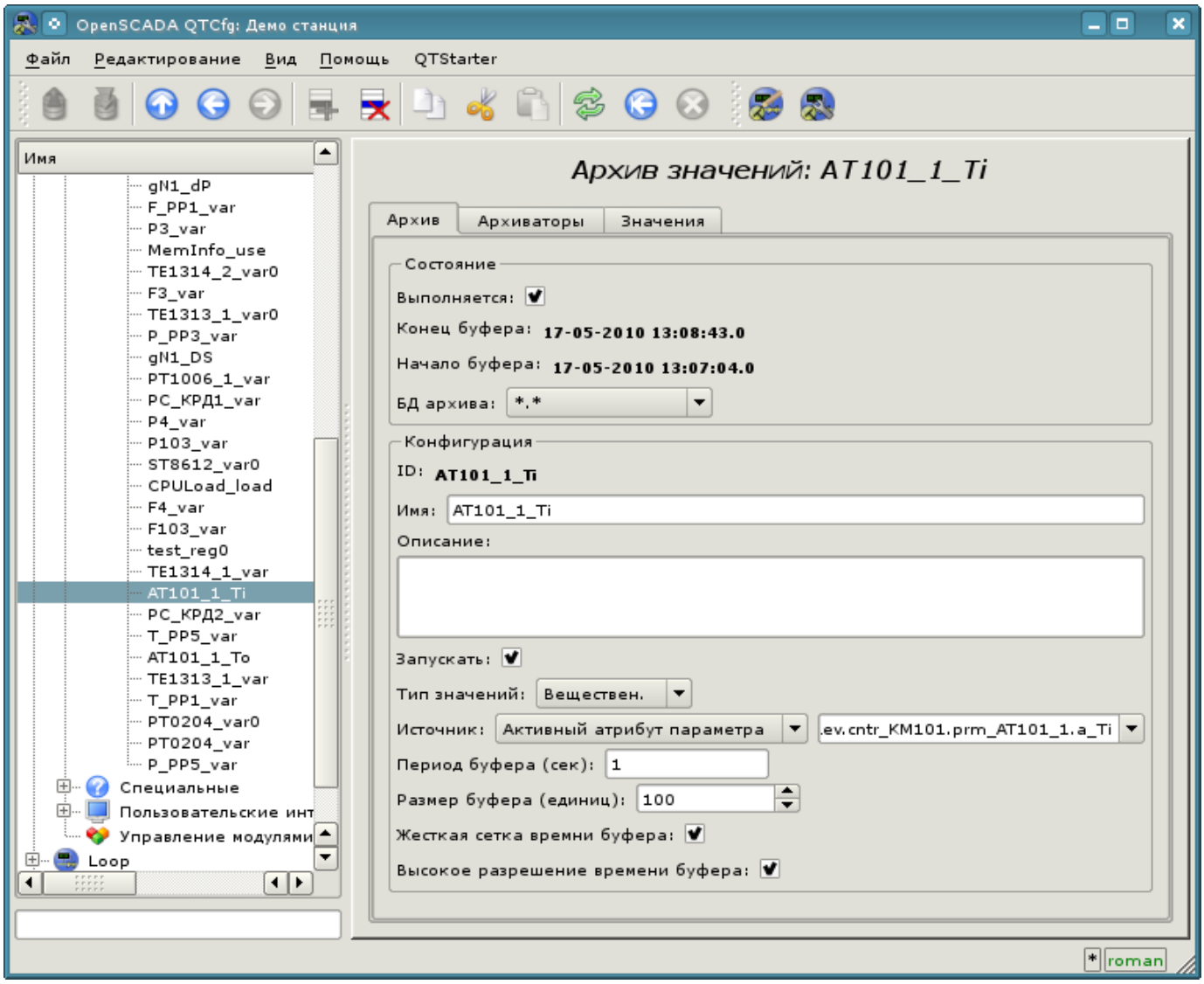


Рис. 4.3.2. Страница объекта архива атрибута "Ti" параметра AT101_1.

Обычно настройки архива менять не нужно, однако, если потребуется особая конфигурация, то её можно сделать на указанной выше странице. Чаще может понадобиться получение информации об архиве. Например, узнать размер архива как по времени, так и на носителе, а также взглянуть на график параметра (рис.4.3.3).

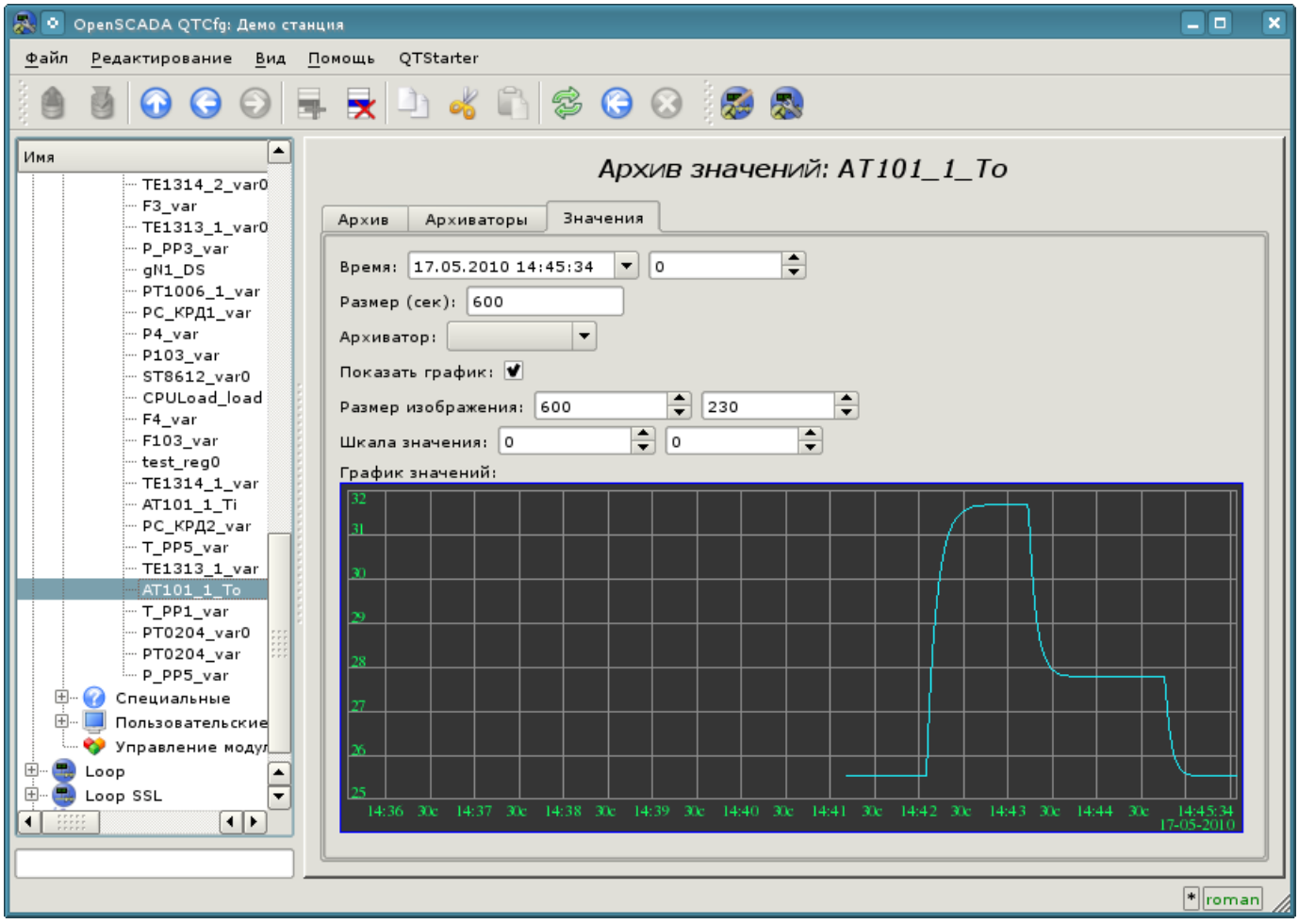


Рис. 4.3.3. Вкладка "Значения" страницы объекта архива атрибута "Ti" параметра AT101_1.

5. Формирование визуального представления

Формирование визуального представления может выполняться на трёх уровнях сложности и пользователь может выбрать любой из них, в зависимости от уровня своих знаний и наличия библиотек с готовыми образами и шаблонами.

Первый уровень требует минимальной квалификации пользователя, но подразумевает наличие библиотек шаблонных кадров, нужных для решения его задачи. В рамках первого уровня пользователю достаточно знать, как подключить динамику к страницам шаблонных кадров и как добавить новые страницы шаблонных кадров.

Второй уровень предусматривает дополнительную способность создавать новые кадры на основе готовых комплексных элементов, путём простого их размещения в области кадра. Для обеспечения этого квалификационного уровня пользователю понадобятся библиотеки комплексных элементов, нужных для решения его задач.

И третий уровень предусматривает владение всеми инструментами среды разработки визуальных интерфейсов OpenSCADA, включая создание новых комплексных элементов и разработки новых интерфейсов пользователя в проекте.

Все работы над интерфейсом визуализации будем выполнять в окружении модуля "Vision" подсистемы "Пользовательские интерфейсы". Для открытия окна интерфейса "Vision" нажмём вторую иконку справа на панели инструментов конфигуратора. В результате получим окно, ранее изображённое на рис.3.3.

5.1. Добавление шаблонной страницы в проект и подключение динамики

Рассмотрим задачу первого уровня сложности, когда в уже разработанном интерфейсе нужно подключить динамику к шаблонной странице. Понятие "Шаблон страницы" подразумевает страницу, на основе которой путём наследования может создаваться множество конечных страниц визуализации с индивидуальным перечнем динамики. Примером таких страниц являются: "Группа графиков", "Группа контуров", "Обзорный кадры" и "Сводная таблица". На рис.5.1.1 представлена шаблонная страница "Группа графиков" в дереве проекта "Группы сигнализаций (шаблон)".

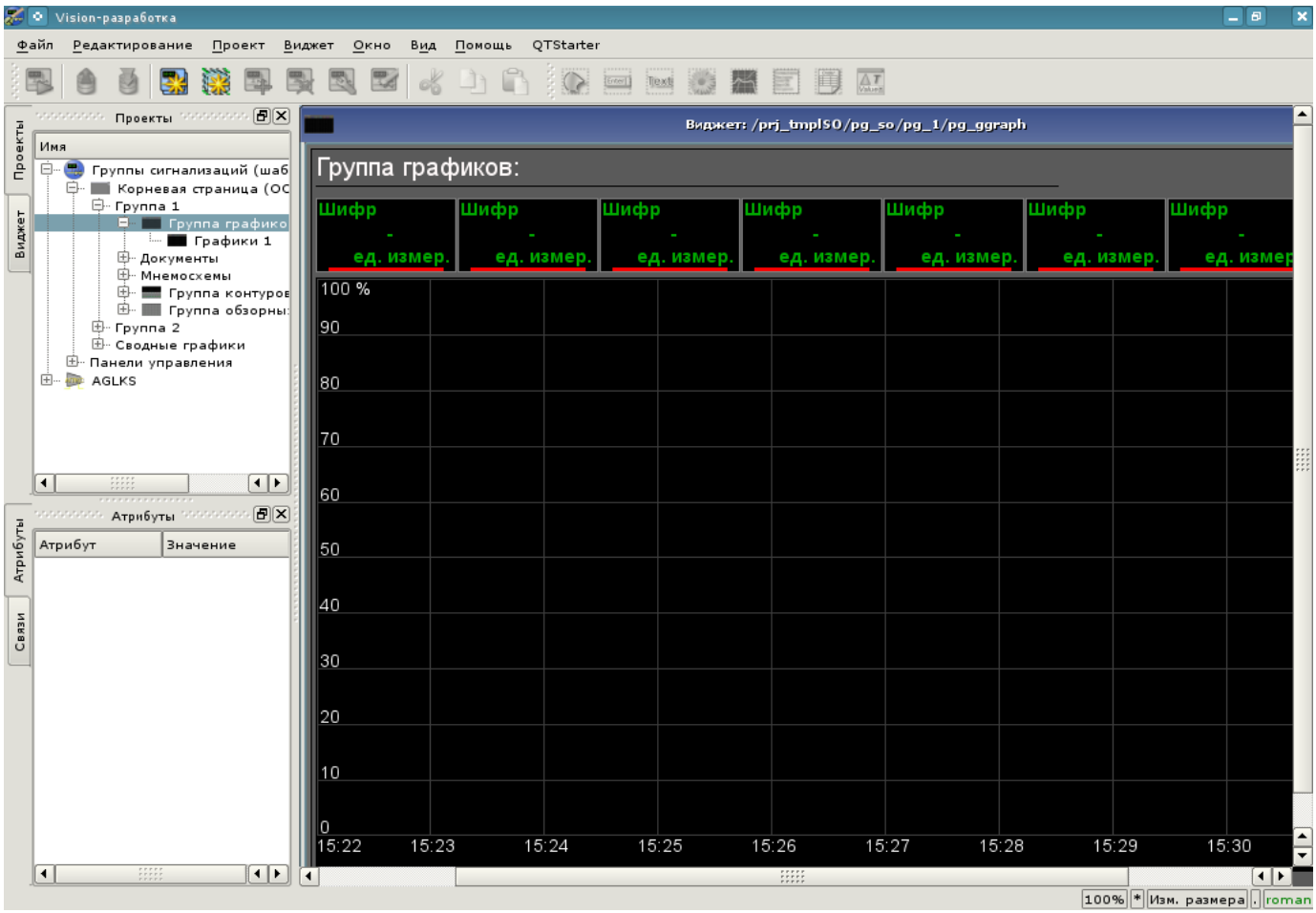


Рис. 5.1.1. Шаблонная страница "Группа графиков".

Шаблонная страница "Группа графиков" предоставляет возможность подключить до восьми сигналов для одновременного их отображения на графике. Элементы отображения значения сверху автоматически скрываются для неустановленных ссылок.

Создадим новую группу графиков "Графики 2" в шаблонном контейнере "Группа графиков" первой группы корневой страницы проекта "Группы сигнализаций (шаблон)". Для этого в контекстном меню пункта "Группа графиков" выберем "Добавить визуальный элемент" (рис.5.1.2). Для ввода идентификатора и имени нового визуального элемента появится диалог для их ввода (рис.5.1.3). Введём идентификатор "2" и имя "Графики 2".

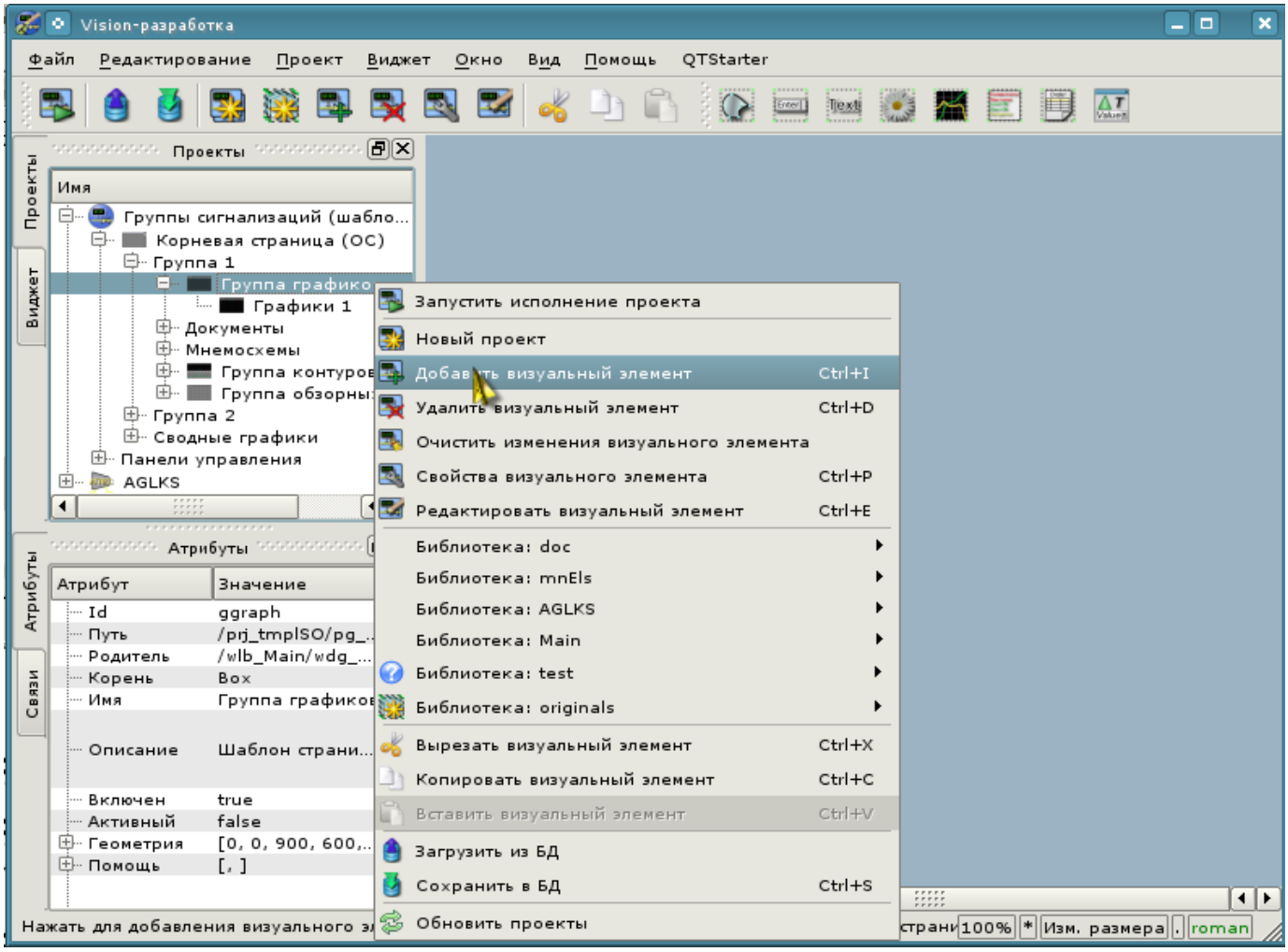


Рис. 5.1.2. Добавление группы графиков "Графики 2".

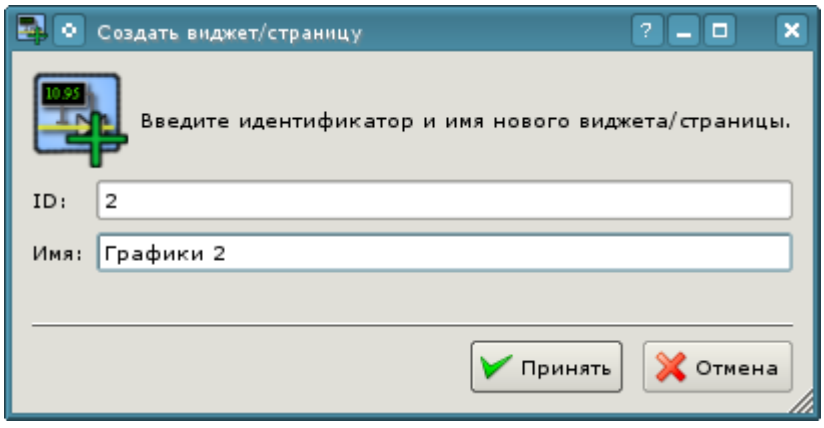


Рис. 5.1.3. Диалог ввода идентификатора и имени.

После подтверждения ввода имени будет создана новая страница. Однако, для её активации нам понадобится её включить. Включить страницу можно в диалоге редактирования свойств страницы (рис.5.1.4). Открыть эту страницу можно посредством выбора пункта меню "Свойства визуального элемента" в контекстном меню вновь созданной страницы.

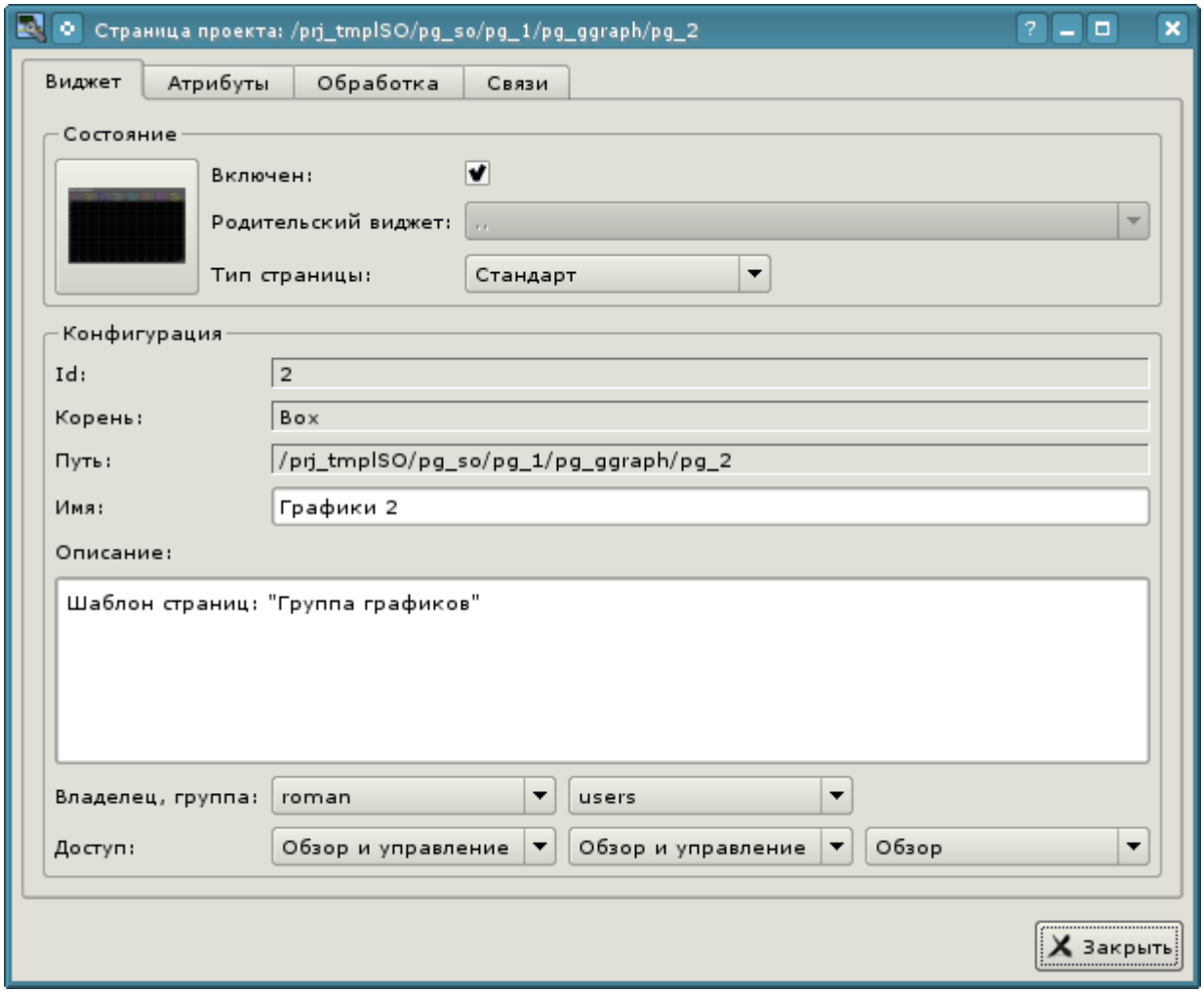


Рис. 5.1.4. Диалог редактирования свойств визуального элемента.

После включения страницы можно приступать к установке связей на созданные в предыдущей главе параметры контроллеров. Для этого, не покидая диалога редактирования свойств вновь созданной страницы (рис.5.1.4), перейдём на вкладку "Связи" (рис.5.1.5). На этой вкладке мы увидим дерево с элементами "el1" ... "el8". Развернув любой из элементов мы увидим ветку "Parameter", вот в ней мы и должны указать или выбрать адрес наших атрибутов "Ti" и "To". Итого заполним четыре элемента. При заполнении элементов часть свойств нужно указывать как постоянные. Например, обязательно нужно указать:

- *name* - "val:AT101_1 Ti"
- *ed* - "val:град.C"
- *max* - "val:150" (для Ti) и "val:100" (для To)
- *min* - "val:0"

Если заранее предусмотреть наличие атрибутов, указанных постоянными в шаблоне параметра контроллера, то можно будет указывать только параметр, а атрибуты расставятся автоматически.

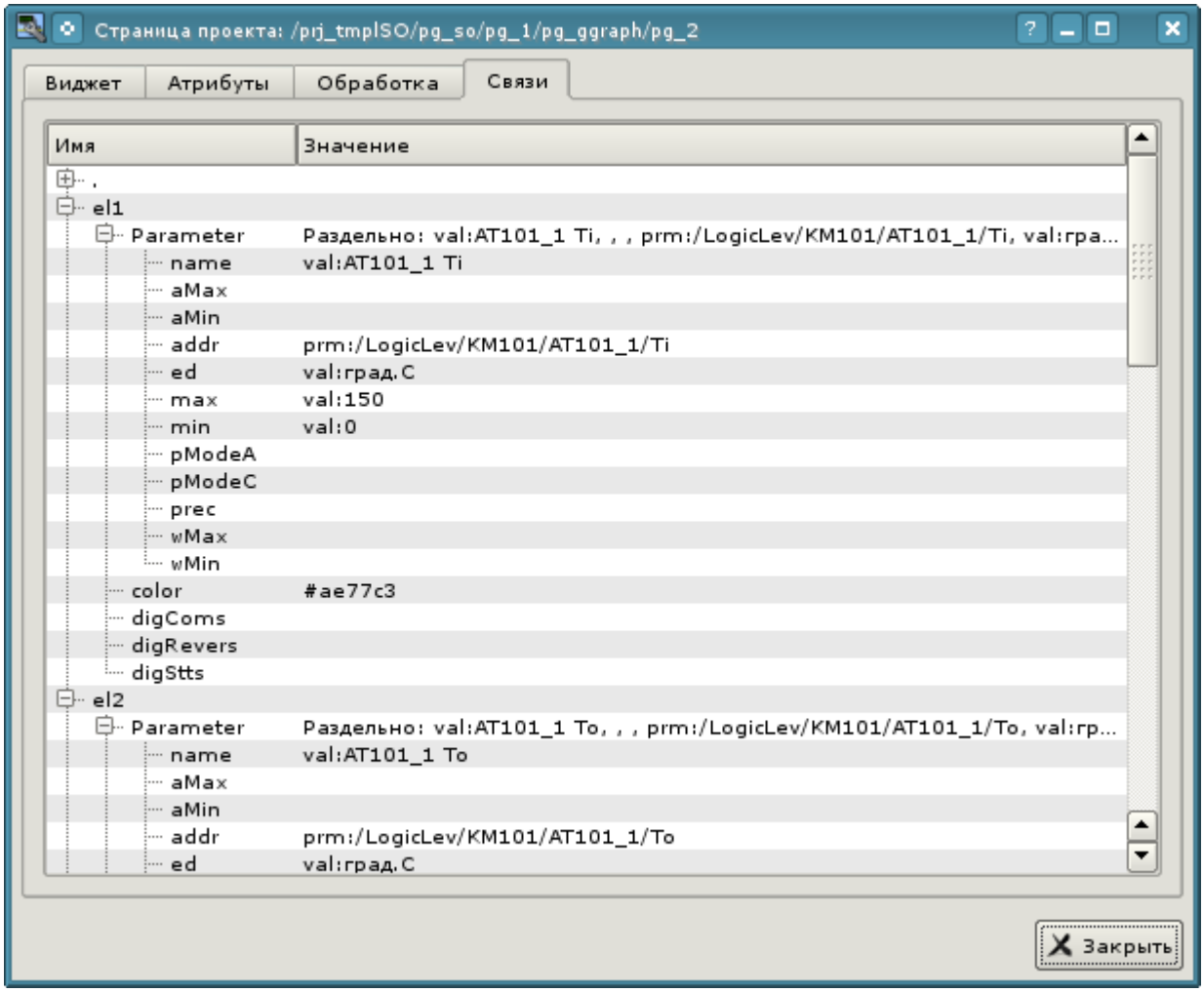


Рис. 5.1.5. Вкладка "Связи" диалога редактирования свойств визуального элемента.

Закончив ввод связей, можем проверить, что получилось в результате наших усилий. Для этого закроем оно диалога свойств и запустим проект "Группы сигнализаций (шаблон)" на исполнение, про кнопку запуска мы помним из первых глав. Затем выберем графики и переключимся на вторую страницу. При безошибочной конфигурации мы должны увидеть что-то подобное изображённому на рис.5.1.6.

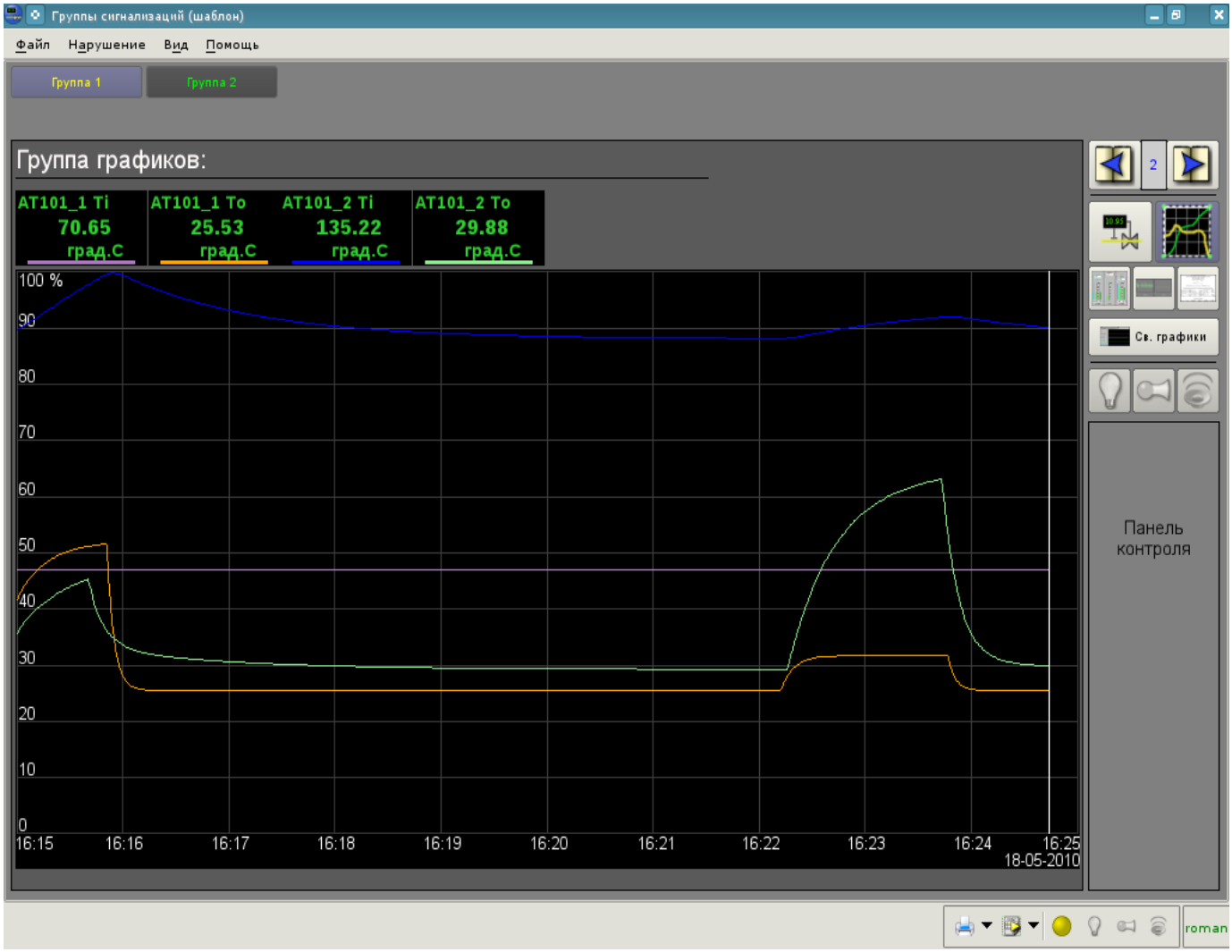


Рис. 5.1.6. Созданная группа графиков с четырьмя подключенными сигналами.

5.2. Создание нового кадра, мнемосхемы

Поднимем планку и создадим новый кадр, куда поместим базовые элементы отображения значений параметров наших контроллеров. Такие кадры обычно называются мнемосхемами и кроме отображения динамики, и даже в первую очередь, содержат статическое изображение технологического процесса в мнемоническом представлении. Мы же не будем акцентировать внимание на создание статики, а добавим элементы динамики и подключим к ней параметры наших контроллеров. Ну и поместим созданный кадр в дерево уже известного нам проекта.

Новые кадры, предназначенные в последствии для помещения в проект, принято создавать в библиотеке виджетов. Создадим новую библиотеку виджетов "KM101"; выбрав вертикальную вкладку "Виджет" и в контекстном меню окна библиотек виджетов выберем пункт "Новая библиотека" (рис.5.2.1). В диалоге ввода имени укажем идентификатор "KM101" и имя "KM 101", а затем подтвердим.

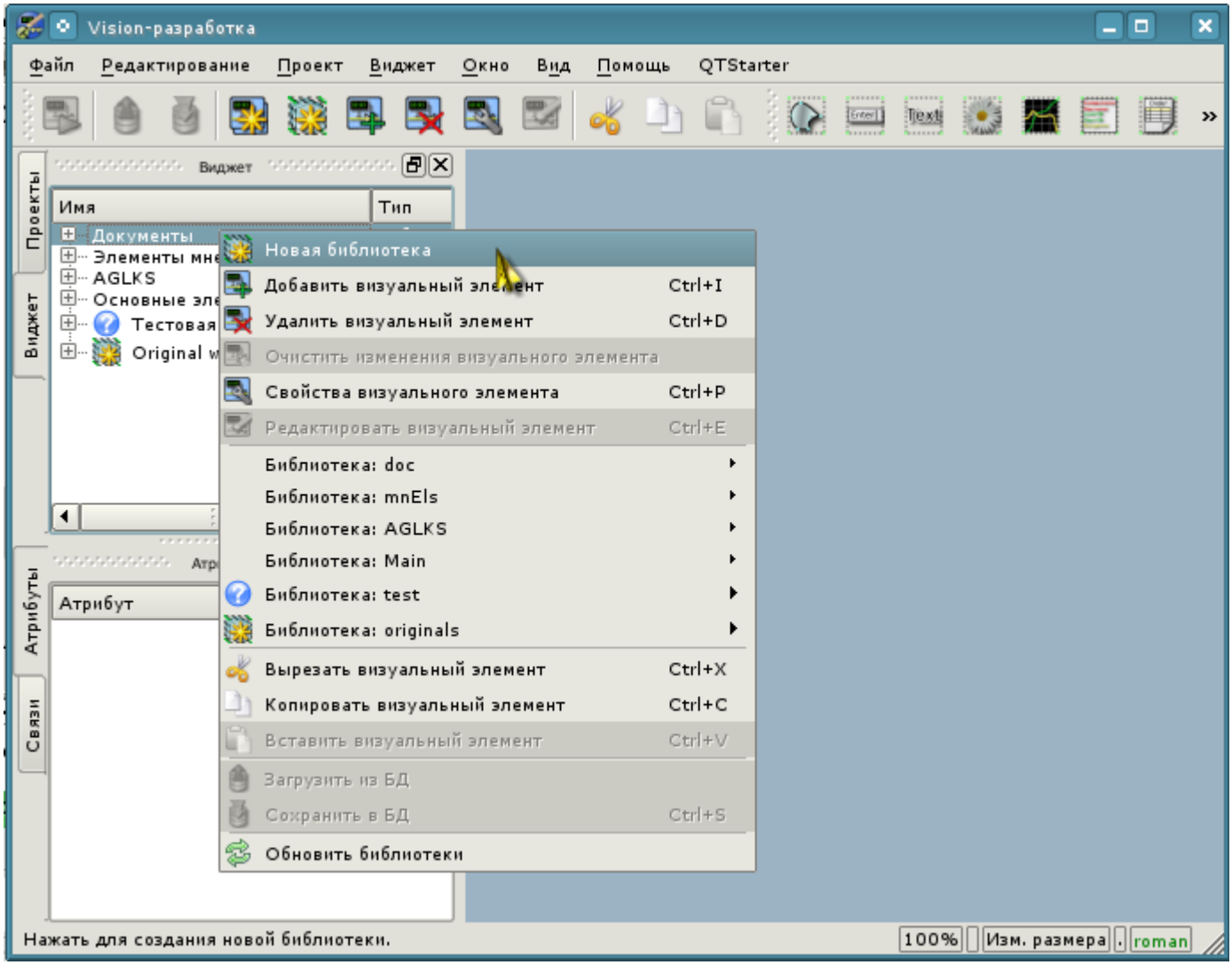


Рис. 5.2.1. Добавление новой библиотеки виджетов.

Далее добавляем новый кадр "AT101", выбрав пункт "Библиотека: originals"->"Группа элементов" в контекстном меню созданной библиотеки "KM101" (рис.5.2.2). В диалоге ввода имени укажем идентификатор "AT101" и имя "AT 101", а затем подтвердим. В основе любого кадра и страницы должен лежать элемент "Группа элементов" ("Box"), поэтому мы его и выбрали.

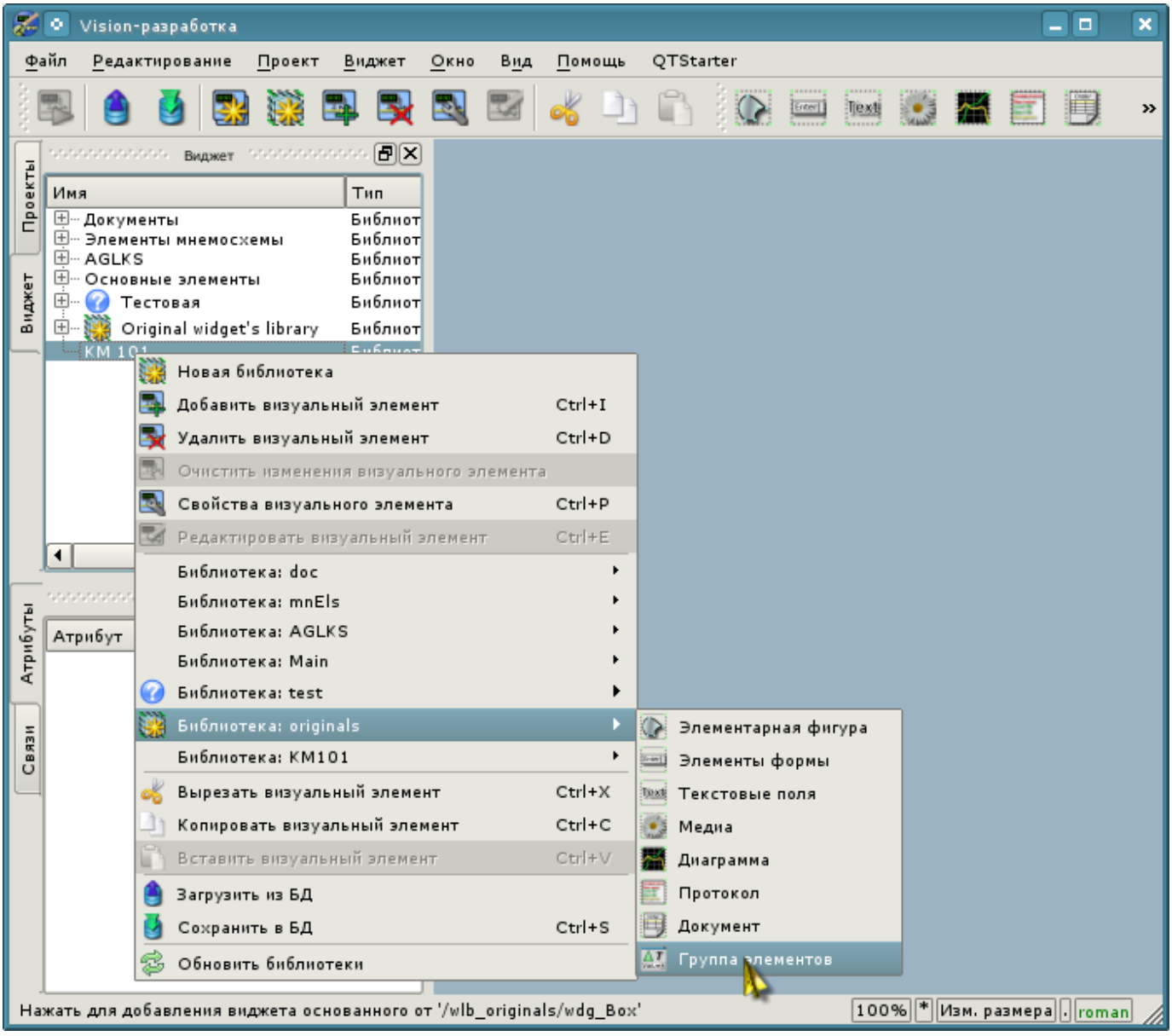


Рис. 5.2.2. Добавление нового кадра.

Сразу после создания элемента нового кадра нужно установить его базовые свойства, характерные для кадра мнемосхемы. Свойства или атрибуты любого визуального элемента можно указать в панели инструментов "Атрибуты", предварительно выбрав нужный визуальный элемент. Выберем созданный кадр "AT 101" и установим следующие свойства:

- *Геометрия:ширина* - 900;
- *Геометрия:высота* - 600;
- *Фон:цвет* - "#5A5A5A";
- *Граница:ширина* - 1;
- *Граница:цвет* - "black".

В результате получим пустой кадр (рис.5.2.3), готовый для добавления элементов на него. Для редактирования или просмотра вида кадра необходимо в контекстном меню кадра выбрать пункт "Редактировать визуальный элемент".

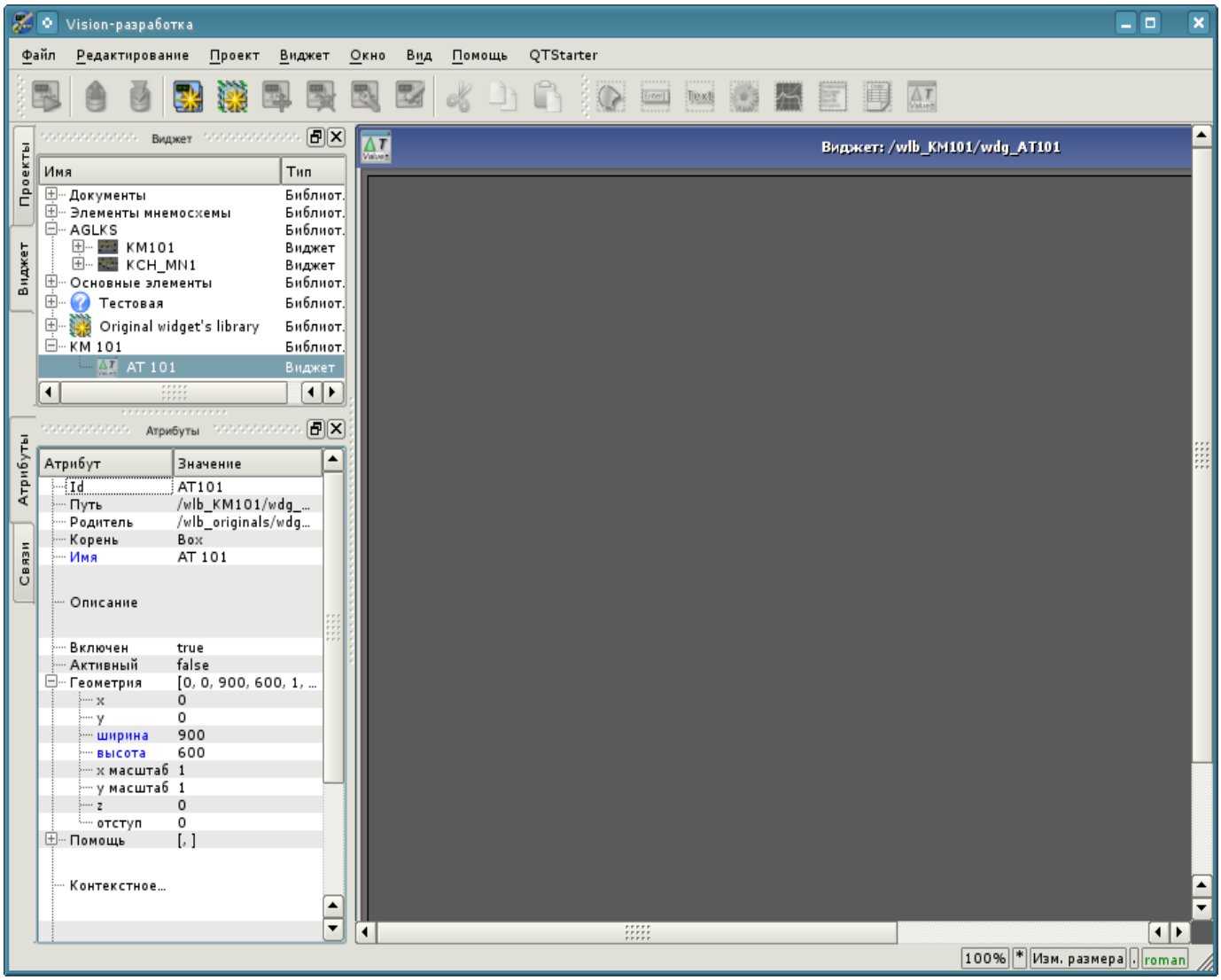


Рис. 5.2.3. Вид нового кадра и установленных атрибутов для мнемосхемы.

Теперь добавим на кадр элементы отображения значения аналогового параметра для наших четырёх сигналов. Для помещения на мнемосхему элемента отображения аналогового сигнала нужно выбрать нашу мнемосхему, а затем в меню окна выбрать пункт меню "Виджет"->"Библиотека: Main"->"Отобр аналог"; после чего появится курсор с образом этого элемента, который нужно подвести в желаемую область мнемосхемы и нажать левую кнопку мыши. В момент добавления появится диалог с запросом имени нового элемента. Добавлять подобным образом будем четыре элемента, которые назовём: "A1_Ti", "A1_To", "A2_Ti" и "A2_To". Добавленные элементы можно в последствии расположить как нужно, просто выделяя и перетаскивая мышью. После выполнения подобных манипуляций у нас должна получиться мнемосхема с видом, похожим представленной на рис.5.2.4.

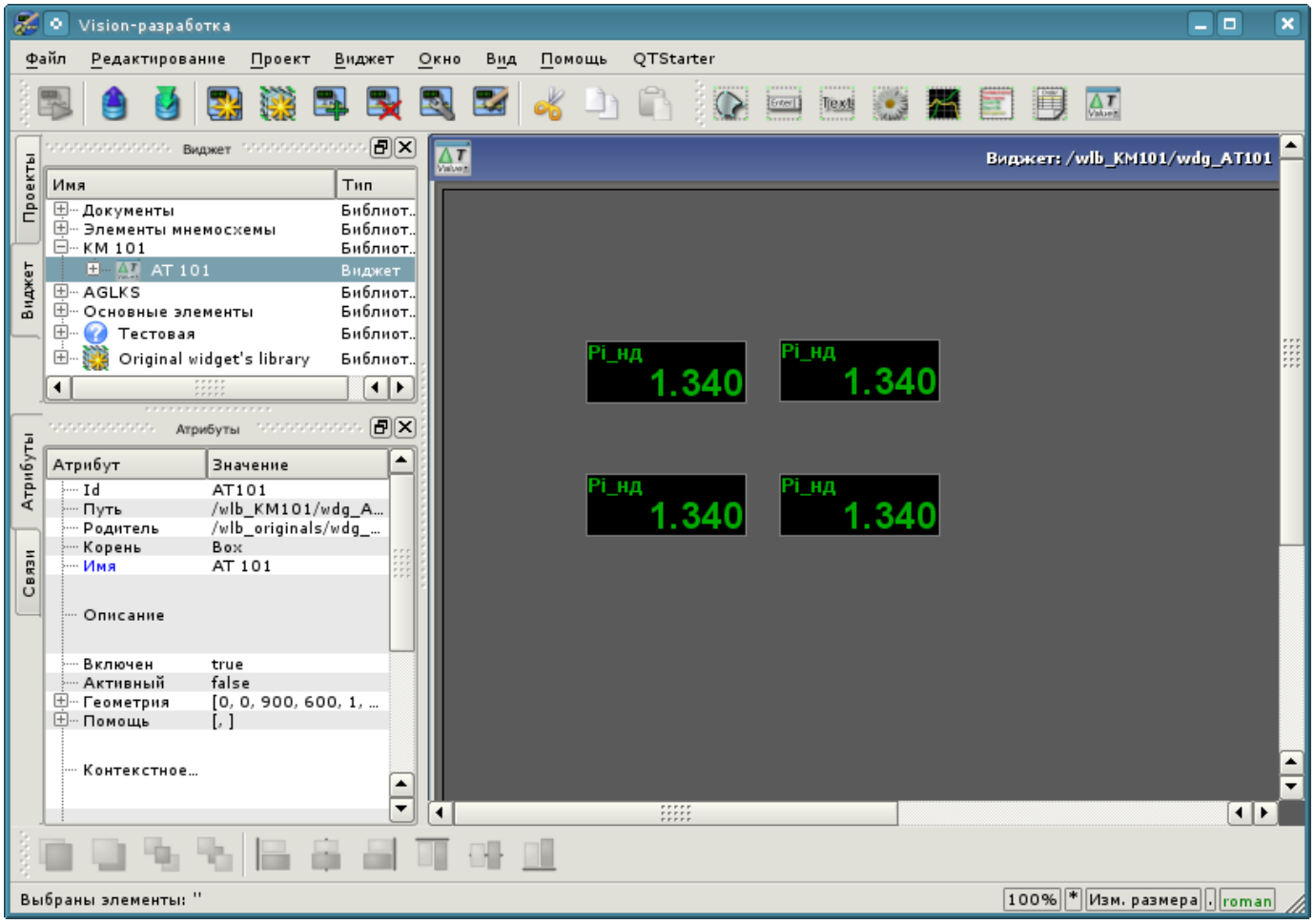


Рис. 5.2.4. Вид нового кадра и установленных атрибутов для мнемосхемы.

На этом процедуру создания мнемосхемы будем считать законченной. Сохраним новую библиотеку виджетов "KM101" и приступим к этапу размещения нашей мнемосхемы в дереве проекта "Группы сигнализаций (шаблон)".

Поместим нашу мнемосхему в ветвь "Группы сигнализаций (шаблон)"->"Корневая страница"->"Группа 1"->"Мнемосхемы" путём выбора в контекстном меню для пункта "Мнемосхемы" пункта "Библиотека: KM101"->"AT 101". Идентификатор для новой мнемосхемы установим в "2", при этом поле имени оставим пустым. Сразу после добавления нужно установить базовое свойство мнемосхемы "Страница: группа" в значение "so".

Далее нужно произвести уже знакомую нам операцию по предыдущей главе, а именно установку связей на созданные в предыдущей главе параметры контроллеров. Для этого откроем диалог редактирования свойств мнемосхемы на вкладке "Связи" (рис.5.2.5). На этой вкладке мы увидим дерево с элементами "A1_Ti", "A1_To", "A2_Ti" и "A2_To". Развернув любой из элементов, мы увидим ветку "Parameter", вот в ней мы и должны указать или выбрать адрес значений наших атрибутов "Ti" и "To" соответственно. При заполнении элементов часть свойств нужно указывать как постоянные. Например, обязательно нужно указать:

- *pName* - "val:AT101_1 Ti"

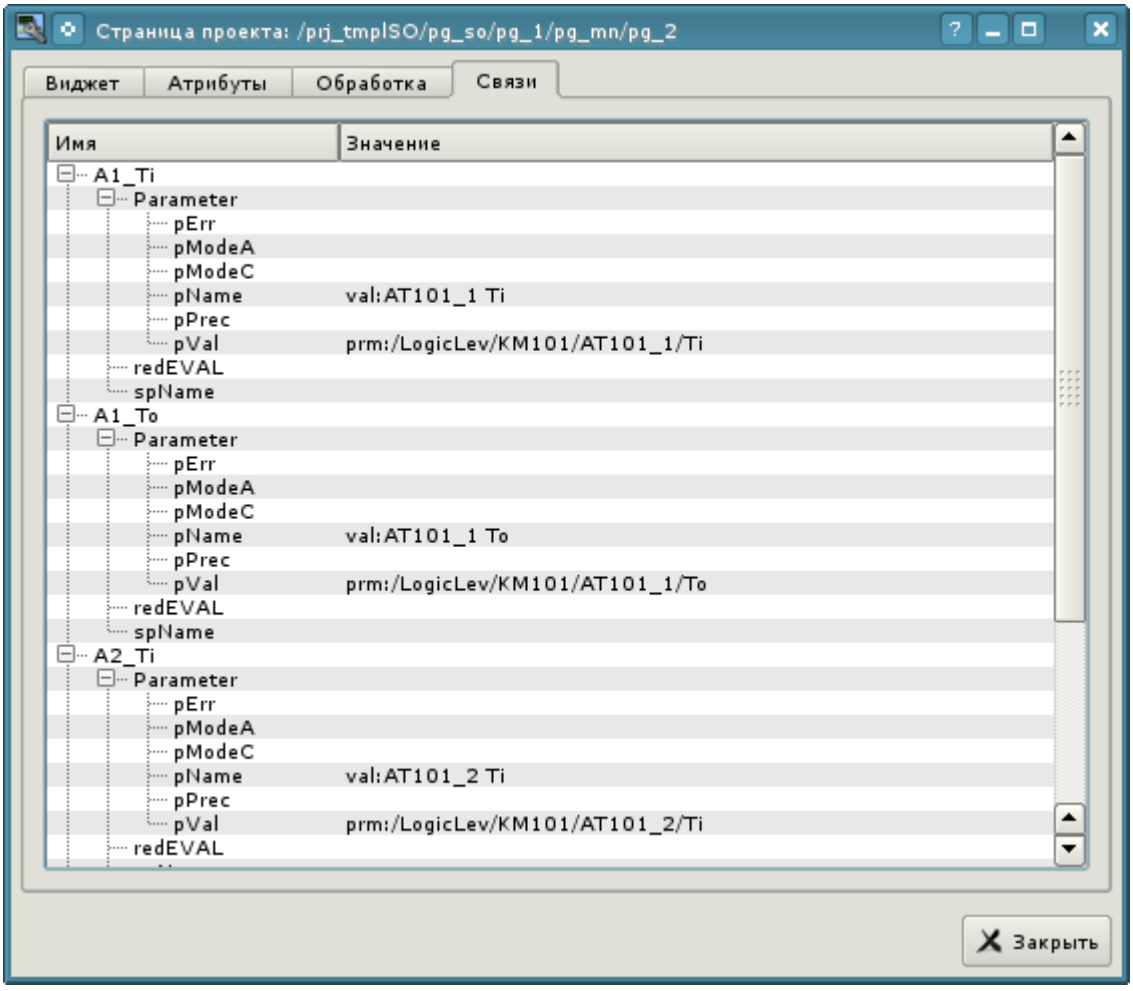


Рис. 5.2.5. Вкладка "Связи" диалога редактирования свойств мнемосхемы.

Теперь можем сохранить нашу мнемосхему и проверить, что получилось. Для этого закроем окно диалога свойств и запустим проект "Группы сигнализаций (шаблон)" на исполнение. Затем переключимся на вторую мнемосхему кнопками листания. При безошибочной конфигурации мы должны увидеть что-то подобное изображённому на рис.5.2.6.

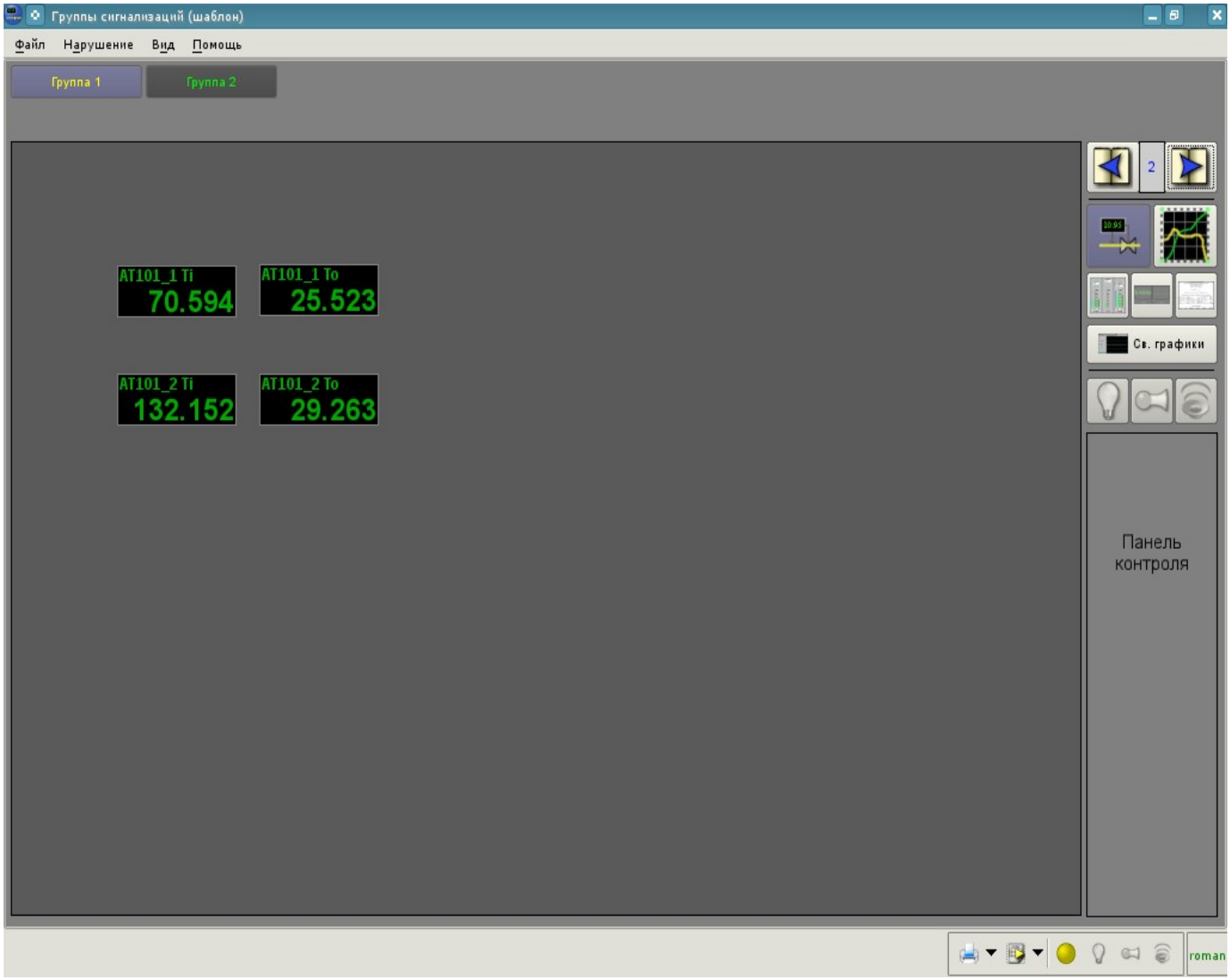


Рис. 5.2.6. Созданная мнемосхема с четырьмя подключенными сигналами.

5.3. Создание нового комплексного элемента

Приступим к рассмотрению задач третьего уровня сложности, а именно к созданию комплексного элемента. Создание нового комплексного элемента, включающего в себя комбинацию различных базовых примитивов, может осуществляться в несколько этапов. В качестве примера рассмотрим задачу, состоящую из двух этапов:

- Создание виджета "Воздушный холодильник" на основе примитива "Элементарная фигура".
- Создание финального скомпонованного виджета "Холодильник" на основе примитива "Группа элементов".

5.3.1. Создание виджета "Воздушный холодильник" на основе примитива "Элементарная фигура".

Виджет будем создавать в ранее нами созданной библиотеке "KM101". Для этого кликаем правой кнопкой манипулятора "мышь" по пункту этой библиотеке и выбираем пункт "Библиотека: originals"->"Элементарная фигура", как это показано на рисунке 5.3.1.1. Для нового элемента указываем идентификатор "air_cooler" и имя "Воздушный холодильник".

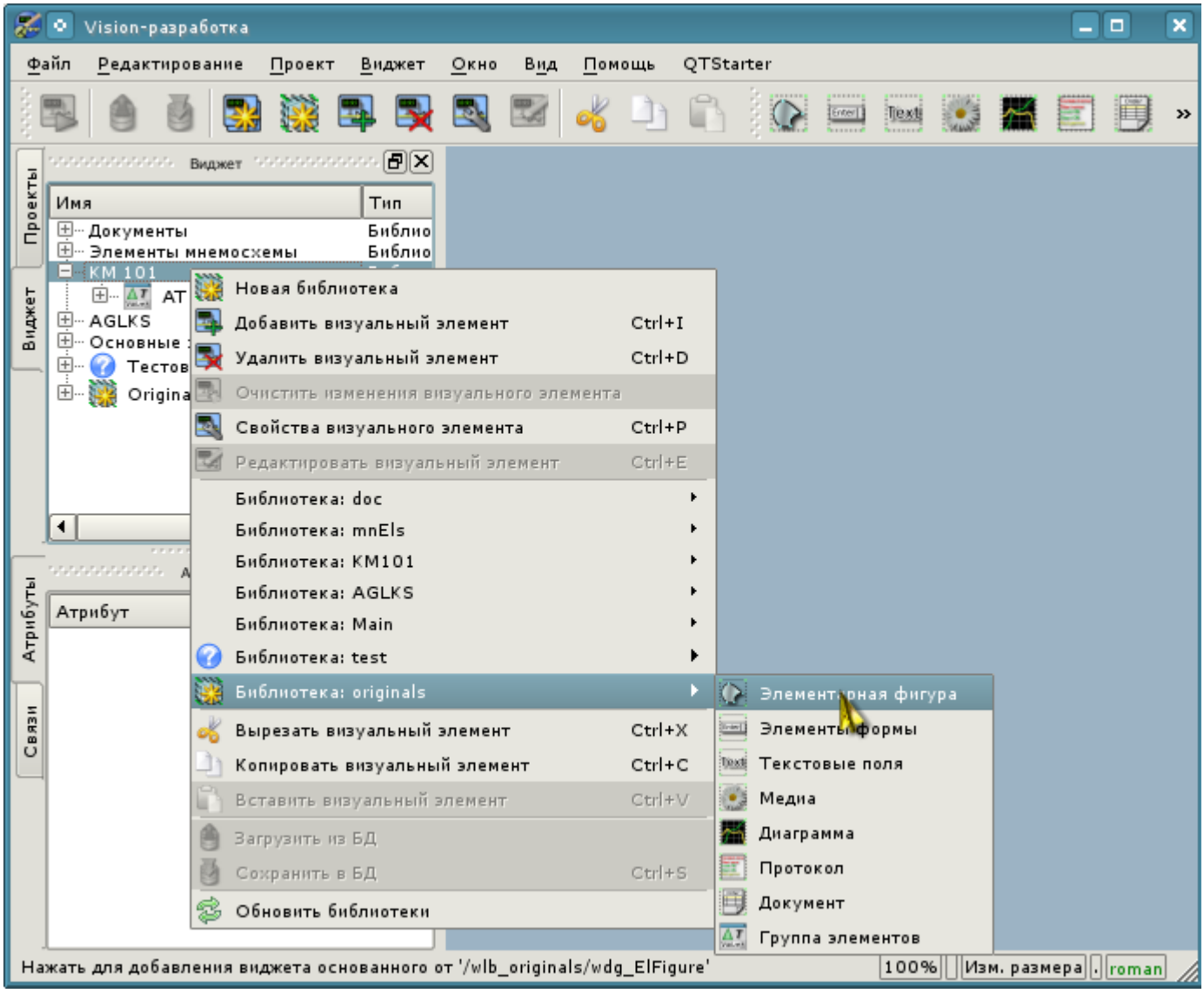


Рис. 5.3.1.1. Добавление виджета на основе примитива "Элементарная фигура" в библиотеку "KM101".

После подтверждения у нас появится объект нового виджета с именем "Воздушный холодильник". Выберем его в списке виджетов библиотеки "KM101" и откроем для редактирования посредством контекстного меню нового элемента. Зададим теперь во вкладке "Атрибуты" в пункте "Геометрия" ширину и высоту виджета в 200 пикселов (рис. 5.3.1.2).

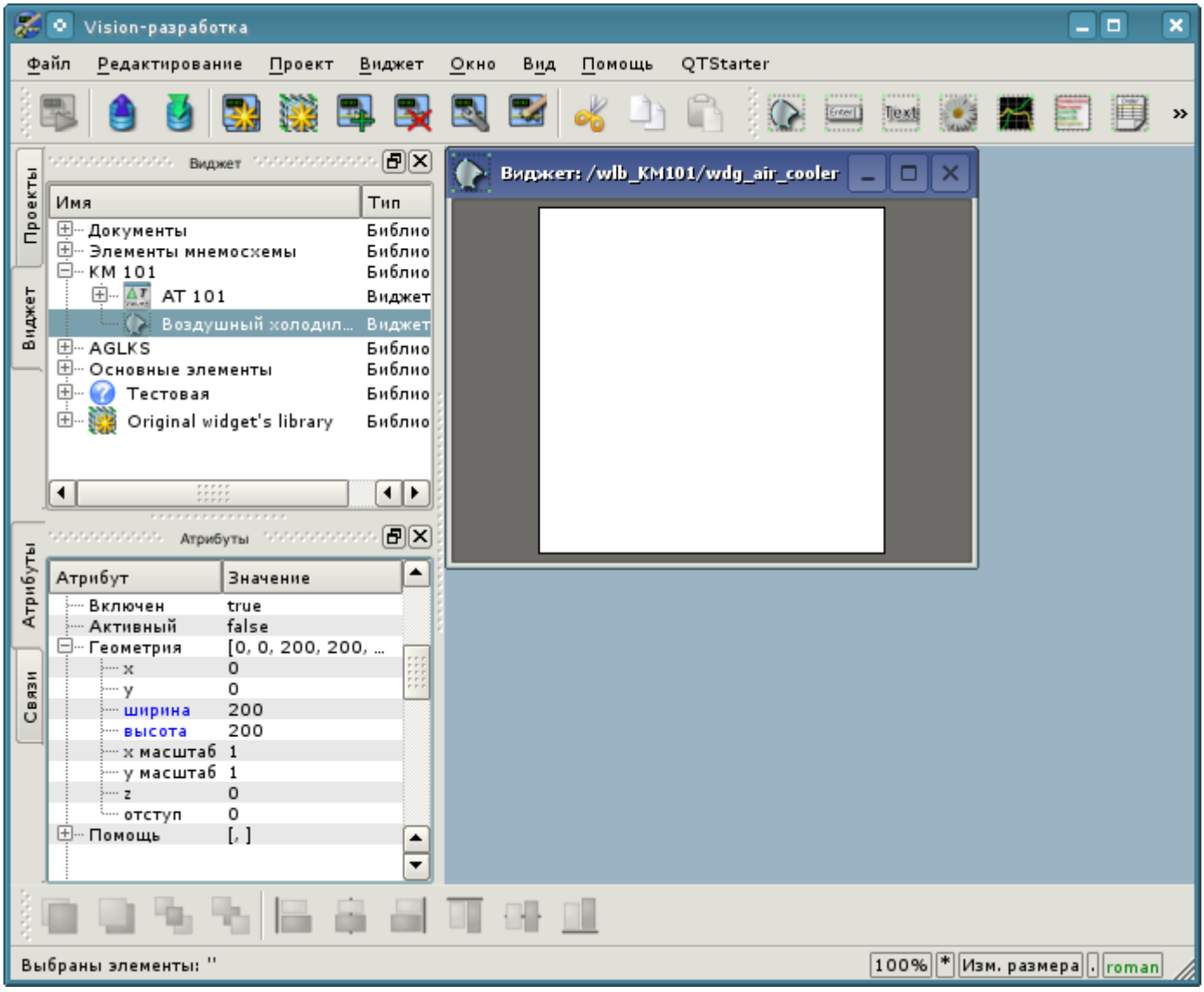


Рис. 5.3.1.2. Задание геометрических размеров виджета.

Теперь нарисуем визуальное представление виджета. Эту процедуру можно проделать двумя нижеописанными способами:

- Нарисовать желаемое изображение манипулятором "мышь", используя "Линию", "Дугу", "Кривую Безье" и "Заливку". Соответствующая панель ("Панель элементарных фигур") появится после входа в режим редактирования (рисования). Вход в этот режим осуществляется, как показано на рис. 5.3.1.3, либо двойным нажатием левой кнопки манипулятора "мышь" на теле виджета.
- Вручную заполнить поле "Список элементов", введя перечень необходимых элементов и координат точек.

Дополнительную информацию о редакторе можно прочитать здесь: ftp://ftp.oscada.org/OpenSCADA/2009FOSS/FOSS_2009.pdf

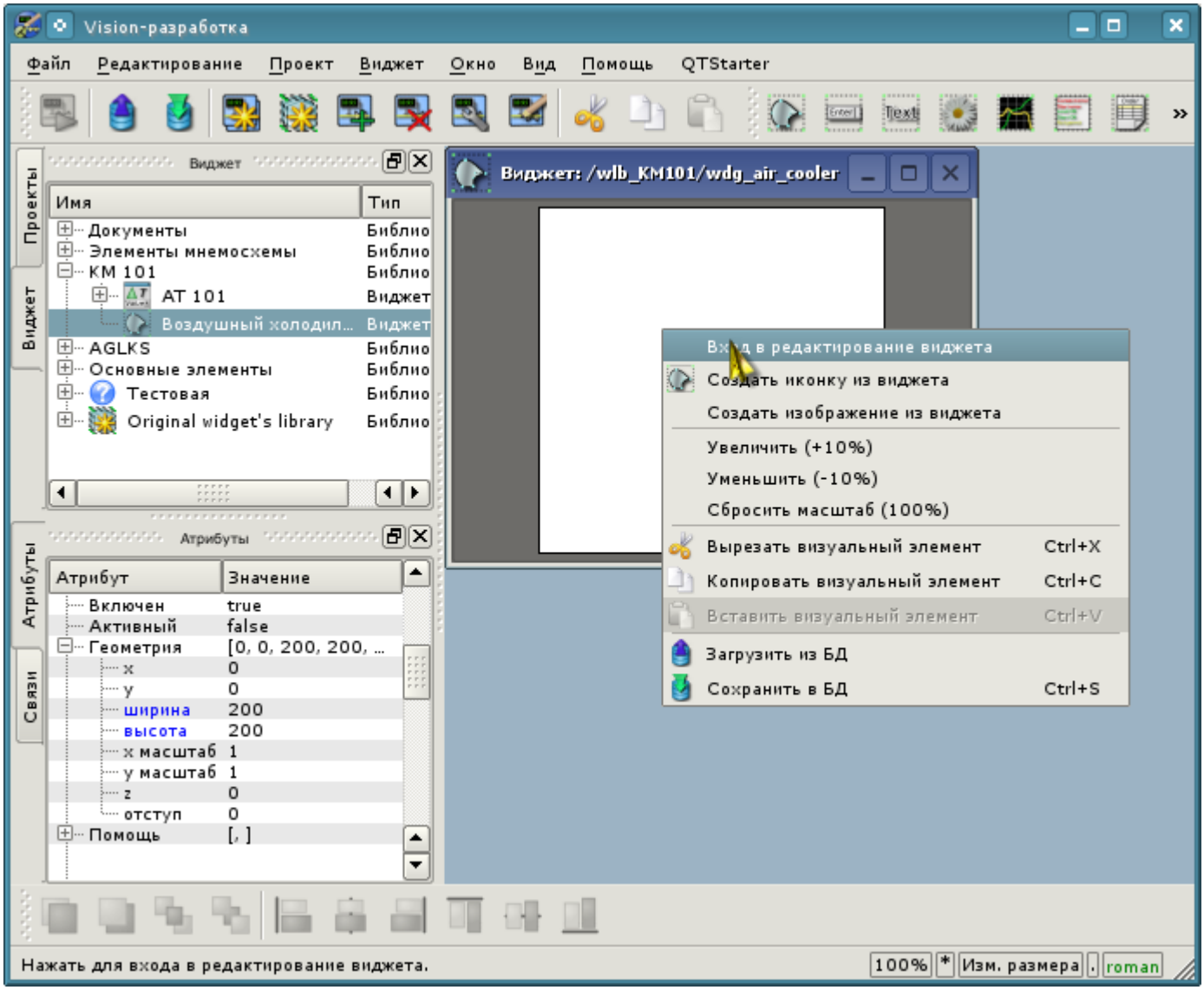


Рис. 5.3.1.3. Вход в режим рисования виджета, основанного на примитиве "Элементарная фигура".

В нашем примере мы воспользуемся вторым способом. Для этого в поле "Список элементов" инспектора атрибутов введем нижеприведенный перечень и нажмем "Ctrl"+"Enter".

```
Line:(20|80):(100|20)
line:(100|20):(180|80)
line:(180|80):(100|140)
line:(100|140):(20|80)
line:(100|20):(100|140)
line:(20|80):(180|80)
line:(50|165):(100|140)
line:(100|140):(150|165)
line:(150|165):(50|165)
fill:(20|80):(100|20):(180|80):(100|140)
fill:(50|165):(100|140):(150|165)
```

Все точки, в нашем случае, указаны в статическом виде, так как не предусматривается динамизация и смена координат в режиме исполнения, а все остальные параметры оставлены по умолчанию.

Вследствие этого наш виджет примет вид, изображенный на рис. 5.3.1.4.

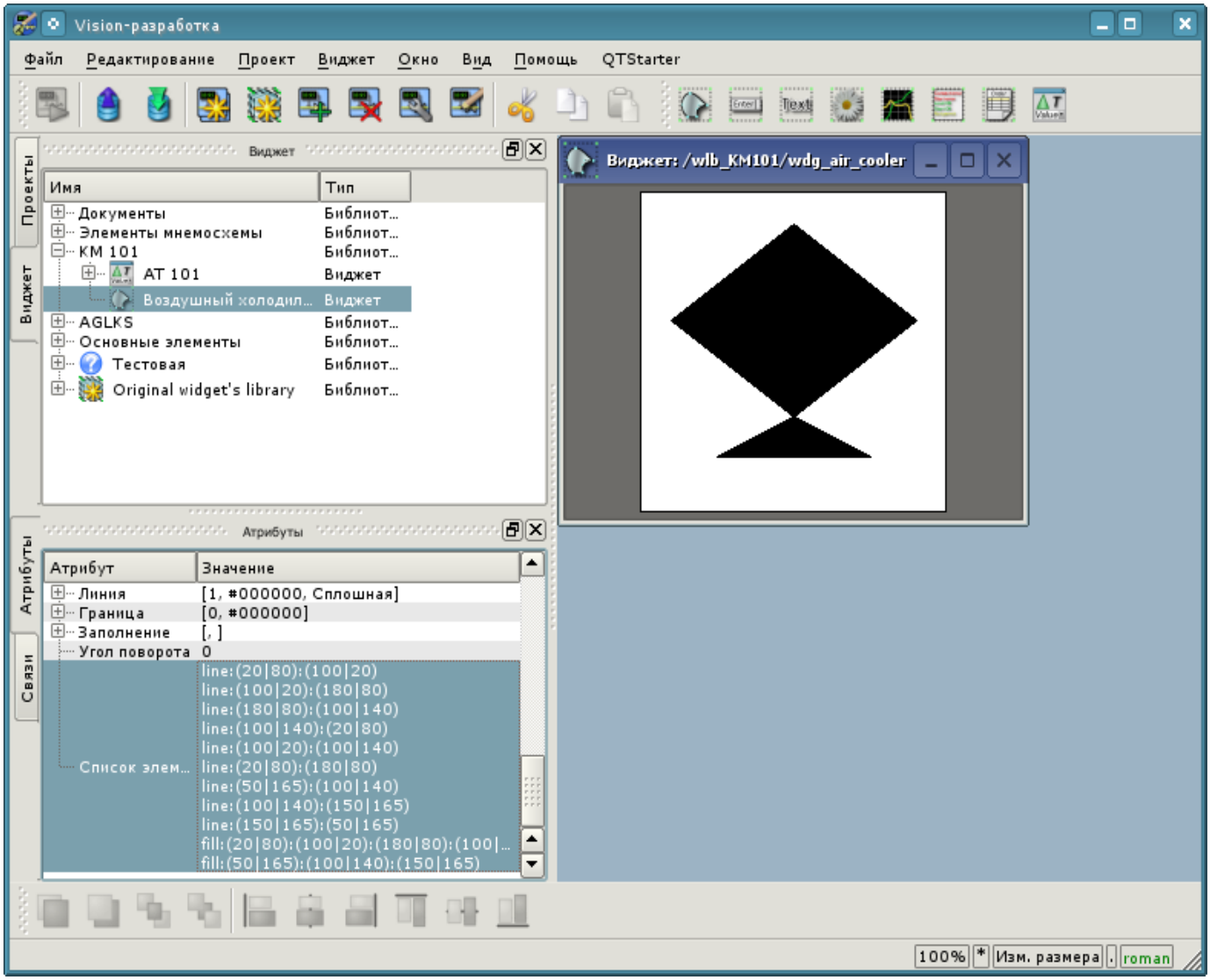


Рис. 5.3.1.4. Изображение, соответствующее "Списку элементов" виджета.

Теперь поменяем цвет заливки (черный, если не указан никакой другой (по умолчанию)) во вкладке "Атрибуты" в пункте "Заполнение" на "lightgrey" (рис. 5.3.1.5). Цвет можно задавать, как с помощью названий цветов, так и в формате #RRGGBB(#RRGGBB-AAA).

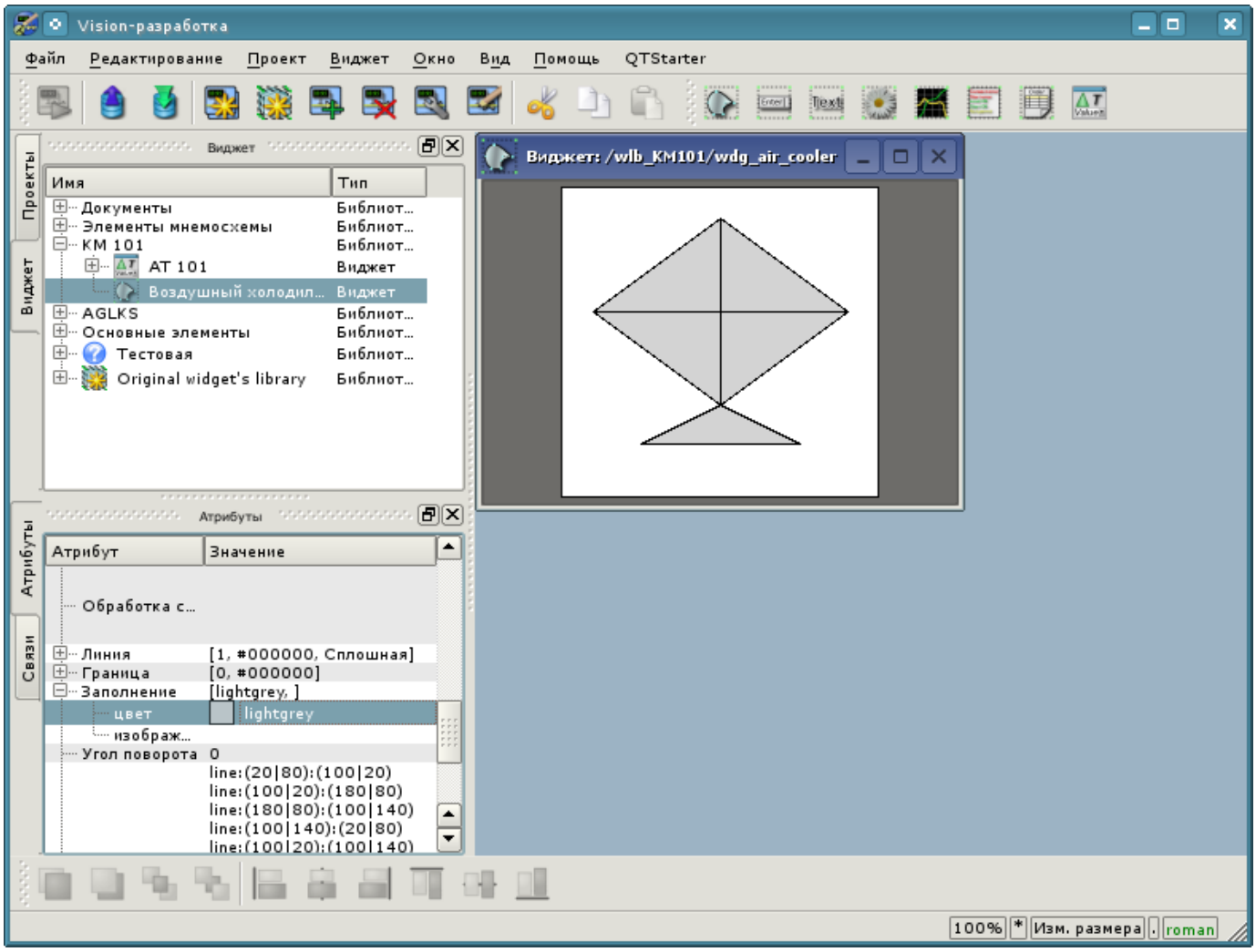


Рис. 5.3.1.5. Изменение цвета заполнения (заливки).

Создадим иконку для нашего виджета, которая будет видна в дереве виджетов библиотеки "KM101" (рис. 5.3.1.6).

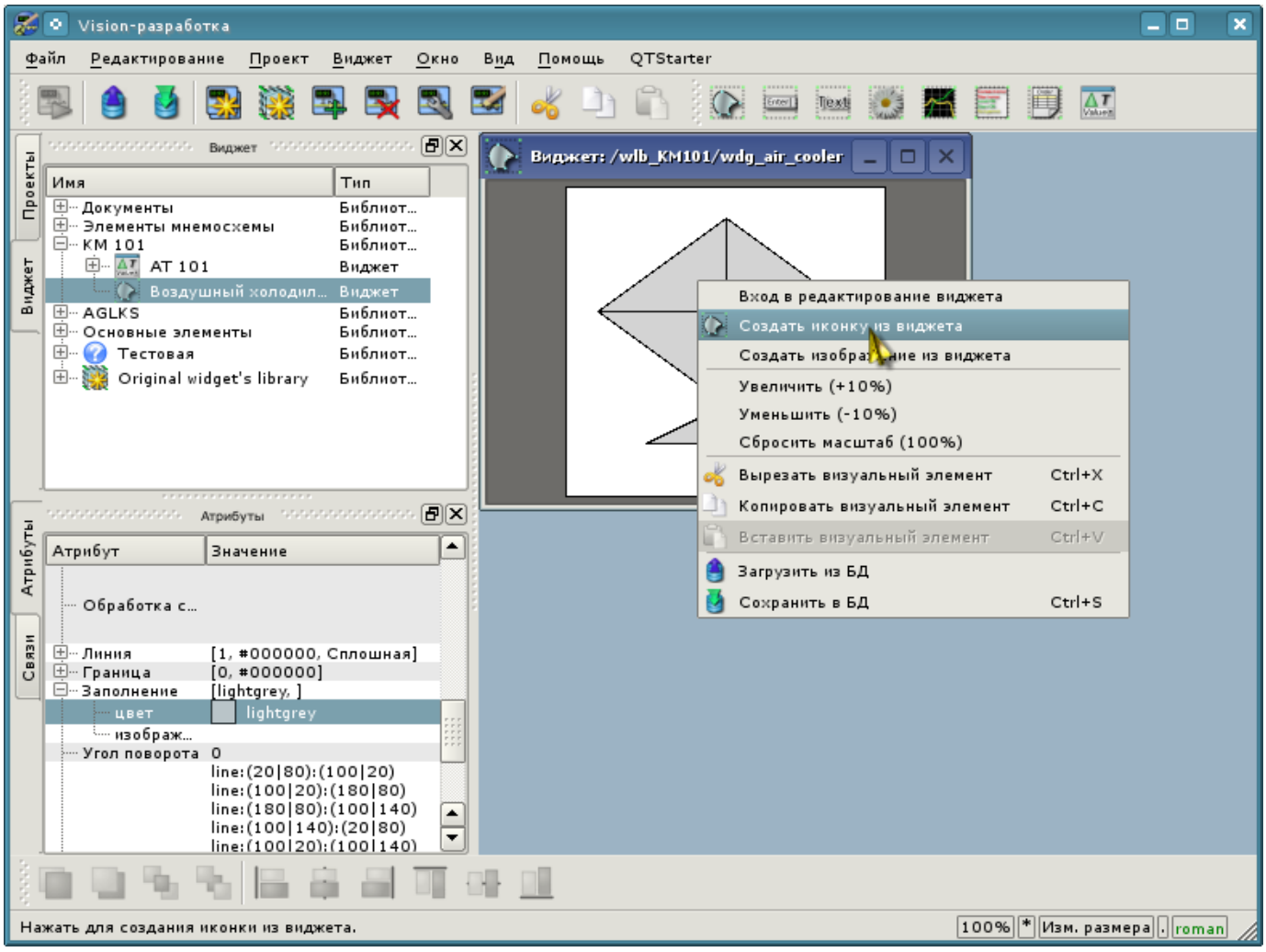


Рис. 5.3.1.6. Создание иконки для виджета.

На этом процесс создания первого виджета можно считать завершенным. Перейдем теперь к этапу компоновки и созданию результирующего виджета.

5.3.2. Создание финального скомпонованного виджета "Холодильник" на основе примитива "Группа элементов"

Результирующий виджет будем создавать в библиотеке "KM 101". Для этого кликаем правой кнопкой манипулятора "мышь" по этой библиотеке и выбираем примитив "Группа элементов", как это показано на рисунке 5.3.2.1. Для нового элемента указываем идентификатор "elCooler" и имя "Холодильник".

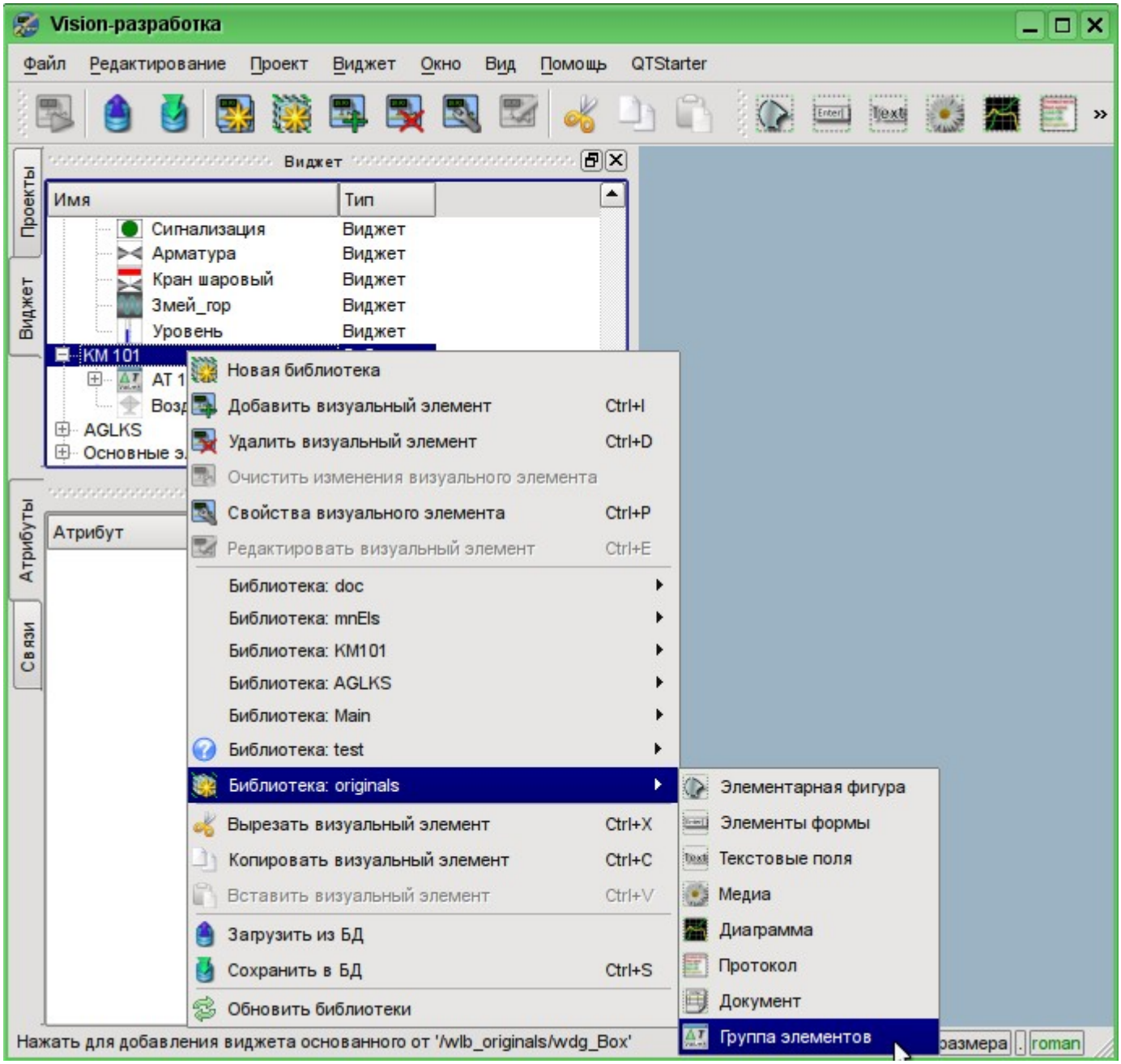


Рис. 5.3.2.1. Добавление виджета на основе примитива "Группа элементов" в библиотеку "KM 101".

После подтверждения у нас появится объект нового виджета с именем "Холодильник". Выберем его в списке виджетов библиотеки "KM 101" и открываем для редактирования. Зададим теперь во вкладке "Атрибуты" в пункте "Геометрия" ширину и высоту виджета в 250 и 200 пикселов соответственно.

Возьмём ранее созданный элемент "Воздушный холодильник" (air_cooler) и перетащим его (нажав на нем левую кнопку манипулятора "мышь", и перемещая курсор "мыши" до области виджета; после этого нужно отпустить кнопку) на вновь созданный нами виджет (рис. 5.3.2.2).

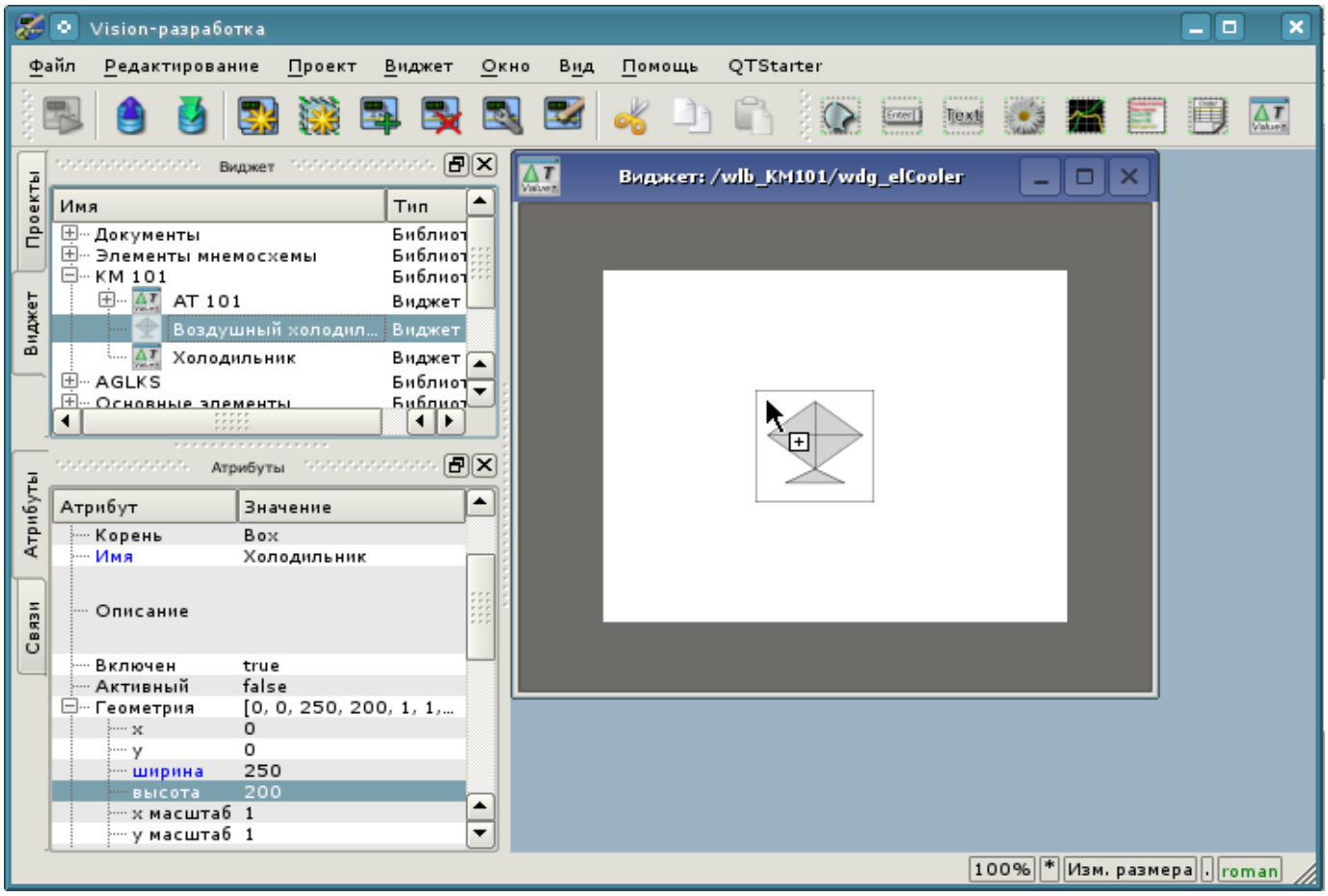


Рис. 5.3.2.2. Перетаскивание(Drag&Drop) виджета "air_cooler" в виджет-контейнер "elCooler".

В результате появится окно диалога с предложением ввести идентификатор и имя нового виджета. Идентификатор и имя могут быть заданы произвольно. Мы введём идентификатор "air_cooler", а имя оставим пустым (оно унаследуется от родителя - элемента "air_cooler"). Таким образом, вновь созданный виджет внутри контейнера "elCooler" унаследует элемент - "Воздушный холодильник" ("air_cooler"). После подтверждения ввода идентификатора и имени виджет "Воздушный холодильник" ("air_cooler") добавится в наш виджет-контейнер "elCooler" (рис. 5.3.2.3)

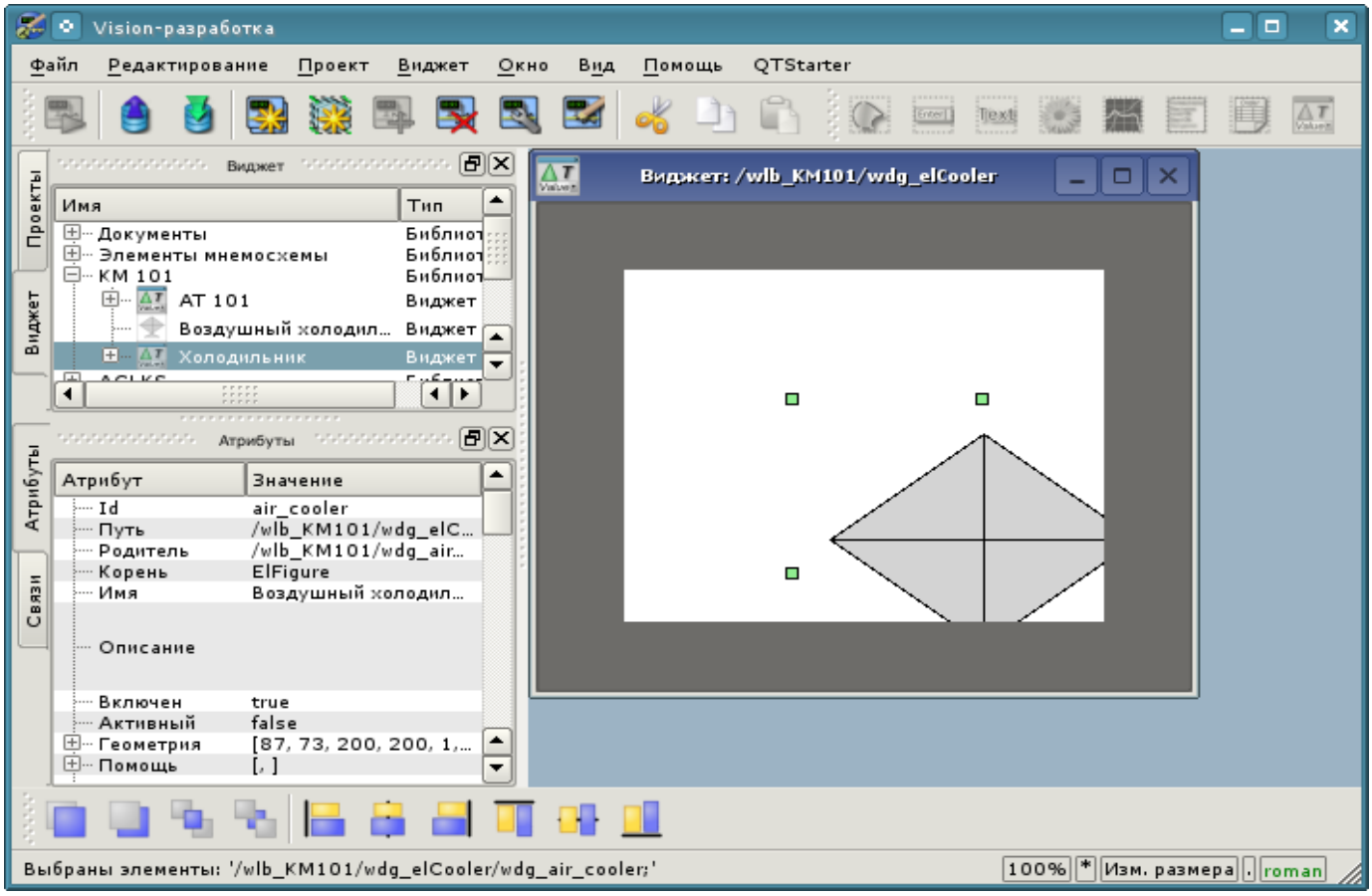


Рис. 5.3.2.3. Добавление унаследованного виджета "air_cooler".

Зададим на пенели атрибутов виджета в пункте "Геометрия" координаты "x" и "y" верхнего левого угла виджета в 25 и 0 пикселов соответственно.

Далее развернем библиотеку "Элементы мнемосхемы", найдем там элемент "Вентилятор"(cooler2) и перетащим его на виджет-контейнер. Этот элемент будет динамически отображать интенсивность работы воздушного холодильника. В результате появится окно диалога для ввода идентификатора и имени нового виджета. Введём идентификатор "cooler2", а имя снова оставим пустым. Таким образом, вновь созданный виджет внутри контейнера "elCooler" унаследует элемент библиотеки "Элементы мнемосхемы" - "Вентилятор" ("cooler2"). После подтверждения ввода идентификатора и имени виджет "Вентилятор" ("cooler2") добавится в наш виджет-контейнер "elCooler". Если необходимо, "поднимем" виджет "cooler2" наверх относительно виджета "air_cooler" внутри виджета-контейнера "elCooler" с помощью панели инструментов размещения снизу. Зададим во вкладке "Атрибуты" в пункте "Геометрия" координаты "x" и "y" верхнего левого угла виджета "Вентилятор" в 75 и 30 пикселов соответственно. Изменим у унаследованного виджета "Вентилятор" альфа канал (прозрачность) цветов заливок (заполнений). Для этого во вкладке "Атрибуты" в полях "Цвет1" и "Цвет2" изменим значения цветов, добавив к ним "-200", где 200 - значение прозрачности ("0" - полностью прозрачный, а "255" - полностью непрозрачный), как показано на рис. 5.3.2.4.

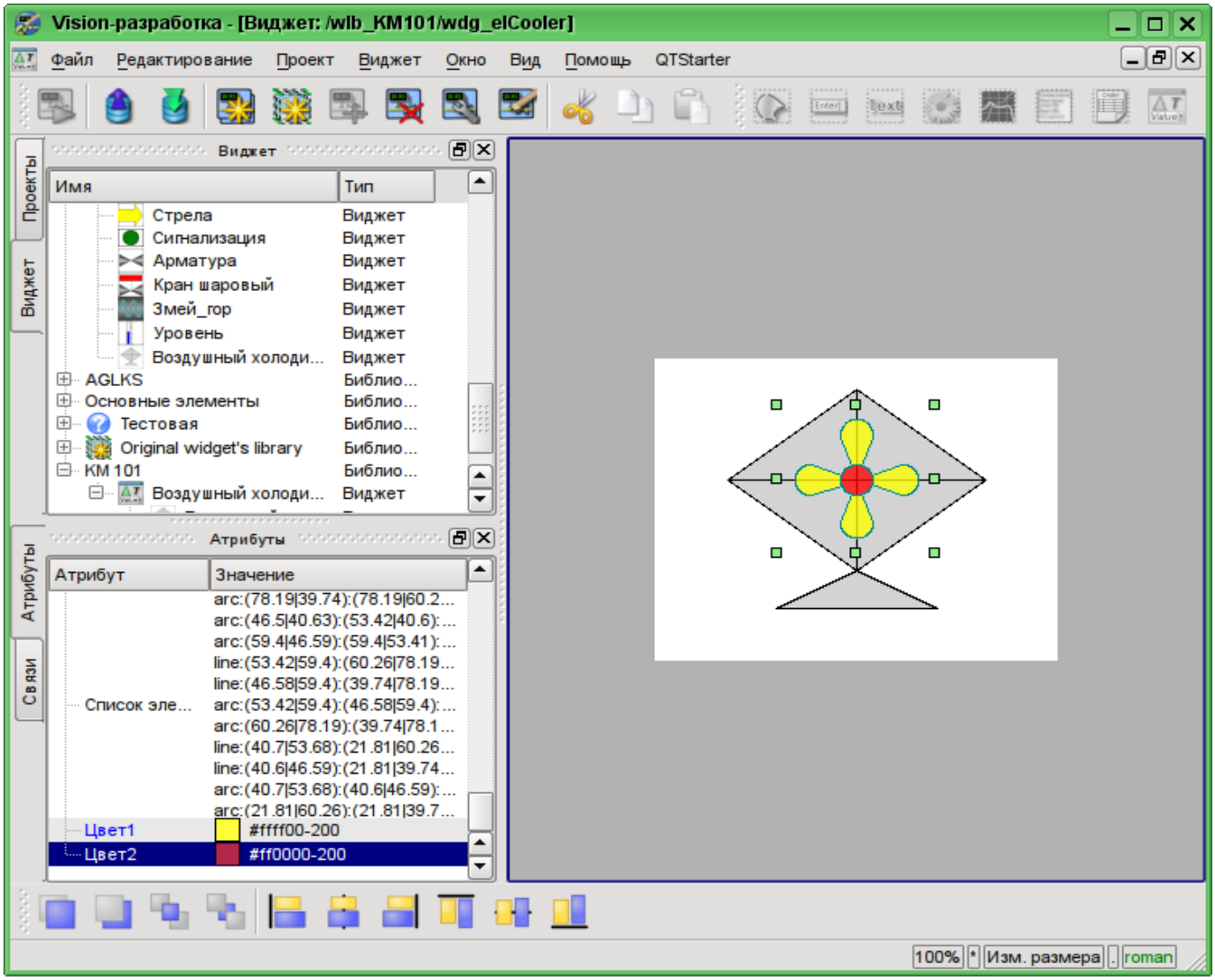


Рис. 5.3.2.4. Изменение прозрачности цветов заполнений у унаследованного виджета "cooler2".

Теперь добавим в виджет-контейнер "elCooler" два текстовых поля, основанных на примитиве "Текст", с целью отображения входной и выходной температур потока. Для этого в библиотеке "KM 101" выделим виджет "холодильник" и нажмем на панели визуальных элементов на иконку примитива "Текст", как это показано на рис 5.3.2.5.

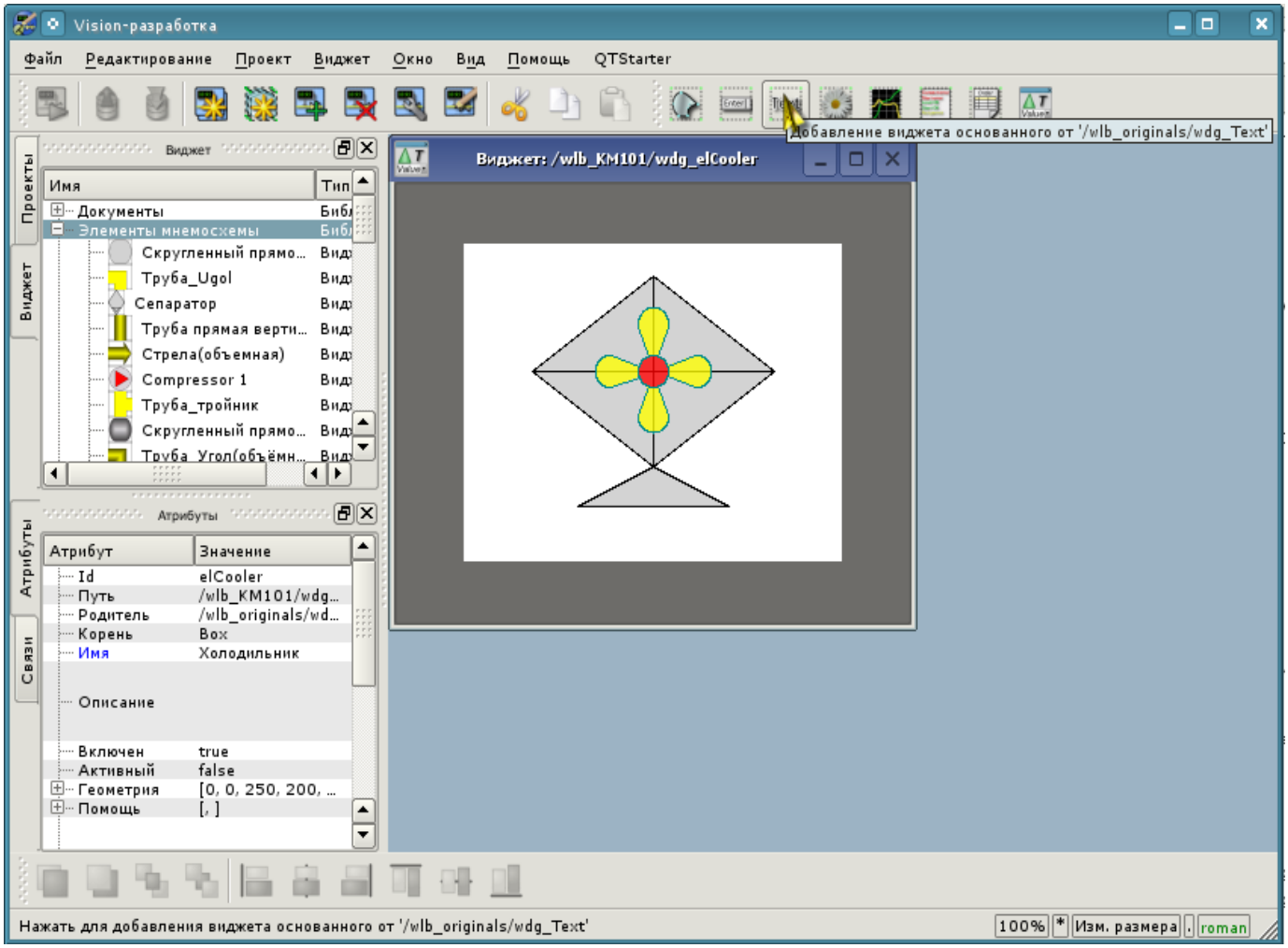


Рис. 5.3.2.5. Добавление в контейнер нового элемента, основанного на примитиве "Текст".

В результате появится диалог ввода идентификатора и имени вновь создаваемого элемента. Введем идентификатор "Ti" для первого текстового поля, а имя оставим пустым. Зададим геометрические размеры и координаты верхнего левого угла виджета, как это показано на рис. 5.3.2.6

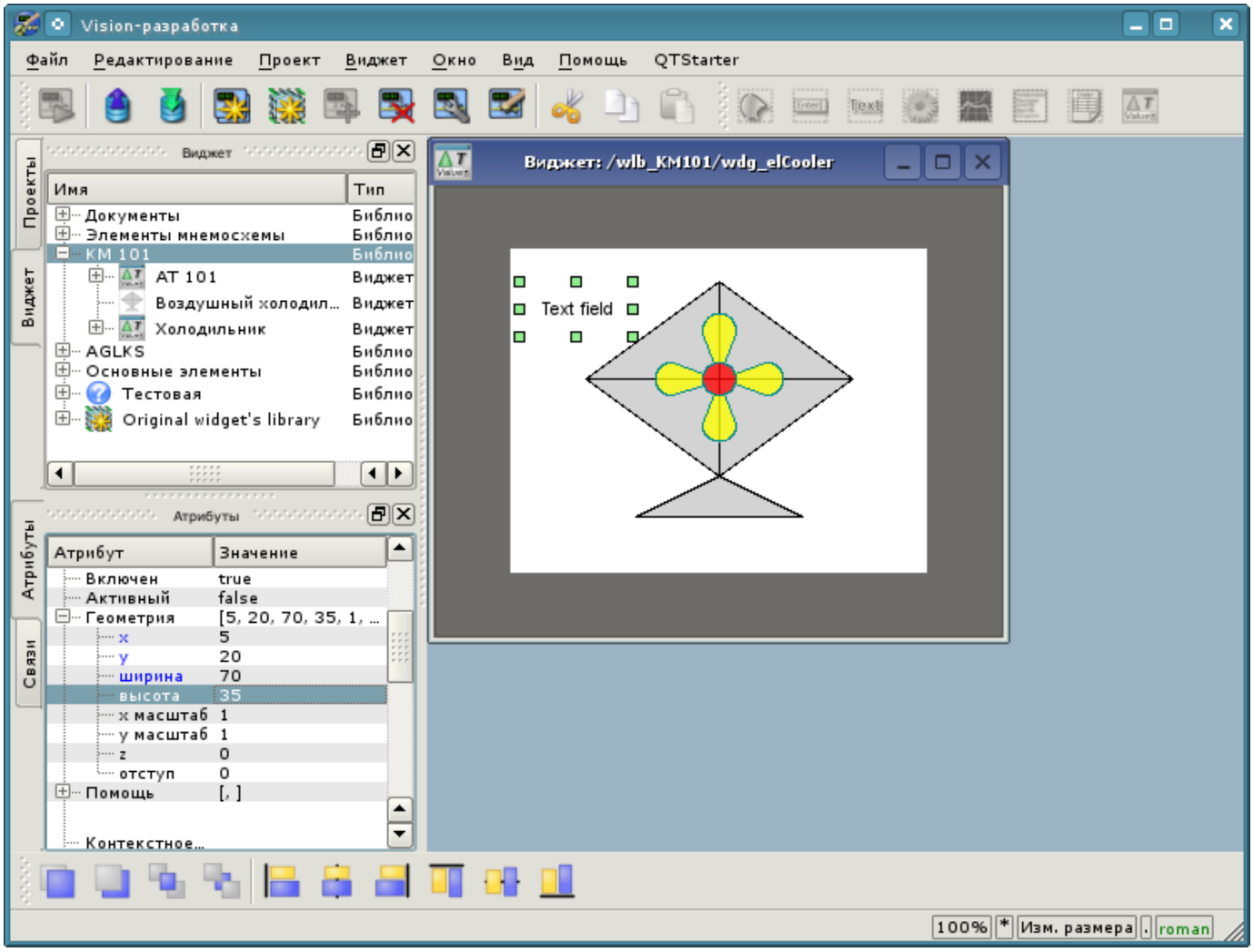


Рис. 5.3.2.6. Задание геометрии виджета "Ti".

Изменим размер и установим усиление шрифта для этого элемента (рис. 5.3.2.7). Обратим внимание, что измененные поля у унаследованных виджетов подсвечиваются синим цветом для удобства отслеживания изменений и последующей их "очистки"(отката) с помощью клика правой кнопкой манипулятора "мышь" по измененному атрибуту.

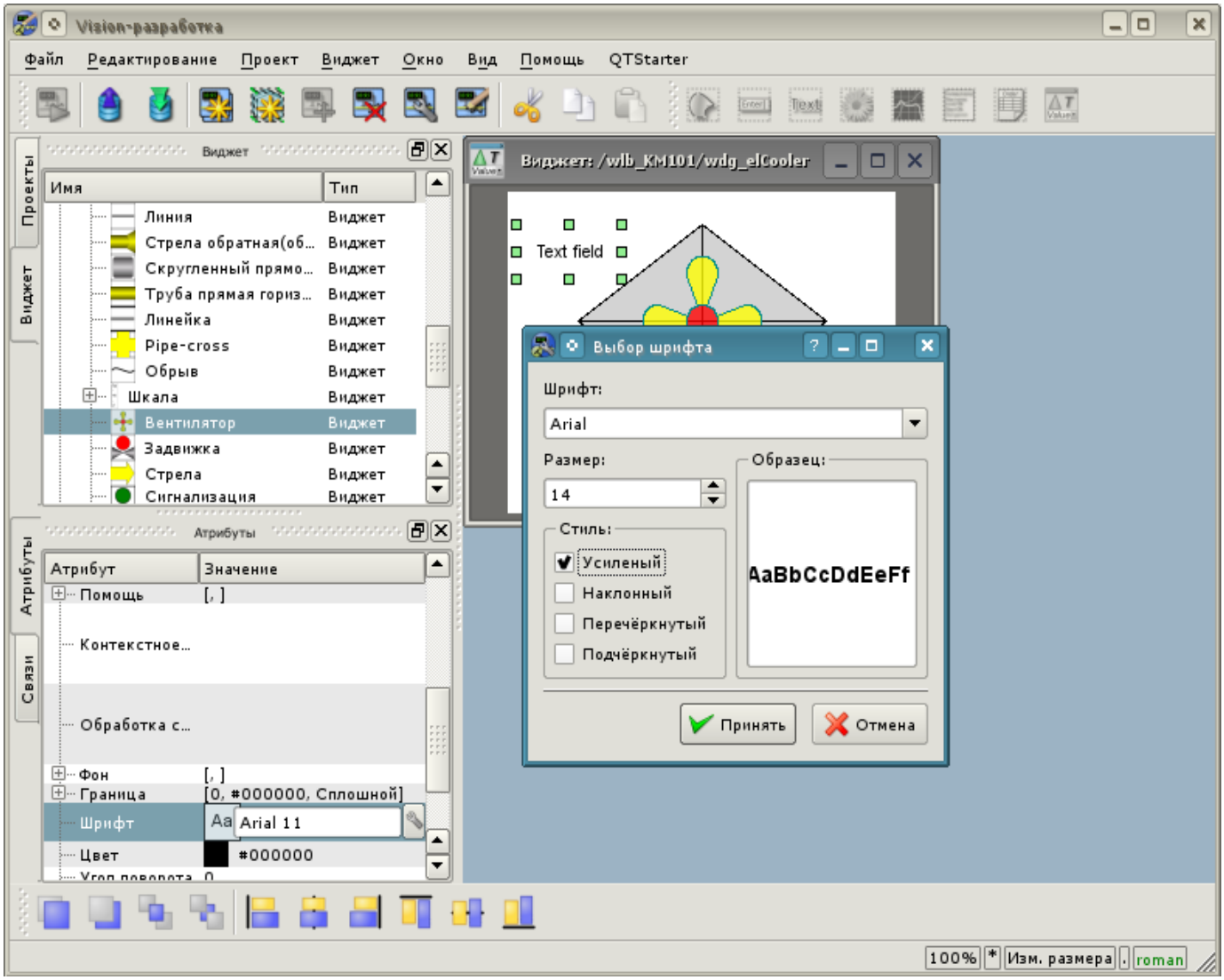


Рис. 5.3.2.7. Изменение размера шрифта для виджета "Ti".

Теперь изменим поле "Текст" виджета "Ti", указав в нем наличие аргумента "%1", в который впоследствии будет передаваться реальное значение входной температуры (рис. 5.3.2.8).

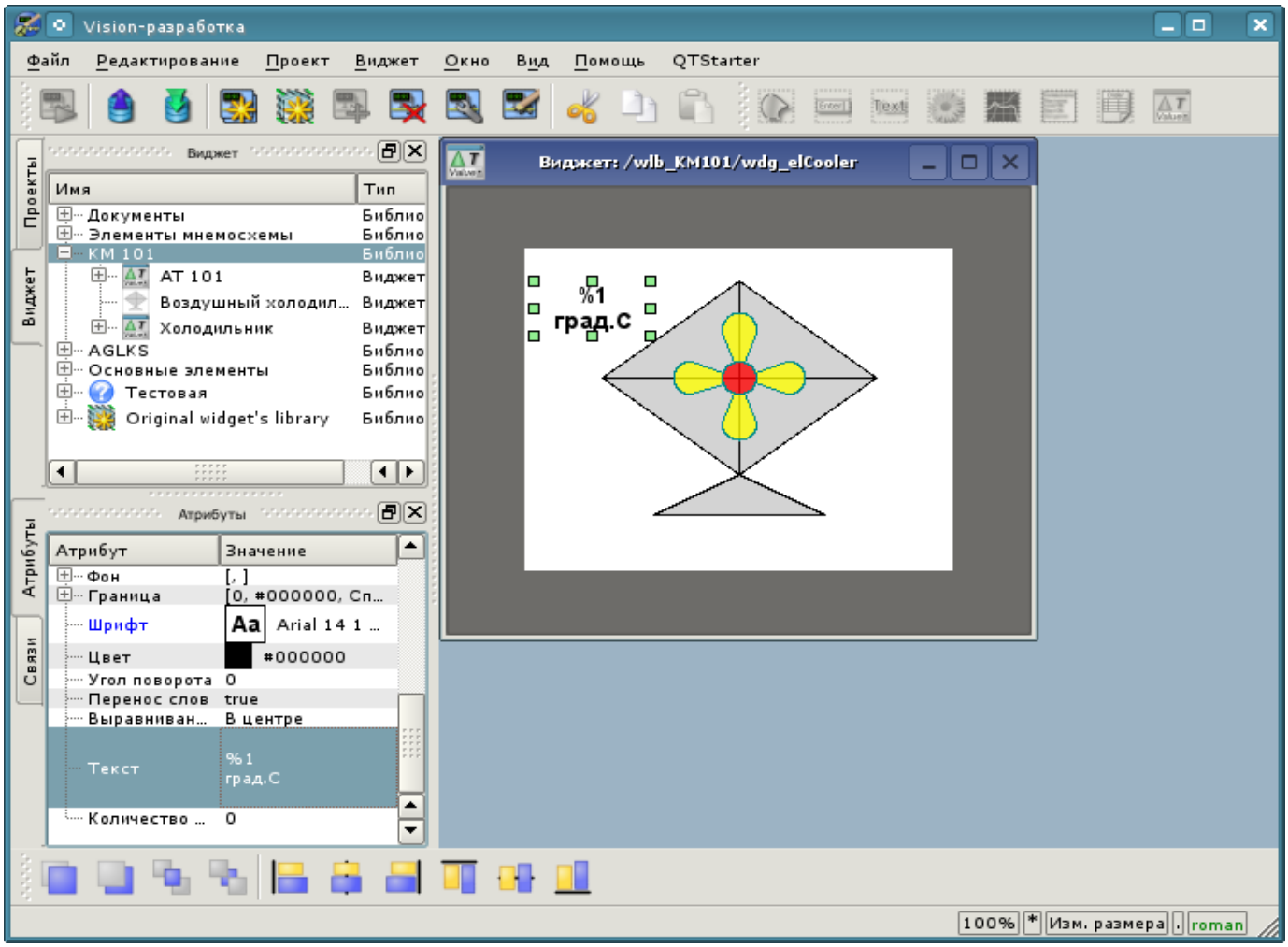


Рис. 5.3.2.8. Изменение поля "Текст" и указание в нем наличия атрибута для виджета "Ti".

Далее в перечне атрибутов виджета "Ti" в пункте "Количество аргументов" укажем "1" и сконфигурируем аргумент (рис. 5.3.2.9). Число "300.25" задано лишь только с целью наглядности, в режиме исполнения оно будет подменяться реальным значением входной температуры.

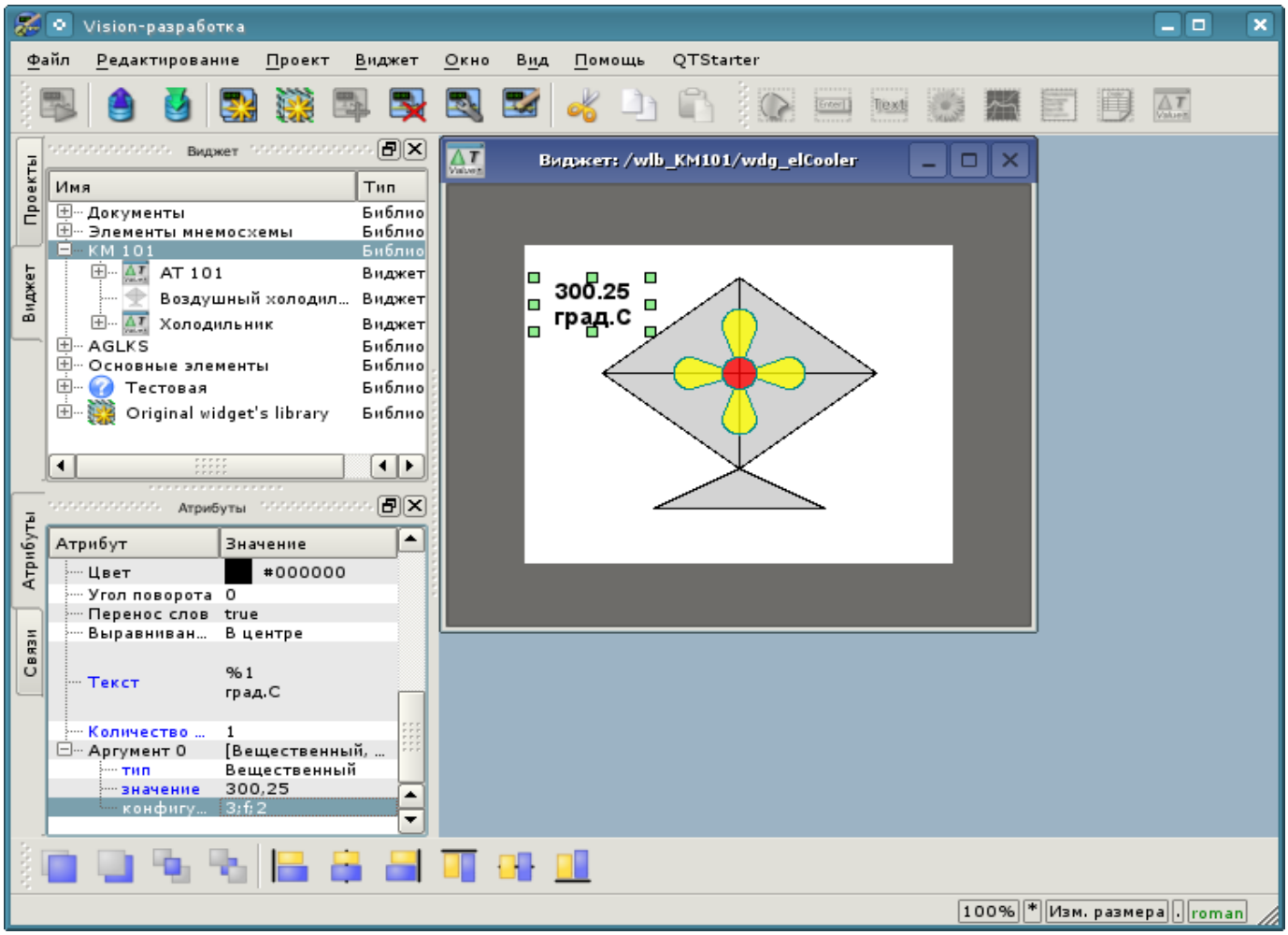


Рис. 5.3.2.9. Конфигурация аргумента для виджета "Ti".

Теперь скопируем виджет "Ti" с целью создания аналогичного ему виджета "To" (выходная температура), как это показано на рисунке 5.3.2.10.

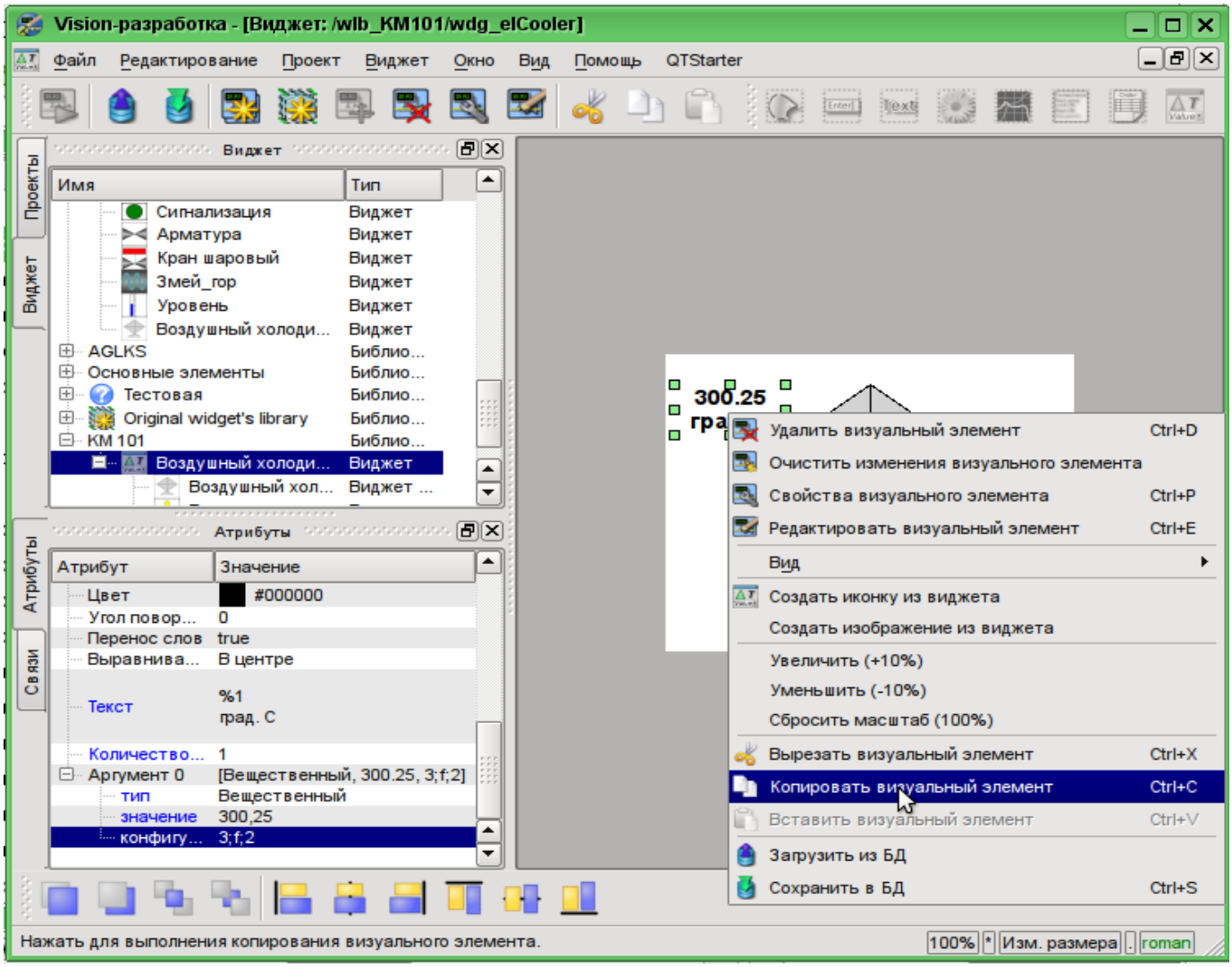


Рис. 5.3.2.10. Копирование виджета "Ti".

Вставим скопированный виджет в виджет-контейнер "Холодильник" в библиотеке "KM 101" (рис. 5.3.2.11). В диалоге ввода идентификатора и имени для вновь созданного виджета в поле "ID" укажем "To", а имя оставим пустым.

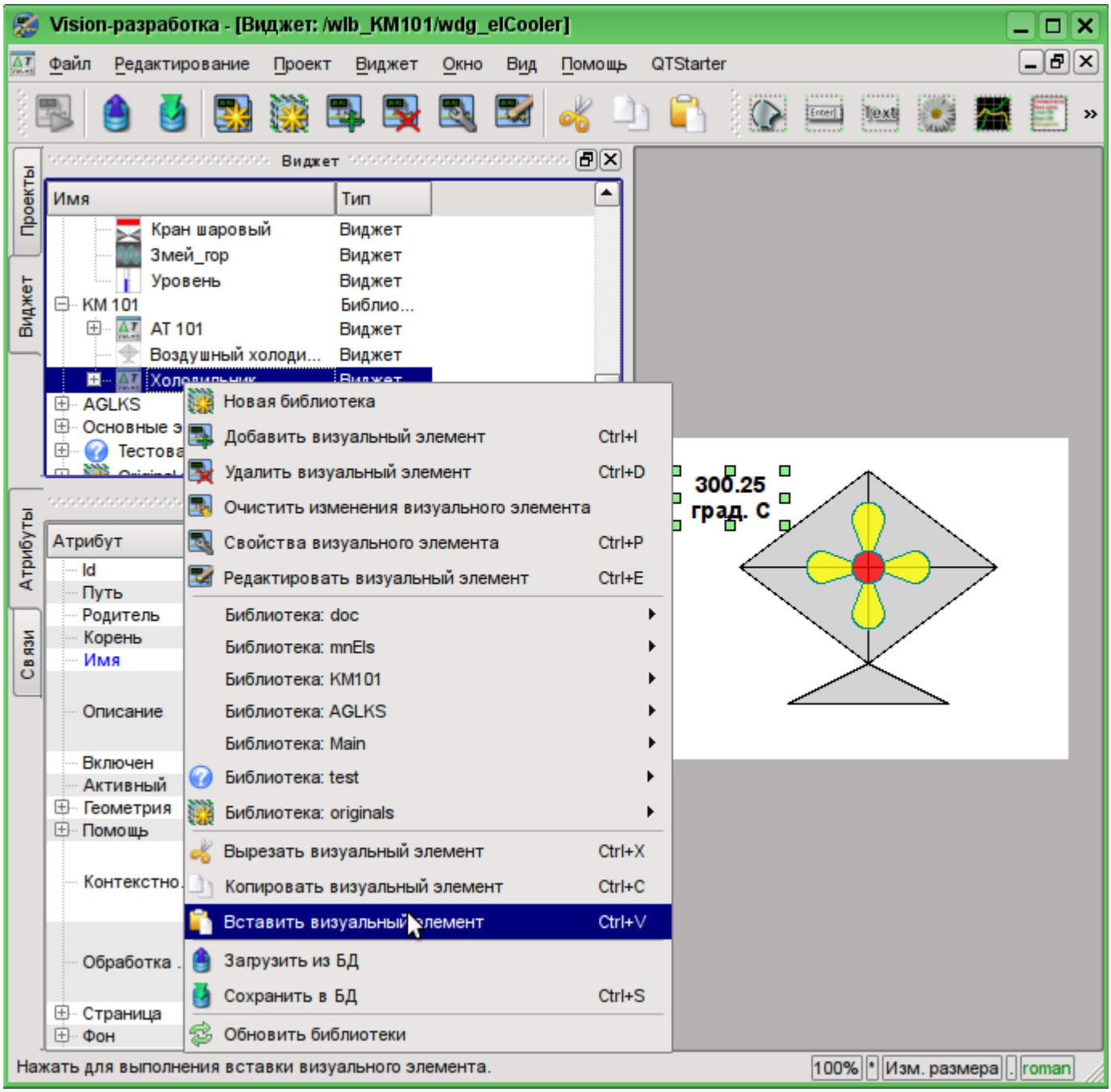


Рис. 5.3.2.11. Вставка скопированного виджета в виджет-контейнер "Воздушный холодильник" библиотеки "KM 101" .

Изменим геометрию виджета "To", как показано на рис. 5.3.2.12.

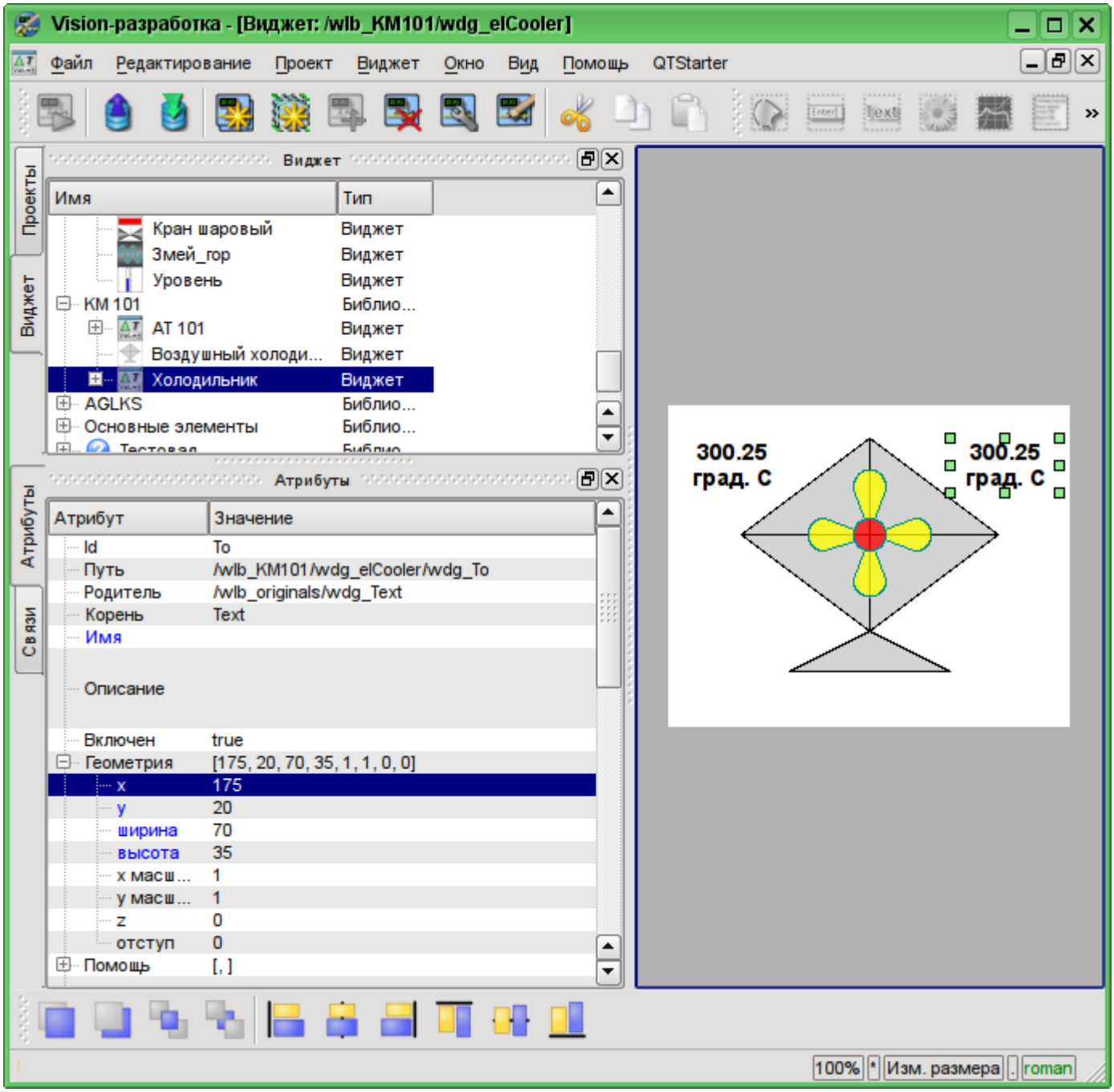


Рис. 5.3.2.12. Изменение геометрии виджета "To".

Теперь добавим виджет, основанный на примитиве "Элементы формы", который будем использовать в качестве ComboBox для выбора заданий производительности холодильника. В качестве идентификатора укажем "cw", и поле "Имя" оставим пустым. (рис. 5.3.2.13)

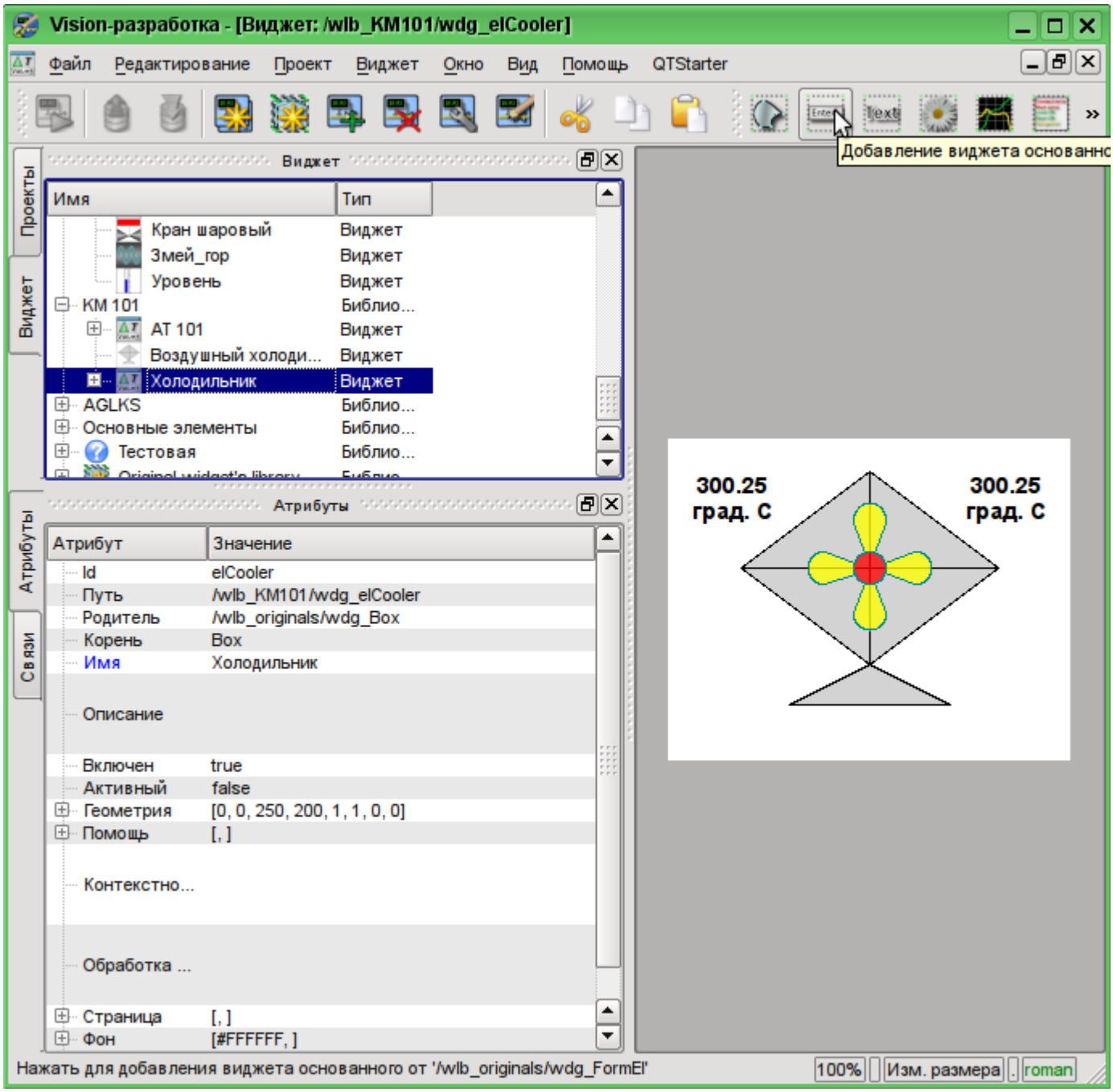


Рис. 5.3.2.13. Добавление виджета, основанного на примитиве "Элементы формы".

Зададим параметры пункта "Геометрия" во вкладке "Атрибуты" для вновь добавленного виджета: координаты "x", "y" верхнего левого угла, ширину и высоту в 60, 158, 60, 40 соответственно. Изменим "Тип элемента" в Combo Box, как это показано на рис. 5.3.2.14.

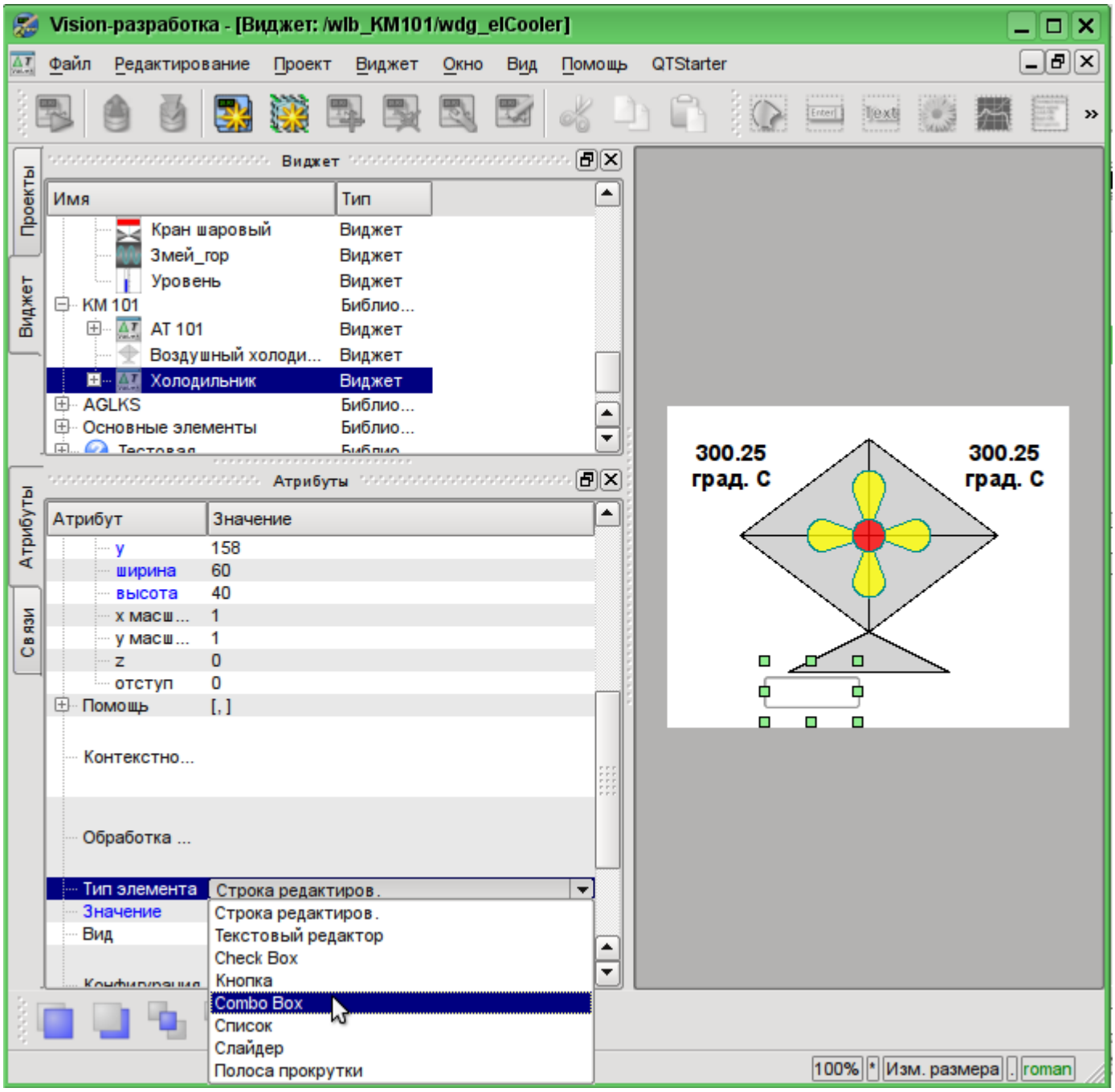


Рис. 5.3.2.14. Изменение "Геометрии" и "Типа элемента" для вновь созданного виджета.

Заполним поля: "Значение", "Элементы" и "Шрифт", как это показано на рис. 5.3.2.15. Кроме этого важно комбобокс поднять над всеми и сделать активным. Для активизации виджета комбобокса нужно установить соответствующее свойство для него.

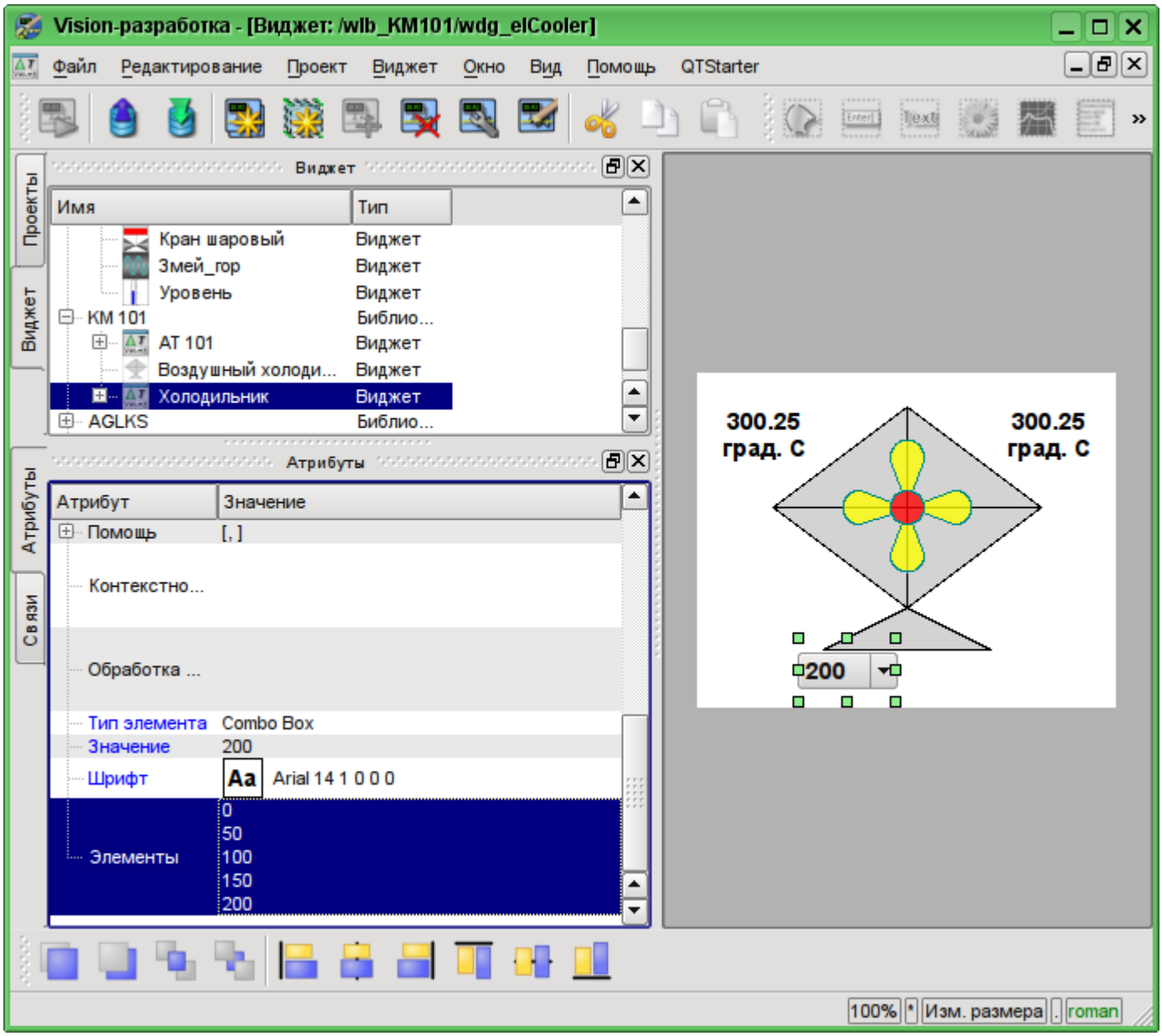


Рис. 5.3.2.15. Заполнение параметров ComboBox "cw".

Для отображения единицы измерения производительности холодильника добавим виджет на основе примитива "Текст". Делаем ту же процедуру, что и для виджета "Ti". Идентификатором вновь созданного виджета сделаем "dimension", изменим геометрию: координаты верхнего левого угла "x", "y" укажем в 125, 168 соответственно, а ширину и высоту - в 60 и 20 соответственно. Поменяем размер шрифта на "14 усиленный", а в поле "Текст" впишем "об./мин.", что и будет нашей единицей измерения (рис. 5.3.2.16).

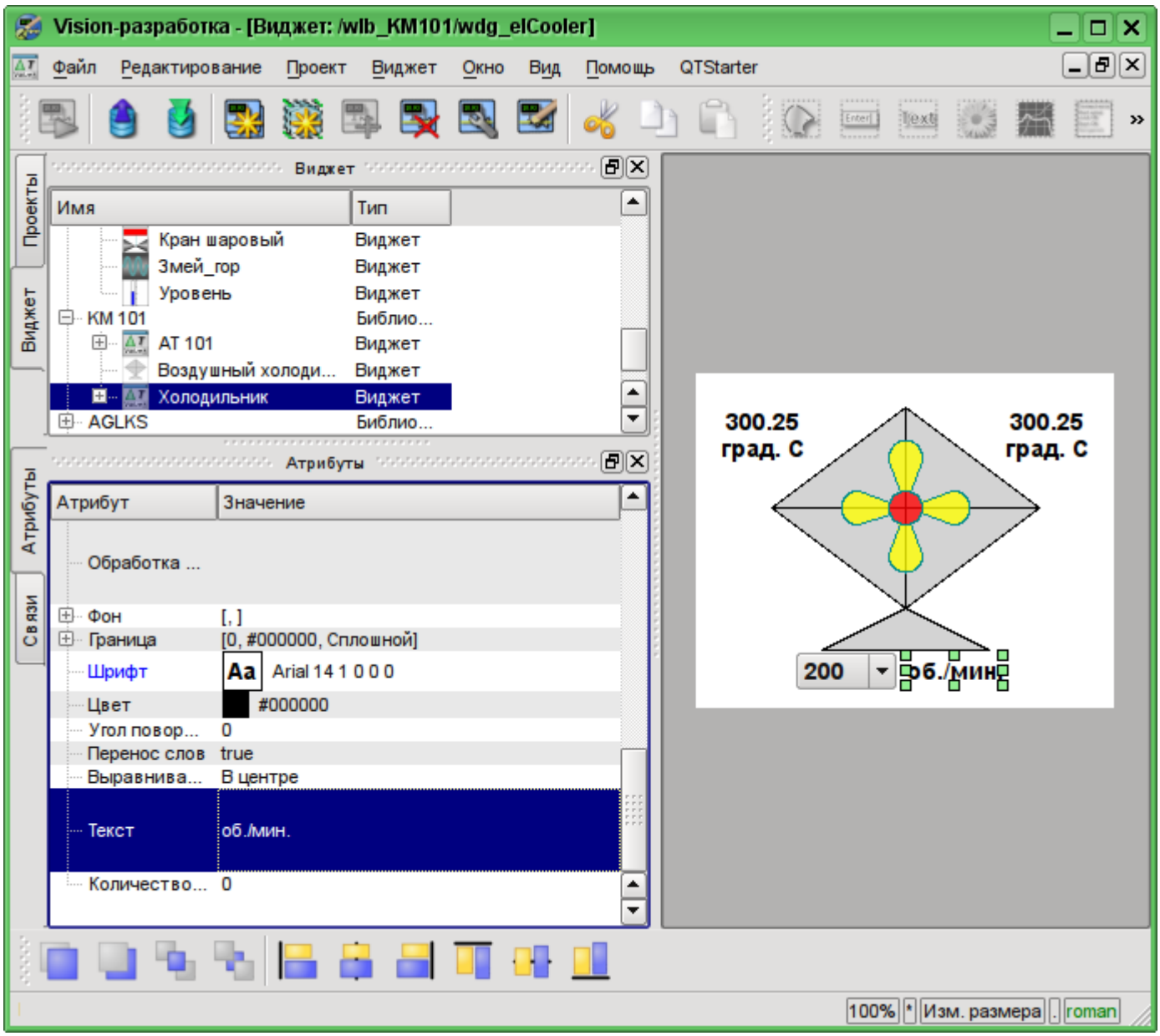


Рис. 5.3.2.16. Добавление виджета "dimension", основанного на примитиве "Текст" и изменение его параметров.

Для добавления логики обработки виджета "Холодильник"(elCooler) откроем диалог редактирования свойств этого визуального элемента и перейдём на вкладку "Обработка". На этой вкладке мы увидим дерево атрибутов виджета и поле текста программы, для обработки атрибутов. Для решения нашей задачи нужно добавить три атрибута: Ti, To, Cw (рис. 5.3.2.17). Для добавления атрибута нужно развернуть корневой элемент ".", выбрать любой элемент внутри корневого и нажать кнопку "Добавить атрибут" снизу.

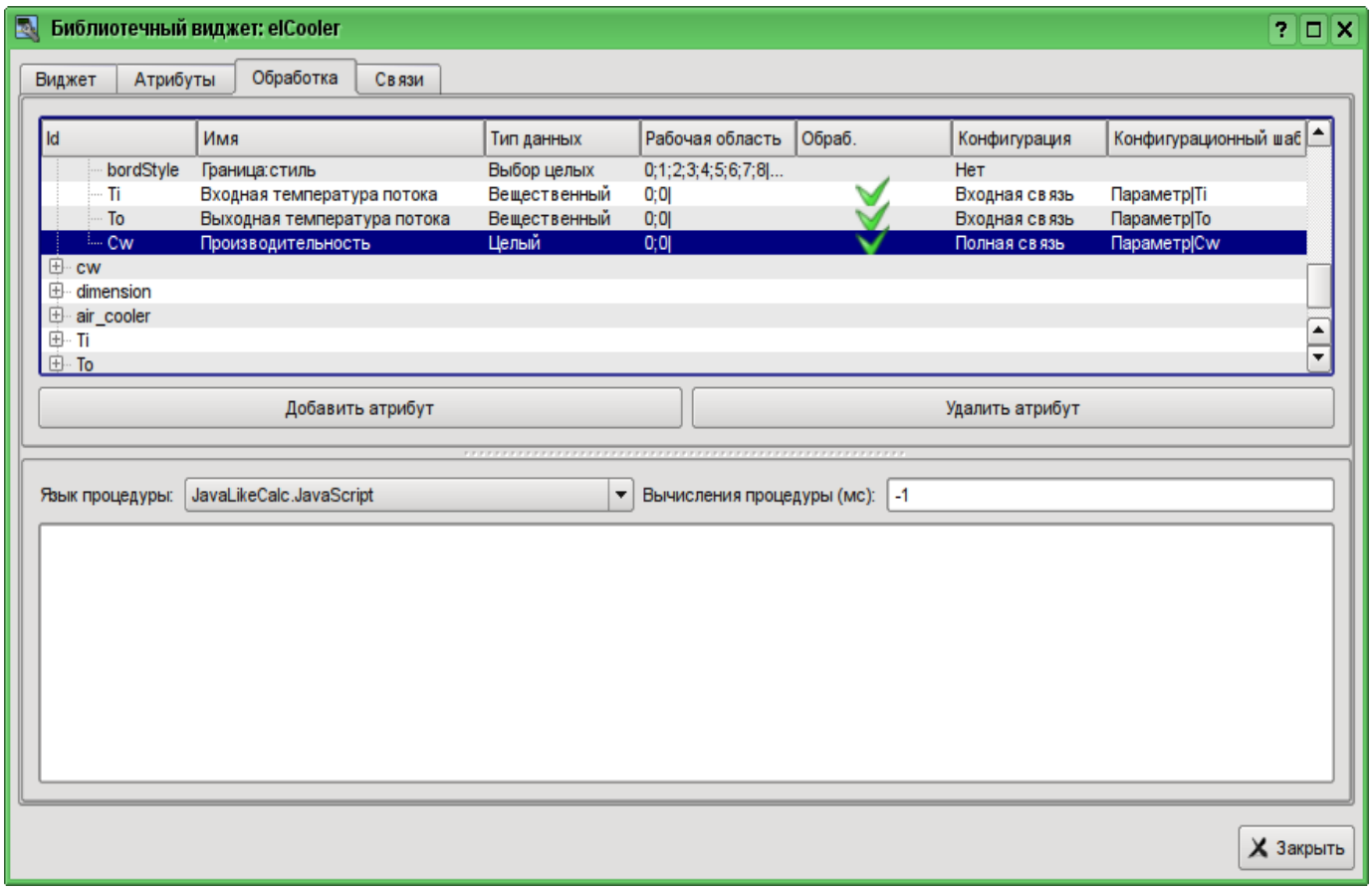


Рис. 5.3.2.17. Добавление трех атрибутов для элемента "elCooler" библиотеки "KM 101".

Далее включим в обработку атрибут "value" комбобокса "cw", как это показано на рис. 5.3.2.18. Аналогично включим в обработку атрибут "arg0val" для Ti и To, а также атрибут "speed" у элемента "cooler2".

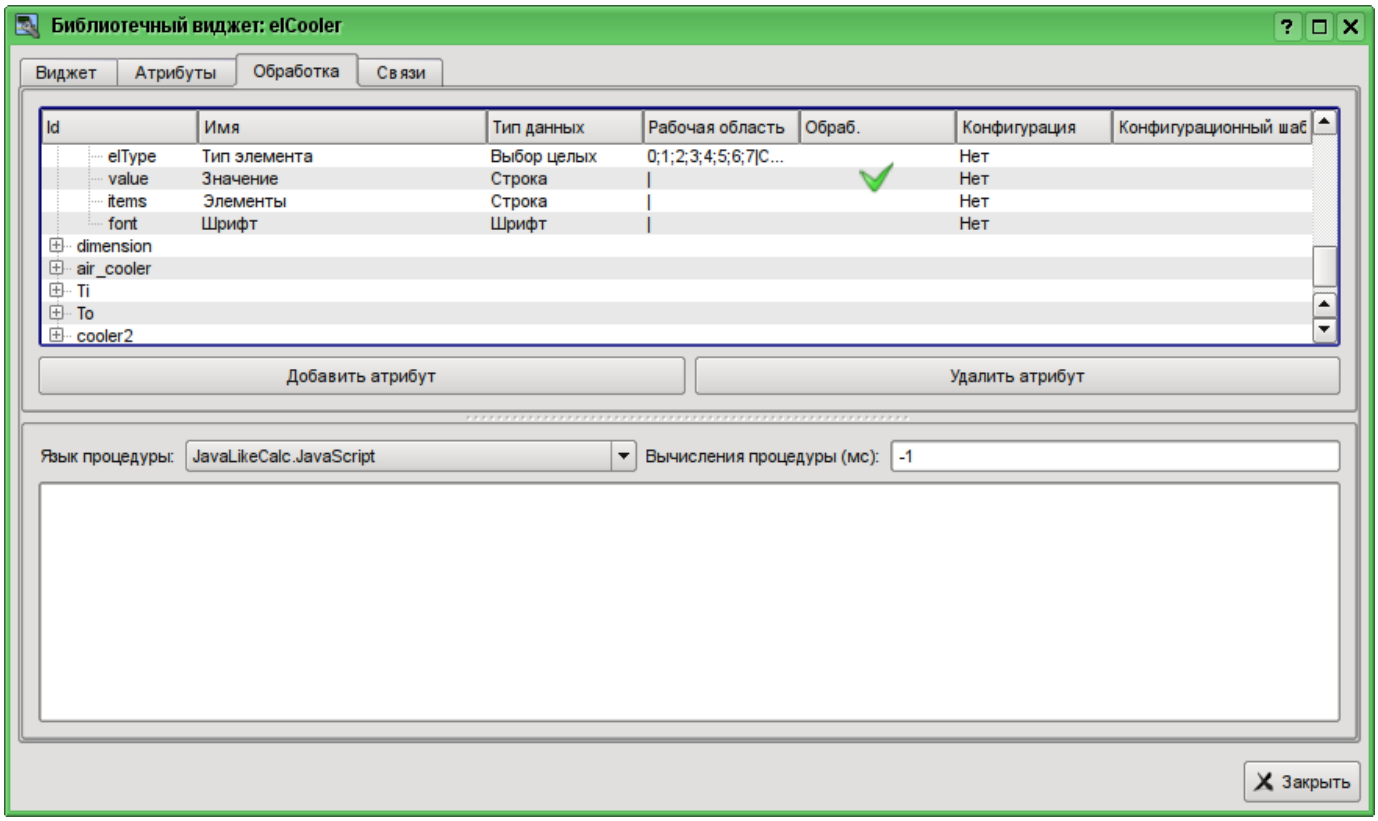


Рис. 5.3.2.18. Включение в обработку атрибута "value" комбобокса "cw".

В завершении установим язык пользовательского программирования для процедуры в "JavaLikeCalc.JavaScript" и напишем программу обработки этого виджета:

```
Ti_arg0val = Ti;
To_arg0val = To;

ev_wrk=ev_rez="";
off=0;
while(true)
{
      ev_wrk=Special.FLibSYS.strParse(event,0,"\n",off);
      if( ev_wrk == "" ) break;
      if( ev_wrk == "ws_CombChange:/cw" ) Cw = cw_value;
      else ev_rez += ev_wrk+"\n";
}
cw_value = Cw;
cooler2_speed = Cw/5;
```

Результирующий вид вкладки обработка виджета "elCooler" библиотеки "KM 101" будет иметь вид, показанный на рис. 5.3.2.19.

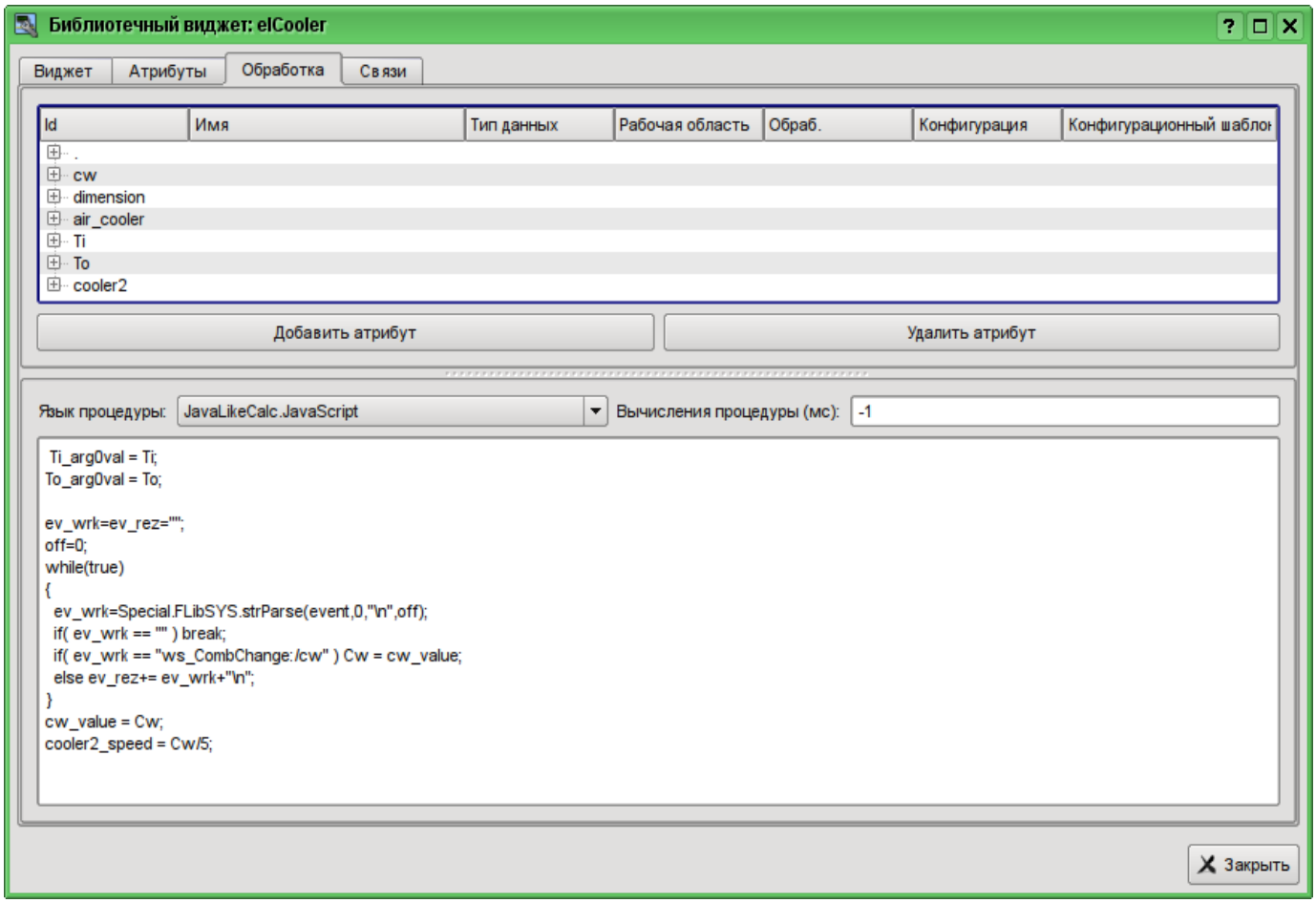


Рис. 5.3.2.19. Результирующий вид вкладки обработка виджета "elCooler" библиотеки "KM 101".

Закроем диалог редактирования свойств визуального элемента, создадим иконку на основе нашего элемента, закроем внутреннее окно редактирования и сохраним это всё.

На этом разработку комплексного элемента можно считать законченной.

5.3.3. Добавление комплексного элемента на мнемосхему

Для проверки работоспособности и оценки результатов наших усилий добавим созданный виджет на мнемосхему, разработанную в разделе 5.2. Выполним эту операцию для двух холодильников "AT101_1" и "AT101_2".

Для этого откроем кадр мнемосхемы "AT 101" на редактирование. После чего хватаем "мышью" наш комплексный элемент и тащим на мнемосхему, где отпускаем в нужной нам позиции. В диалоге запроса имени вводим идентификаторы "AT101_1" и "AT101_2" соответственно. Поле имени опускаем. Добавленные элементы располагаем как нам удобно. После выполнения подобных манипуляций у нас должна получиться мнемосхема с видом, похожим представленной на рис.5.3.3.1.

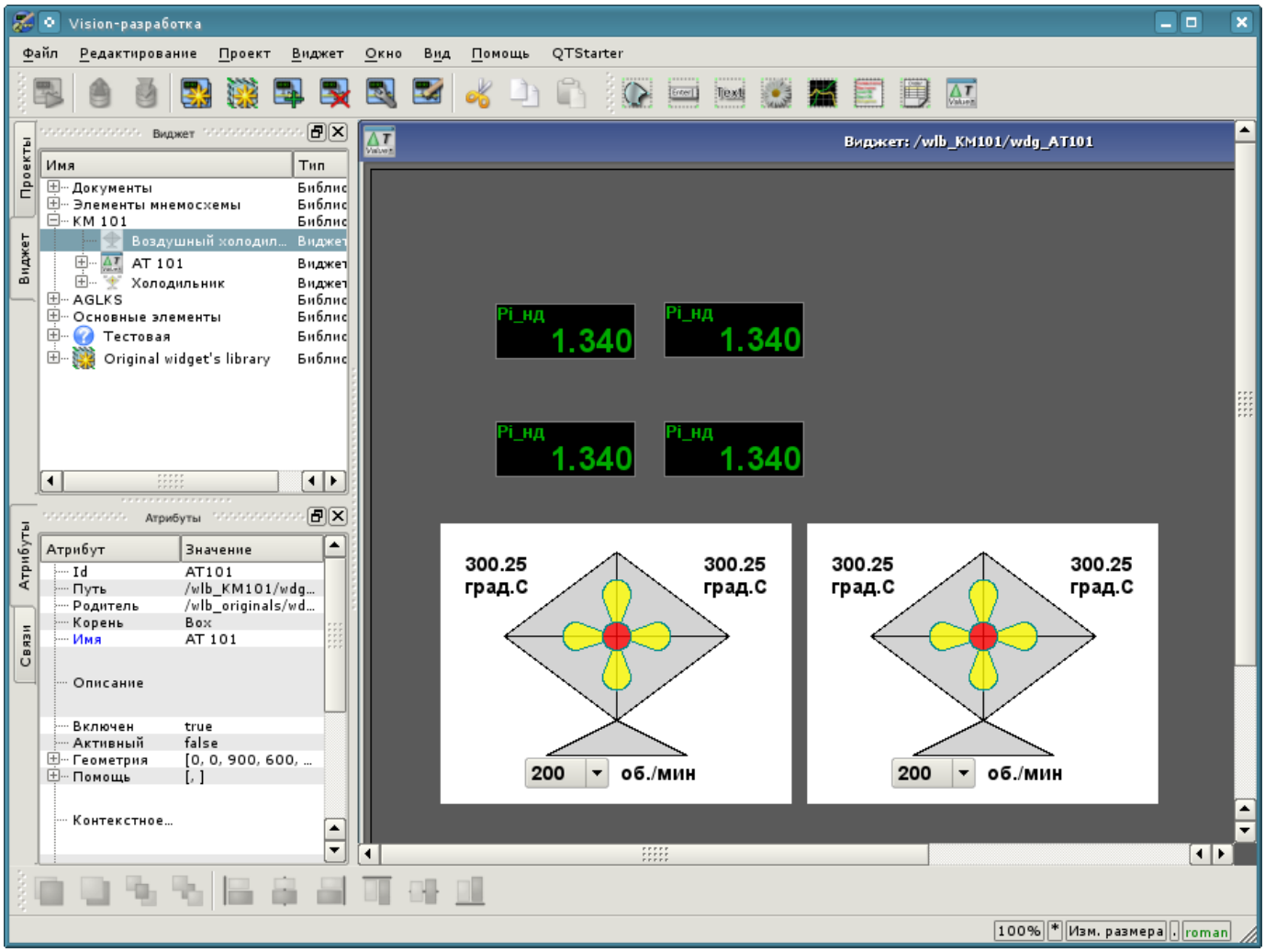


Рис. 5.3.3.1. Вид мнемосхемы с комплексными элементами.

Сохраним новую мнемосхему и закроем её окно. Далее перейдём в проект и откроем эту мнемосхему в дереве проекта "Группы сигнализаций (шаблон)"->"Корневая страница"->"Группа 1"->"Мнемосхемы"->"AT 101". Как можно заметить, наши новые элементы появились здесь автоматически. И нам осталось только подключить связи к новым элементам. Для этого откроем диалог редактирования свойств мнемосхемы на вкладке "Связи" (рис.5.3.3.2). На этой вкладке мы увидим дерево с элементами "AT101_1" и "AT101_2". Развернув любой из элементов, мы увидим ветку "Параметр" как раз с атрибутами "Ti", "To" и "Cw", Таким образом, мы можем просто указать адрес параметра "prm:/LogicLev/KM101/AT101_1" в поле "Параметр", а атрибуты будут расставлены автоматически.

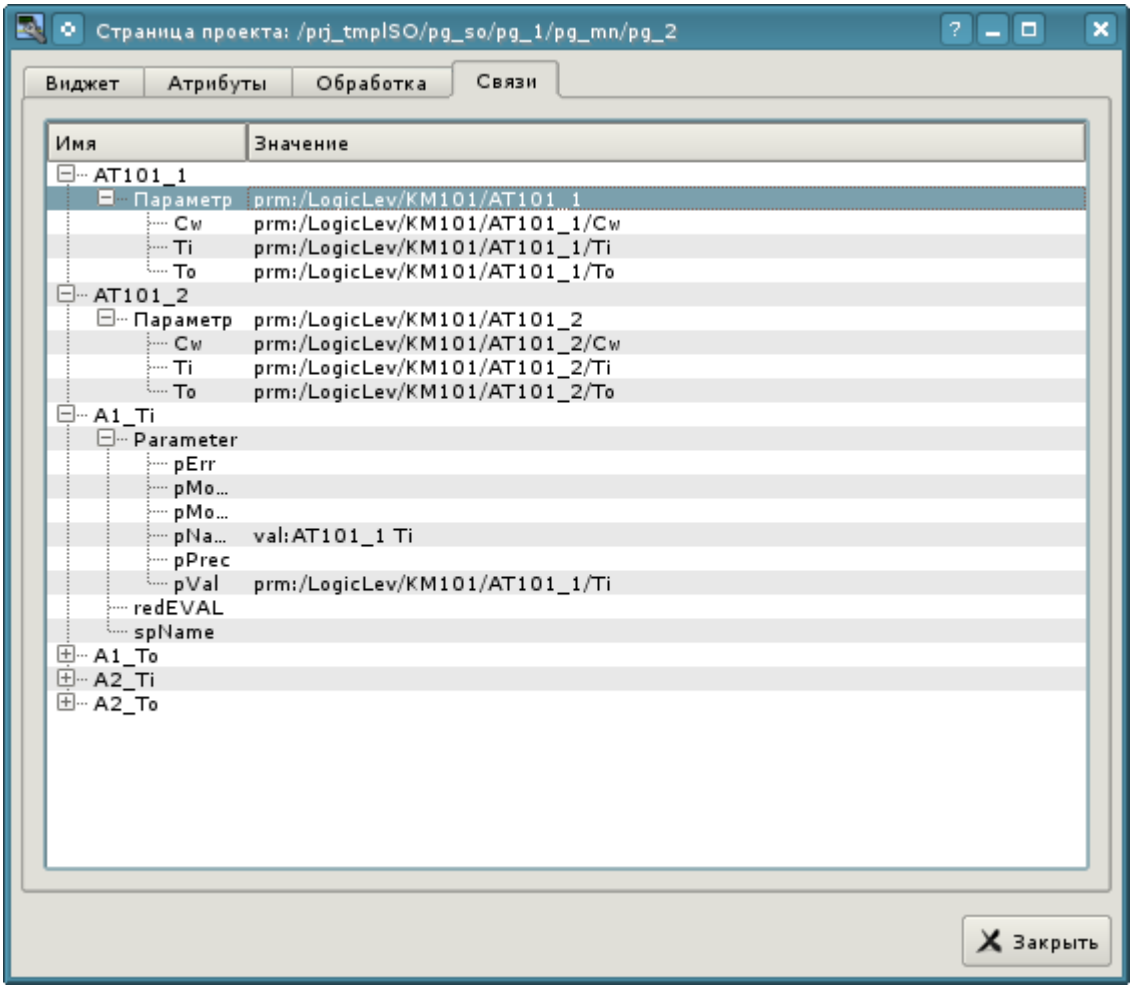


Рис. 5.3.3.2. Вкладка "Связи" диалога редактирования свойств мнемосхемы.

Сохраним нашу мнемосхему и проверим, что получилось. Для этого закроем окно диалога свойств и запустим проект "Группы сигнализаций (шаблон)" на исполнение. Затем переключимся на вторую мнемосхему кнопками листания. При безошибочной конфигурации мы должны увидеть что-то подобное изображённому на рис.5.3.3.3.

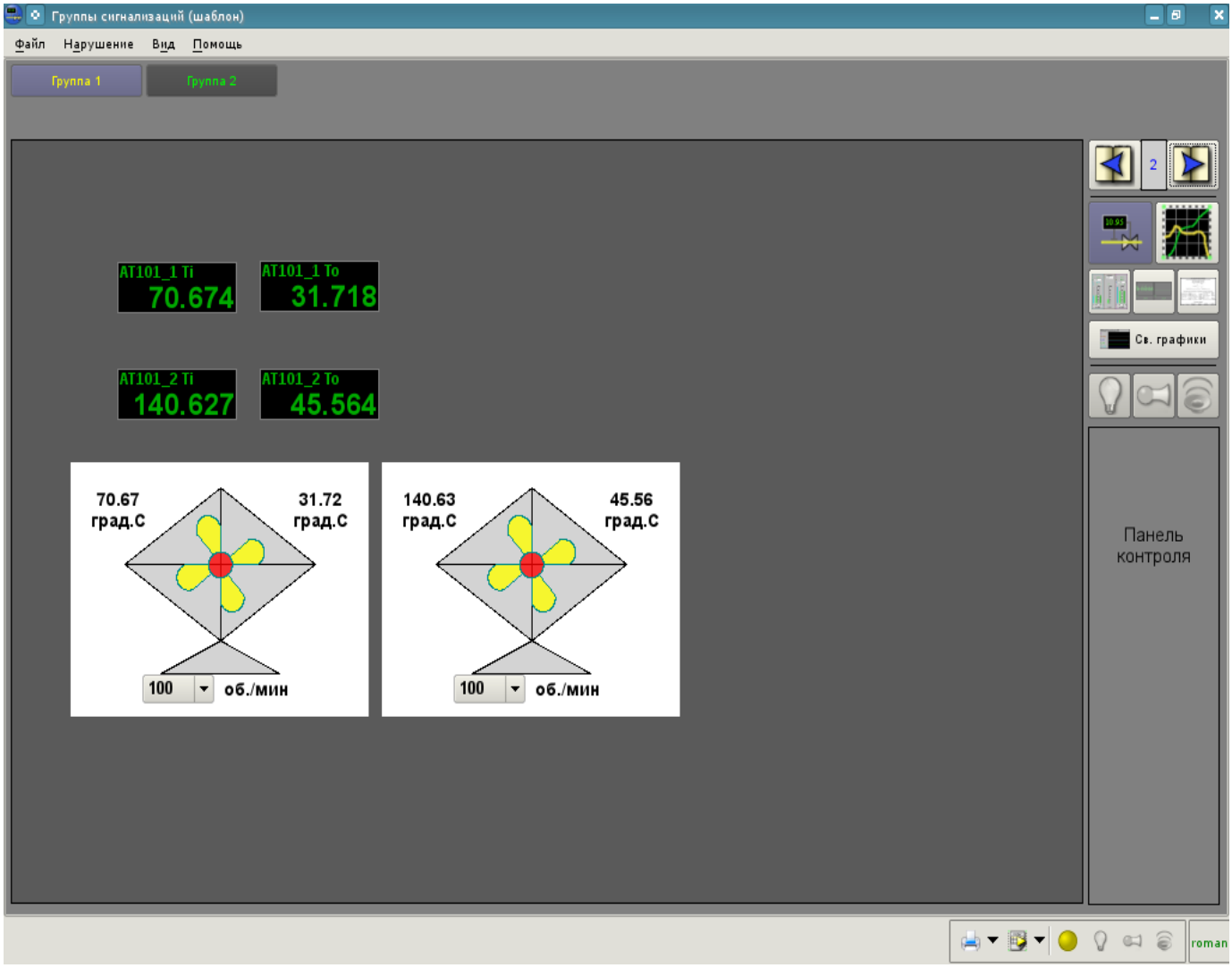


Рис. 5.3.3.3. Результирующая мнемосхема.

На этой мнемосхеме посредством наших комплексных элементов мы можем не только наблюдать но и управлять производительностью холодильников, просто меняя значение в комбобоксе. Меняя производительность, мы можем заметить и изменение температуры. Историю изменения мы можем увидеть на созданной нами в разделе 5.1 группе графиков.

6. Рецепты

Данный раздел предназначен для предоставления описания рецептов решения часто встречающихся проблем и задач пользователя. Рецепты решения задач и проблем для помещения в этот раздел могут предлагаться пользователями.

Заключение

Данный документ достаточно детально описывает основной процесс создания элементов пользовательского интерфейса, с предварительной подготовкой и конфигурацией источника данных. В целом это позволяет достаточно быстро получить представление о работе с системой OpenSCADA, а также целенаправленно искать решения попутных проблем.

Модуль подсистемы “Архивы” <FSArch>

Модуль:	FSArch
Имя:	Архиватор на файловую систему
Тип:	Архив
Источник:	arh_FSArch.so
Версия:	1.4.1
Автор:	Роман Савоченко
Описание:	Модуль архива. Предоставляет функции архивирования сообщений и значений на файловую систему.
Лицензия:	GPL

Модуль предназначен для архивирования сообщений и значений системы OpenSCADA на файловую систему.

Любая SCADA система предоставляет возможность архивирования собранных данных, т.е. формирование истории изменения (динамики) процессов. Архивы условно можно разделить на два типа: архивы сообщений и архивы значений.

Особенностью архивов сообщений является то, что архивируются, так называемые, события. Характерным признаком события является его время возникновения. Архивы сообщений обычно используются для архивирования сообщений в системе, т.е. ведение логов и протоколов. В зависимости от источника сообщения могут классифицироваться по различным критериям. Например, это могут быть протоколы аварийных ситуаций, протоколы действий операторов, протоколы сбоев связи и др.

Особенностью архивов значений является их периодичность, определяемая промежутком времени между двумя смежными значениями. Архивы значений применяются для архивирования истории непрерывных процессов. Поскольку процесс непрерывный, то и архивировать его можно только путём введения понятия квантования времени опроса, поскольку иначе мы получаем архивы бесконечных размеров ввиду непрерывности самой природы процесса. Кроме этого, практически мы можем получать значения с периодом ограниченным самими источниками данных. Например, довольно качественные источники данных, в промышленности, редко позволяют получать данные с частотой более 1кГц. И это без учёта самих датчиков, имеющих ещё менее качественные характеристики.

Для ведения архивов в системе OpenSCADA предусмотрена подсистема «Архивы». Данная подсистема, в соответствии с типами архивов, состоит из двух частей: архив сообщений и архивы значений. Подсистема в целом является модульной, что позволяет создавать архивы, основанные на различной природе и способах хранения данных. Данный модуль предоставляет механизм архивирования на файловую систему как для потока сообщений, так и для потока значений.

1. Архиватор сообщений

Архивы сообщений формируются архиваторами. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделять архивирование различных классов сообщений.

Архиватор сообщений этого модуля позволяет хранить данные как в файлах формата языка XML, так и в формате плоского текста. Язык разметки XML является стандартным форматом, который с лёгкостью понимают многие сторонние приложения. Однако открытие и разбор файлов в таком формате требует значительных ресурсов. С другой стороны, формат плоского текста требует значительно меньше ресурсов, хотя и не является унифицированным, а также требует знания его структуры для разбора.

В любом случае поддерживаются оба формата и пользователь может выбрать любой из них, в соответствии со своими требованиями.

Файлы архивов именуются архиваторами, исходя из даты первого сообщения в архиве. Например так: <2006–06–21 17:11:04.msg>.

Файлы архивов могут ограничиваться по размеру и времени. После превышения лимита создаётся новый файл. Максимальное количество файлов в директории архиватора также может быть ограничено. После превышения лимита на количество файлов, старые файлы начнут удаляться!

С целью оптимизации использования дискового пространства архиваторы поддерживают упаковку старых архивов упаковщиком gzip. Упаковка производится после продолжительного неиспользования архива.

При использовании архивов в формате языка XML соответствующие файлы загружаются целиком! Для выгрузки неиспользуемых продолжительное время архивов применяется таймаут доступа к архиву, после превышения которого архив выгружается из памяти, а затем и пакуется.

Модулем предоставляются дополнительные параметры настройки процесса архивирования (рис.1).

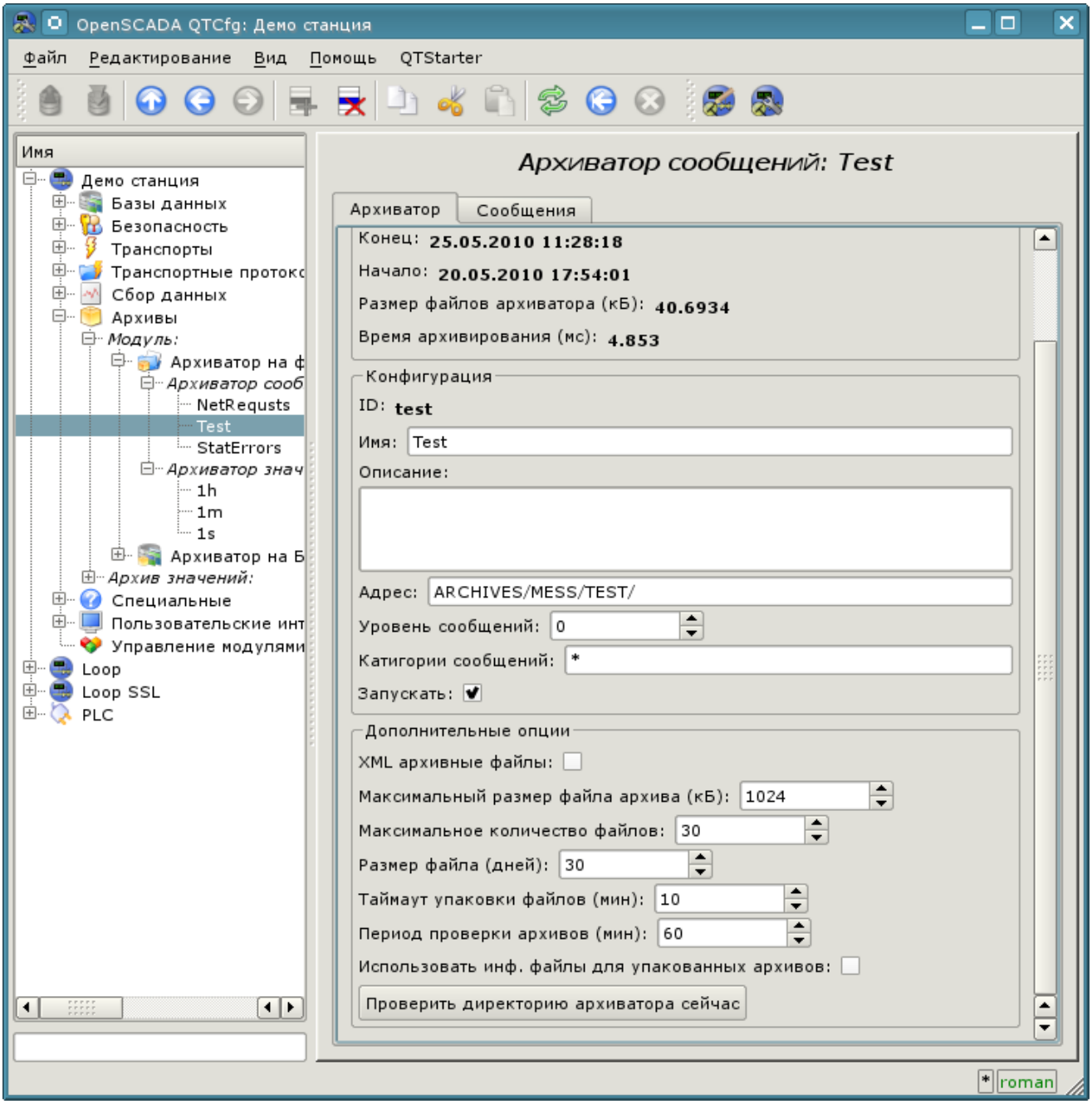


Рис.1. Дополнительные параметры настройки процесса архивирования сообщений модуля FSArch

В число этих параметров входят:

- В число этих параметров входят:
- Выбора XML-формата архивных файлов.
- Максимальный размер одного файла архива.
- Максимальное количество архивных файлов.
- Ограничение размера файла архива по времени.
- Таймаут упаковки файлов архива.
- Периодичность проверки файлов архиватором на предмет поиска новых архивов и удаление старых.
- Включение использования информационных файлов для упакованных файлов.
- Команда немедленной проверки директории архиватора. Может использоваться при размещения в директории архиватора файлов архивов из другой станции.

1.1. Формат файлов архива сообщений

В таблице ниже приведен синтаксис файла архива, построенного на XML-языке:

Тег	Описание	Атрибуты	Содержит
FSArch	Корневой элемент. Идентифицирует файл как принадлежащий данному модулю.	*Version* — версия файла архива; *Begin* — время начала архива (hex – UTC в секундах от 01/01/1970); *End* — время окончания архива (hex – UTC в секундах от 01/01/1970).	(m)
m	Тег отдельного сообщения.	*tm* — время создания сообщения (hex – UTC в секундах от 01/01/1970); *tmu* — микросекунды времени сообщения; *lv* — уровень сообщения; *cat* — категория сообщения.	Текст сообщения

Архивный файл на основе плоского текста состоит из:

- заголовка в формате: [FSArch <vers> <charset> <beg_tm> <end_tm>]
  Где:
  - <*vers*> — версия модуля архивирования;
  - <*charset*> — кодировка файла (обычно UTF8);
  - <*beg_tm*> — UTC время начала архива с эпохи 01.01.1970 в шестнадцатеричной форме;
  - <*end_tm*> — UTC время конца файла архива с эпохи 01.01.1970 в шестнадцатеричной форме.
- записей сообщений в формате: [<tm> <lev> <cat> <mess>]
  Где:
  - **<*tm*> —** время сообщения в виде: **<utc_sec:usec>**, где:
    - *utc_sec* — UTC время с эпохи 01.01.1970 в шестнадцатеричной форме;
    - *usec* — микросекунды времени в десятичной форме.
  -
  - <*lev*> — уровень важности сообщения;
  - <*cat*> — категория сообщения;
  - <*mess*> — текст сообщения.

Текст сообщения и категория кодируются с целью исключения символов разделителей (символ пробела).

1.2. Пример файла архива сообщения

Пример содержимого архивного файла в формате языка XML:

```
<?xml version='1.0' encoding='UTF-8' ?>
<FSArch Version="1.3.0" Begin="4a27dfbc" End="4a28c990">
<m tm="4a28c982" tmu="905587" lv="4"
cat="/DemoStation/sub_DAQ/mod_DiamondBoards/">Ошибка dscInit.</m>
<m tm="4a28c990" tmu="595549" lv="4"
cat="/DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/">Ошибка подключения к
Internet сокету: Операция выполняется в данный момент!</m>
</FSArch>
```

Пример содержимого архивного файла в формате плоского текста:

```
FSArch 1.2.0 UTF-8 4a27dfbb 4a28c991
4a28c98f:432619 1 /DemoStation/ Запуск!
4a28c98f:432858 1 /DemoStation/sub_Transport/ Пуск%20подсистемы.
4a28c98f:455400 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Включение%20контроллера!
4a28c98f:457360 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testTCP/ Включение
   %20контроллера!
4a28c98f:460691 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testRTU/ Включение
   %20контроллера!
4a28c98f:464227 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Включение
   %20контроллера!
4a28c98f:680767 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102cntr/ Включение
   %20контроллера!
4a28c98f:705683 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/
   Включение%20контроллера!
4a28c98f:753659 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/ Включение
   %20контроллера!
4a28c98f:905073 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/ Включение
   %20контроллера!
4a28c990:81670 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM201/ Включение
   %20контроллера!
4a28c990:206208 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM202/ Включение
   %20контроллера!
4a28c990:333471 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/ Включение
   %20контроллера!
4a28c990:457490 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/ Включение
   %20контроллера!
4a28c990:591702 1 /DemoStation/sub_DAQ/mod_System/cntr_AutoDA/ Включение
   %20контроллера!
4a28c990:595549 4 /DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/ Ошибка
   %20подключения%20к%20Internet%20сокету:%20Операция%20выполняется%20в%20данный
   %20момент!
4a28c990:618617 1 /DemoStation/sub_DAQ/mod_SoundCard/cntr_test/ Включение
   %20контроллера!
4a28c990:621487 1 /DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/ Включение
   %20контроллера!
4a28c990:729323 1 /DemoStation/sub_DAQ/mod_JavaLikeCalc/cntr_testCalc/ Включение
   %20контроллера!
4a28c990:733434 1 /DemoStation/sub_DAQ/mod_Siemens/cntr_test/ Включение%20контроллера!
4a28c990:754368 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Включение%20контроллера!
4a28c990:786925 1 /DemoStation/sub_Archive/ Пуск%20подсистемы.
4a28c990:955967 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Запуск
   %20контроллера!
```

2. Архиватор значений

Архивы значений формируются архиваторами значений индивидуально для каждого зарегистрированного архива. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделить архивы по различным параметрам, например, по точности и глубине.

Архив значений является независимым компонентом, который включает буфер, обрабатываемый архиваторами. Основным параметром архива значения является источник данных. В роли источника данных могут выступать атрибуты параметров подсистемы «Сбор данных», а также другие внешние источники данных (пассивный режим). Другими источниками данных могут быть: сетевые архиваторы удалённых OpenSCADA систем, среда программирования системы OpenSCADA и др. Не менее важными параметрами архива являются параметры его буфера. От параметров буфера зависит возможность работы архиваторов. Так, периодичность значений в буфере должна быть не больше периодичности самого быстрого архиватора, а размер буфера не менее двойного размера для самого медленного архиватора. В противном случае возможны потери данных!

Общая схема архивирования значений наглядно изображена на рис. 2.

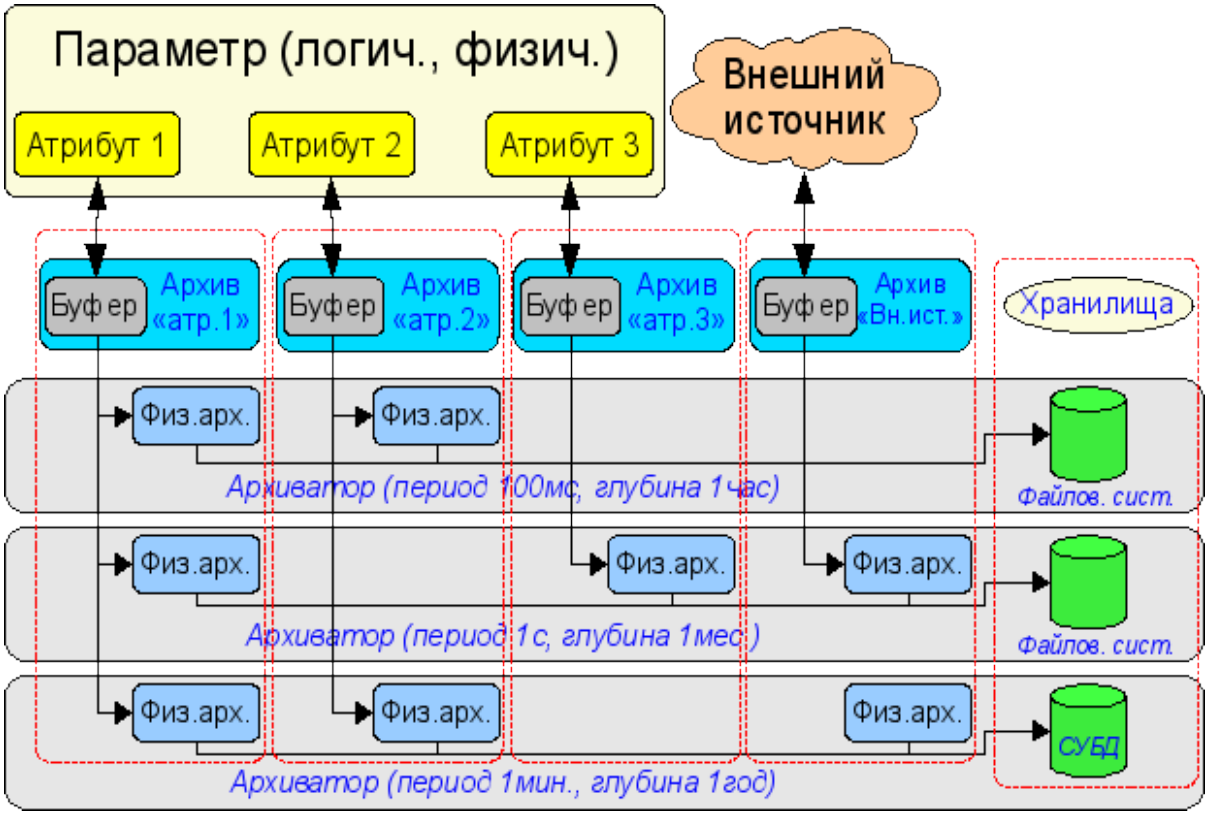


Рис.2. Общая схема процесса архивирования значений модуля FSArch.

Файлы архивов именуются архиваторами, исходя из даты первого значения в архиве и идентификатора архива. Например, таким образом: <MemInfo_use 2006–06–17 17:32:56.val>.

Файлы архивов могут ограничиваться по времени. После превышения лимита создаётся новый файл. Максимальное количество файлов в директории архиватора также может ограничиваться. После превышения лимита на количество файлов старые файлы начнут удаляться!

С целью экономии дискового пространства архиваторы поддерживают упаковку в дополнении к последовательной упаковке старых архивов упаковщиком gzip. Упаковка производится после продолжительного неиспользования архива. Для обеспечения возможности быстрого подключения больших архивов к другой системе можно включить использование информационных файлов для упакованных файлов, что предотвратит предварительную распаковку всех файлов на другой системе.

Модулем предоставляются дополнительные параметры настройки процесса архивирования (рис.3).

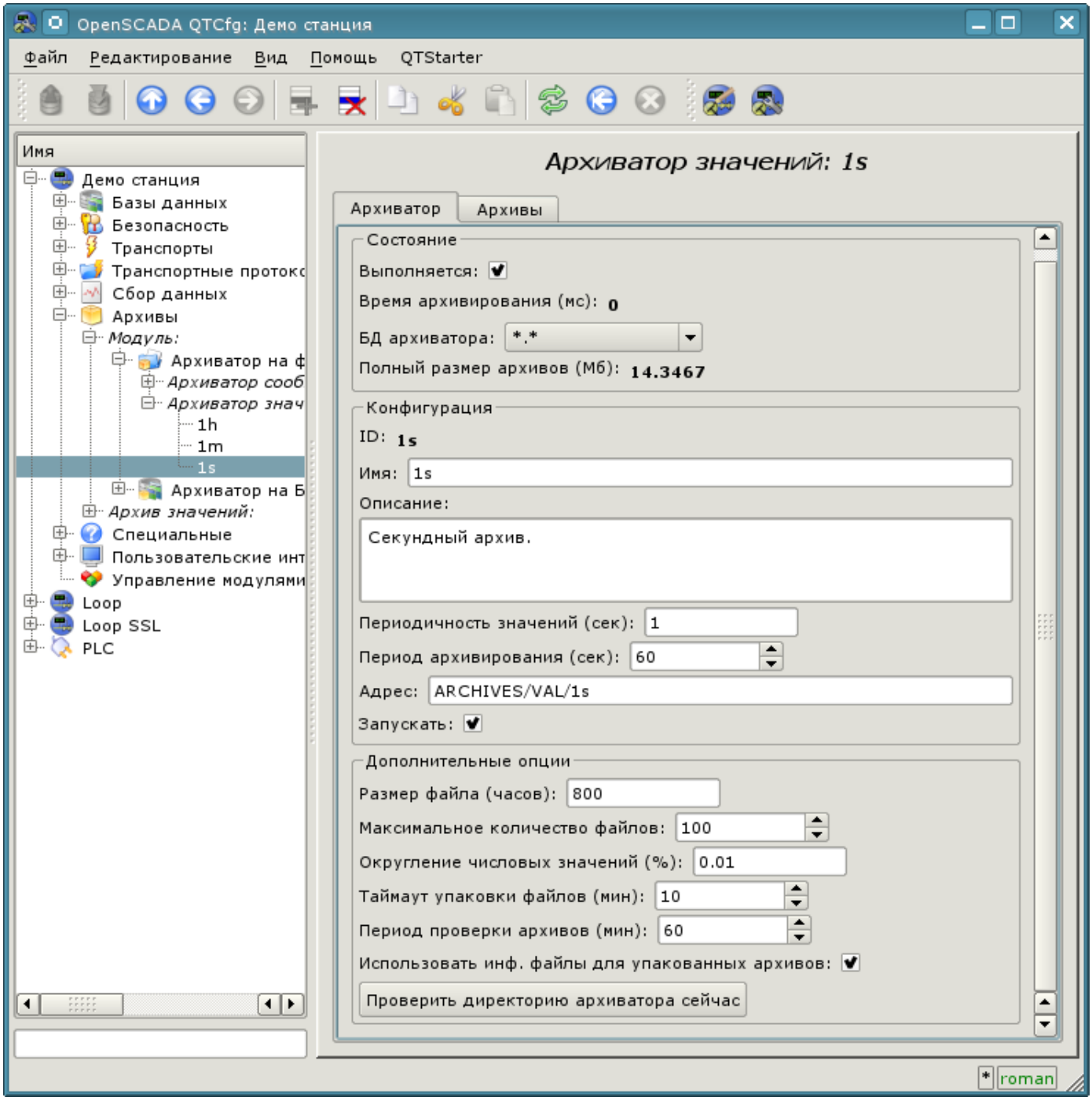


Рис.3. Дополнительные параметры настройки процесса архивирования значений модуля FSArch.

2.1. Формат файлов архива значений

Для реализации архивирования на файловую систему предъявлялись следующие требования:

- быстрый (простой) доступ на добавление в архив и чтение из архива;
- возможность изменения значений в существующем архиве (с целью заполнения дыр в дублированных системах);
- цикличность (ограничение размера);
- возможность сжатия методом упаковки последовательности одинаковых значений, сохраняющим возможность быстрого доступа (последовательная упаковка);
- возможность упаковки устаревших данных стандартными архиваторами (gzip, bzip2 ...) с возможностью распаковки при обращении.

В соответствии с вышеизложенными требованиями организовано архивирование методом множественности файлов (для каждого источника). Цикличность архива реализуется на уровне файлов, т.е. создается новый файл, а самый старый удаляется. Для быстрого сжатия используется метод притягивания к последнему одинаковому значению. Для этих целей в файле архива предусматривается битовая таблица упаковки размером один в один с количеством хранимых данных. Т.е. каждый бит соответствует одному значению в архиве. Значение бита указывает на наличие значения. Для потока одинаковых значений биты обнулены. В случае с архивом строк таблица является не битовой а байтовой и содержит длину соответствующего значения. В случае поступления потока одинаковых значений, длина будет нулевой и читаться будет первое одинаковое значение. Поскольку таблица байтовая, то архив сможет хранить строки длиной не более 255 символов. Таким образом, методики хранения можно разделить на методику данных фиксированного и нефиксированного размера. Общая структура файла архива приведена на рис.4.

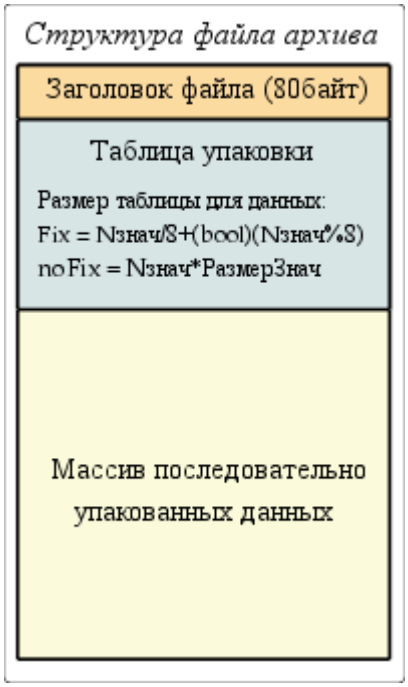


Рис. 4. Общая структура файла архива значений.

При создании нового файла архива формируется: заголовок (структура заголовка в таблице 1), нулевая битовая таблица упаковки архива и первое недостоверное значение. Таким образом, получится архив, инициализированный недостоверными значениями. В дальнейшем новые значения будут вставляться в область значений с корректировкой индексной таблицы упаковки. Из этого следует, что пассивные архивы будут вырождаться в файлы размером в заголовок и битовую таблицу.

Таблица 1. Структура заголовка файла архива

Поле	Описание	Размер байт (бит)
f_tp	Системное имя архива («OpenSCADA Val Arch.»)	20
archive	Имя архива, которому принадлежит файл.	20
beg	Время начала архивных данных (мкс)	8
end	Время конца архивных данных (мкс)	8
period	Периодичность архива (мкс)	8
vtp	Тип значения в архиве (Логический, Целый, Вещественный, Строка)	(3)
hgrid	Признак использования жёсткой сетки в буфере архива	(1)
hres	Признак использования времени высокого разрешения (мкс) в буфере архива	(1)
reserve	Резерв	14
term	Символ окончания заголовка архива (0x55)	1

Разъяснение механизма последовательной упаковки приведено на рис. 5. Как можно видеть из рисунка признак упаковки содержит длину (не фиксированные типы) или признак упаковки (фиксированные типы) отдельно взятого значения. Это значит, что для получения смещения нужного значения необходимо сложить длины всех предыдущих действительных значений. Выполнение данной операции каждый раз и для каждого значения является крайне накладной операцией. Поэтому был внедрён механизм кеширования смещений значений. Механизм кеширует смещения значений через предопределённое их количество, а также кеширует смещение последнего значения к которому производился доступ (отдельно на чтение и запись).

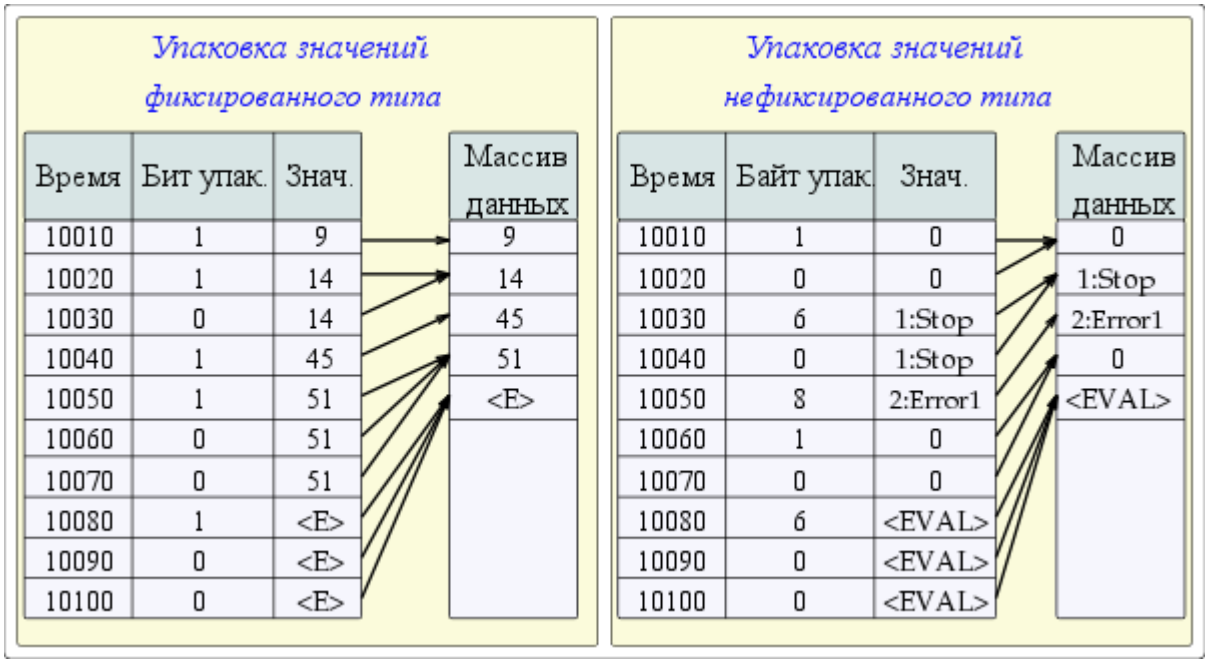


Рис. 5. Механизм последовательной упаковки значений.

Изменения значений внутри существующего архива также предусмотрено. Однако, учитывая необходимость выполнения сдвига хвоста архива, рекомендуется выполнять эту операцию как можно реже и как можно большими блоками.

3. Эффективность

При проектировании и реализации данного модуля были заложены механизмы повышения эффективности процесса архивирования.

Первым механизмом является механизм блочного(покадрового или транзактивного) помещения данных в файлы архивов значений. Такой механизм позволяет достичь максимальной скорости архивирования, а следовательно и позволяет одновременно архивировать больше потоков данных. Опыт практического применение показал, что система K8–3000 с обычным IDE жестким диском способна архивировать до 300000 потоков данных с периодичностью 1 секунда или, система K5–400 с IDE диском (2.5 дюйма) способна архивировать до 100 параметров с периодичностью 1 миллисекунда.

Вторым механизмом является упаковка как текущих значений, так и устаревших файлов архивов для оптимизации используемого дискового пространства. Реализовано два механизма упаковки: механизм последовательной упаковки (архивы значений) и механизм дожатия архивов стандартным упаковщиком (gzip). Данный подход позволил достичь высокой производительности в процессе архивирования текущих данных с эффективным механизмом последовательного сжатия. А дожатие стандартным упаковщиком устаревших архивов завершает общую картину компактного хранения больших массивов данных. Статистика практического применения в условиях реального зашумленного сигнала(худшая ситуация) показала, что степень последовательной упаковки составила 10%, а степень полной упаковки составила 71%.

Модуль подсистемы “Архивы” <DBArch>

Модуль:	DBArch
Имя:	Архиватор на БД
Тип:	Архив
Источник:	arh_DBArch.so
Версия:	0.9.2
Автор:	Роман Савоченко
Описание:	Модуль архива. Предоставляет функции архивирования сообщений и значений на БД.
Лицензия:	GPL

Модуль предназначен для архивирования сообщений и значений системы OpenSCADA на одну из баз данных, поддерживаемых OpenSCADA.

Любая SCADA система предоставляет возможность архивирования собранных данных, т.е. формирование истории изменения (динамики) процессов. Архивы условно можно разделить на два типа: архивы сообщений и архивы значений.

Особенностью архивов сообщений является то, что архивируются, так называемые, события. Характерным признаком события является его время возникновения. Архивы сообщений обычно используются для архивирования сообщений в системе, т.е. ведение логов и протоколов. В зависимости от источника сообщения могут классифицироваться по различным критериям. Например, это могут быть протоколы аварийных ситуаций, протоколы действий операторов, протоколы сбоев связи и др.

Особенностью архивов значений является их периодичность, определяемая промежутком времени между двумя смежными значениями. Архивы значений применяются для архивирования истории непрерывных процессов. Поскольку процесс непрерывный, то и архивировать его можно только путём введения понятия квантования времени опроса, поскольку иначе мы получаем архивы бесконечных размеров ввиду непрерывности самой природы процесса. Кроме этого, практически мы можем получать значения с периодом ограниченным самими источниками данных. Например, довольно качественные источники данных в промышленности редко позволяют получать данные с частотой более 1кГц. И это без учёта самих датчиков, имеющих ещё менее качественные характеристики.

Для ведения архивов в системе OpenSCADA предусмотрена подсистема «Архивы». Данная подсистема, в соответствии с типами архивов, состоит из двух частей: архив сообщений и архивы значений. Подсистема в целом является модульной, что позволяет создавать архивы основанные на различной природе и способах хранения данных. Данный модуль предоставляет механизм архивирования на БД как для потока сообщений, так и для потока значений.

1. Архиватор сообщений

Архивы сообщений формируются архиваторами. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделять архивирование различных классов сообщений.

Архиватор сообщений этого модуля хранит данные в таблице БД, которая именуется таким образом: DBAMsg_{ArchID}. Где:

- *ArchID* — идентификатор архиватора сообщений.

Размер таблицы архива может ограничиваться по времени. После превышения лимита старые записи начнут удаляться!

Модулем предоставляются дополнительные параметры настройки процесса архивирования. У данного модуля таких параметров всего один, и он определяет размер архива по времени.

Таблица БД архиватора сообщений имеет следующую структуру: **{TM, TMU, CATEG, MESS, LEV}**. Где:

- *TM* — UTC время сообщения, секунды от эпохи (01.01.1970). В БД, содержащих специализированный тип для хранения даты и времени, может использоваться этот специализированный тип.
- *TMU* — микросекунды времени.
- *CATEG* — категория сообщения.
- *MESS* — текст сообщения.
- *LEV* — уровень сообщения.

2. Архиватор значений

Архивы значений формируются архиваторами значений индивидуально для каждого зарегистрированного архива. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделить архивы по различным параметрам, например по точности и глубине.

Архив значений является независимым компонентом, который включает буфер, обрабатываемый архиваторами. Основным параметром архива значения является источник данных. В роли источника данных могут выступать атрибуты параметров системы OpenSCADA, а также другие внешние источники данных (пассивный режим). Другими источниками данных могут быть: сетевые архиваторы удалённых OpenSCADA систем, среда программирования системы OpenSCADA и др. Не менее важными параметрами архива являются параметры его буфера. От параметров буфера зависит возможность работы архиваторов. Так, периодичность значений в буфере должна быть не больше периодичности самого быстрого архиватора, а размер буфера не менее двойного размера для самого медленного архиватора. В противном случае возможны потери данных!

Общая схема архивирования значений наглядно изображена на рис. 1.

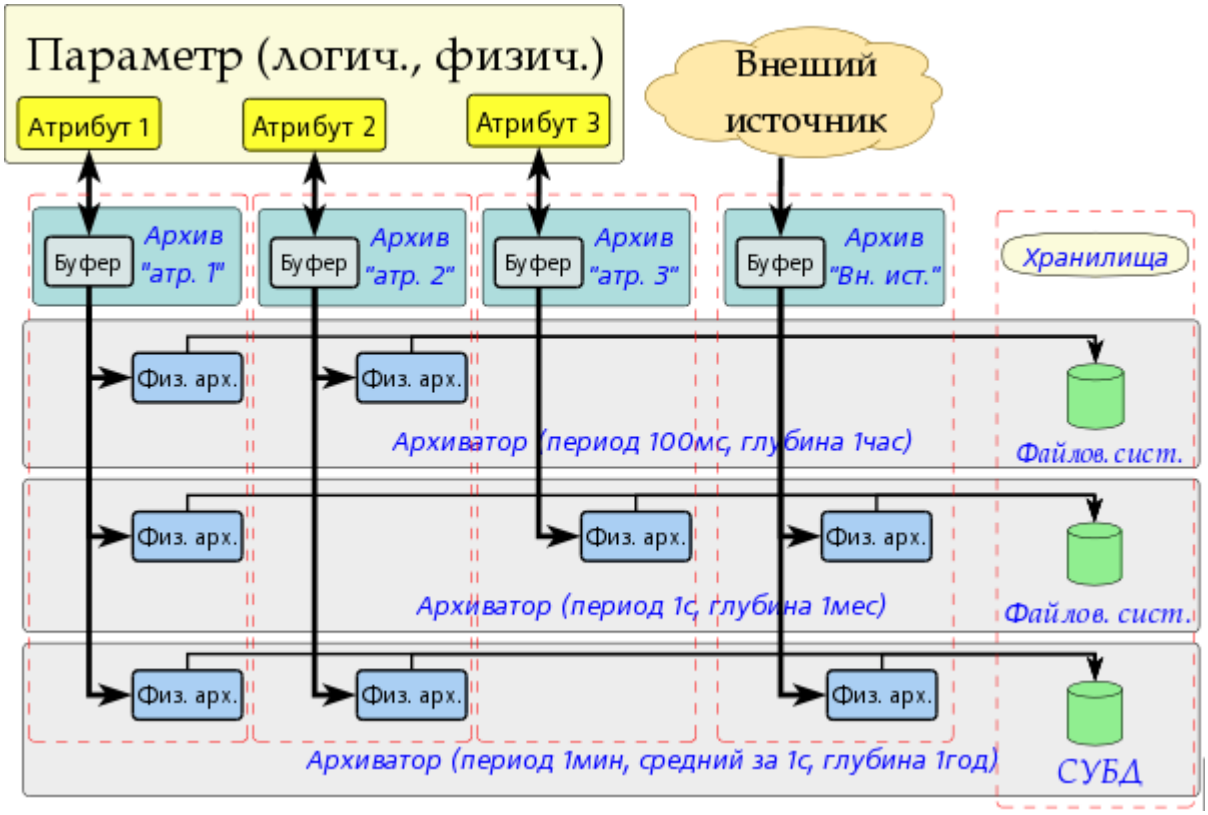


Рис.1. Общая схема процесса архивирования значений модуля FSArch.

Архиватор значений этого модуля хранит данные в таблице БД, которая именуется таким образом: DBAVl_{ArchID}_{ArchiveID}. Где:

- *ArchID* — Идентификатор архиватора значений.
- *ArchiveID* — Идентификатор архива значений.

Размер таблицы архива может ограничиваться по времени. После превышения лимита старые записи начнут удаляться!

Модулем предоставляются дополнительные параметры настройки процесса архивирования. У данного модуля таких параметров всего один, и он определяет размер архива по времени.

Таблица БД архиватора значений имеет следующую структуру: **{TM, TMU, VAL}**. Где:

- *TM* — UTC время значения, секунды от эпохи (01.01.1970). В БД, содержащих специализированный тип для хранения даты и времени, может использоваться этот специализированный тип.
- *TMU* — Время значения, микросекунды.
- *VAL* — Значение, тип значения определяет тип данной колонки.

3. Информационная таблица архивных таблиц

Для хранения начала, конца и иной информации архивов в архивных таблицах создаётся информационная таблица с именем данного модуля: «DBArch". Данная таблица имеет структуру: **{TBL, BEGIN, END, PRM1, PRM2, PRM3}**. Где:

- *TBL* — Имя таблицы архива.
- *BEGIN* — Начало данных в архиве.
- *END* — Конец данных в архиве.
- *PRM1* — Дополнительный параметр 1.
- *PRM2* — Дополнительный параметр 2.
- *PRM3* — Дополнительный параметр 3.

Модуль подсистемы “БД” <DBF>

Модуль:	DBF
Имя:	БД DBF
Тип:	БД
Источник:	bd_DBF.so
Версия:	2.0.2
Автор:	Роман Савоченко
Описание:	Модуль БД. Предоставляет поддержку *.dbf файлов, версии 3.0.
Лицензия:	GPL

Модуль предназначен для предоставление в систему OpenSCADA поддержки файлов БД типа *.dbf. Модуль основан на библиотеке для работы с dbf файлами ПО “Complex2” фирмы НИП “ДІЯ”. Модуль позволяет выполнять действия над базами данных, таблицами и содержимым таблиц.

1. Операции над БД

Поддерживаются операции открытия и закрытия БД с возможностью создания новой БД при открытии и удаления существующей при закрытии. В терминах подсистемы «БД» системы OpenSCADA открытием БД является её регистрация для последующего использования в системе.

Под БД в случае с dbf-файлами подразумевается директория, содержащая dbf-файлы. Следовательно, операции создания и удаления БД – создают и удаляют директории, где таблицы (dbf-файлы) хранятся. В роли адреса БД выступает полное имя директории с dbf-файлами. Доступ к БД определяется системными правами доступа к директории.

Модуль поддерживает кодирование данных в нужную кодировку. С этой целью для БД в целом можно указать рабочую кодировку. В процессе работы будет выполняться кодирование данных базы данных из кодировки БД в системную кодировку OpenSCADA и обратно.

2. Операции над таблицей

Поддерживаются операции открытия и закрытия таблицы с возможностью создания новой таблицы при открытии и удаления существующей при закрытии.

Собственно dbf-файл и является таблицей. Создание и удаления таблицы подразумевает создание и удаление dbf-файла. Имя таблицы представляет собой имя dbf-файла в директории БД. Права доступа к таблице определяются правами доступа к dbf-файлу.

3. Операции над содержимым таблицы

- сканирование записей таблицы;
- запрос значений указанных записей;
- установка значений указанных записей;
- удаление записей.

API подсистемы “БД” предполагает доступ к содержимому таблицы по значению ключевого(ых) поля(ей). Так, операция запроса записи подразумевает предварительную установку ключевых колонок объекта TConfig, по которым и будет выполнен запрос. Создание новой записи(строки) производится операцией установки значений записи, которая отсутствует.

Модуль позволяет динамически менять структуру таблиц БД DBF. Так, в случае несоответствия структуры таблицы и структуры устанавливаемой записи структура таблицы будет приведена к требуемой структуре записи. В случае запроса значений записи и не соответствия структур записи и

таблицы будут получены только значения общих элементов записи и таблицы. Модуль не отслеживает порядка расположения элементов в записи и структуре таблицы.

При доступе к значениям таблиц используется синхронизация путём захвата ресурса на доступ к таблице. Это позволяет избежать разрушения данных в случае многопоточного доступа!

Типы элементов dbf-файлов следующим образом соответствуют типам элементов системы OpenSCADA:

Типы полей системы OpenSCADA	**Тип поля dbf-файла**
TFld::String	“C”
TFld::Integer, TFld::Real	“N”
TFld::Boolean	“L”

4. Производительность БД

Замер производительности БД выполнялся тестом «БД», модуля системных тестов «SystemTests" путём выполнения операций над записями структурой: <name char(20), descr char(50), val double(10.2), id int(7), stat bool>

Операция	**K8–3000+, 256M, 120G**
Создание 1000 записей (сек):	1.07
Обновление 1000 записей (сек):	1.6
Получение 1000 записей (сек):	1.0
Удаление 1000 записей (сек):	0.95

Модуль подсистемы “БД” <MySQL>

Модуль:	MySQL
Имя:	БД MySQL
Тип:	БД
Источник:	bd_MySQL.so
Версия:	1.6.2
Автор:	Роман Савоченко
Описание:	Модуль БД. Предоставляет поддержку БД MySQL.
Лицензия:	GPL

Модуль <MySQL> предоставляет в систему OpenSCADA поддержку БД MySQL. БД MySQL является мощной реляционной и многоплатформенной БД доступной по свободной лицензии. Разработчиком БД MySQL является фирма MySQL AB http://www.mysql.com. Модуль основан на библиотеке C API производителя БД MySQL. Модуль позволяет выполнять действия над базами данных, таблицами и содержимым таблиц.

1. Операции над БД

Поддерживаются операции открытия и закрытия БД с возможностью создания новой БД при открытии и удаления существующей при закрытии. В терминах подсистемы «БД» системы OpenSCADA открытием БД является её регистрация для последующего использования в системе. Также поддерживается операция запроса списка таблиц в БД.

БД MySQL адресуется строкой следующего типа:
[<host>;<user>;<pass>;<bd>;<port>;<u_sock>;<names>]. Где:

- *host* — имя хоста на котором работает сервер БД MySQL;
- *user* — имя пользователя БД;
- *pass* — пароль пользователя для доступа к БД;
- *bd* — имя БД;
- *port* — порт, который слушает сервер БД (по умолчанию 3306);
- *u_sock* — имя UNIX-сокета в случае локального доступа к БД (/var/lib/mysql/mysql.sock);
- names — My SQL кодировка передачи, устанавливаемая SET NAMES.

В случае локального доступа к БД в пределах одного хоста нужно использовать UNIX сокет. Например: *[;roman;123456;OpenSCADA;;/var/lib/mysql/mysql.sock]*

В случае удалённого доступа к БД нужно использовать имя хоста и порт сервера БД. Например: *[server.nm.org;roman;123456;OpenSCADA;3306]*

2. Операции над таблицей

Поддерживаются операции открытия, закрытия таблицы с возможностью создания новой таблицы при открытии и удаления существующей при закрытии, а также запрос структуры таблицы.

3. Операции над содержимым таблицы

- сканирование записей таблицы;
- запрос значений указанных записей;
- установка значения указанных записей;
- удаление записей.

API подсистемы “БД” предполагает доступ к содержимому таблицы по значению ключевого(ых)

поля(ей). Так, операция запроса записи подразумевает предварительную установку ключевых колонок объекта TConfig, по которым будет выполнен запрос. Создание новой записи(строки) производится операцией установки значений записи, которая отсутствует.

Модуль позволяет динамически изменять структуру таблиц БД MySQL. Так, в случае несоответствия структуры таблицы и структуры устанавливаемой записи, структура таблицы будет приведена к требуемой структуре записи. В случае запроса значений записи и не соответствия структур записи и таблицы, будут получены только значения общих элементов записи и таблицы. Модуль не отслеживает порядок расположения элементов в записи и структуре таблицы.

Модулем реализуется механизм поддержки многоязыковых текстовых переменных. Для полей с многоязыковой текстовой переменной создаются колонки отдельных языков в формате **<lang>#<FldID>** (en#NAME). При этом базовая колонка содержит значение для базового языка. Колонки отдельных языков создаются по надобности, в момент сохранения в БД и при исполнении OpenSCADA в соответствующей локали. В случае отсутствия значения для конкретного языка будет использоваться значений для базового языка.

Типы элементов БД MySQL следующим образом соответствуют типам элементов системы OpenSCADA:

Типы полей системы OpenSCADA	Типы полей БД MySQL
TFld::String	char(n), text, mediumtext
TFld::Integer	int(n), DATETIME [для полей с флагом TFld::DateTimeDec]
TFld::Real	double(n,m)
TFld::Boolean	tinyint(1)

4. Права доступа

БД MySQL содержит мощный механизм разделения доступа, который заключается в выборочном указании доступа пользователя БД к отдельным SQL-командам. В таблице ниже перечислены операции над БД и требуемый доступ к командам для этих операций.

Операция	SQL-команды
Создание БД и таблиц	CREATE
Удаление БД и таблиц	DROP
Добавление записей	INSERT
Удаление записей	DELETE
Получение значений записей	SELECT
Установка значений записей	UPDATE
Манипуляция структурой таблицы	ALTER

5. Производительность БД

Замер производительности БД выполнялся тестом «БД» модуля системных тестов "SystemTests" путём выполнения операций над записями структурой: <name char(20), descr char(50), val double(10.2), id int(7), stat bool>.

Операция	K8–3000+, 384M, 120G, MySQL 5.0.51 (local)	MySQL 4.0.24 (remote)
Создание 1000 записей (сек):	0.67	0.99
Обновление 1000 записей (сек):	0.67	1.33
Получение 1000 записей (сек):	0.38	0.49
Удаление 1000 записей (сек):	0.23	0.34

Модуль подсистемы “БД” <SQLite>

Модуль:	SQLite
Имя:	БД SQLite
Тип:	БД
Источник:	bd_SQLite.so
Версия:	1.6.2
Автор:	Роман Савоченко
Описание:	Модуль БД. Предоставляет поддержку БД SQLite.
Лицензия:	GPL

Модуль <SQLite> предоставляет в систему OpenSCADA поддержку БД SQLite. БД SQLite является небольшой, встраиваемой БД поддерживающей SQL-запросы. БД SQLite распространяется по свободной лицензии. Ознакомиться с БД можно на сайте БД – http://sqlite.org. Модуль основан на библиотеке C/C++ API производителя БД. Модуль позволяет выполнять действия над базами данных, таблицами и содержимым таблиц.

1. Операции над БД

Поддерживаются операции открытия и закрытия БД, с возможностью создания новой БД при открытии и удаления существующей при закрытии. В терминах подсистемы «БД», системы OpenSCADA, открытием БД является её регистрация для последующего использования в системе. Также, поддерживается операция запроса перечня таблиц в БД.

БД SQLite адресуется путём указания имени файла БД в формате: *[<FileDBPath>]*. Где:

- *FileDBPath* - полный путь к файлу БД (./oscada/Main.db).
  Используйте пустой путь для создания временной базы данных на диске.
  Используйте ":memory:" для создания временной базы данных в памяти.

Модуль поддерживает кодирование данных в нужную кодировку. С этой целью, для БД в целом, можно указать рабочую кодировку. В процессе работы будет выполняться кодирование данных, базы данных, из кодировки БД в системную кодировку OpenSCADA и обратно.

2. Операции над таблицей

Поддерживаются операции открытия, закрытия таблицы, с возможностью создания новой таблицы при открытии и удаления существующей при закрытии, а также запрос структуры таблицы.

3. Операции над содержимым таблицы

- сканирование записей таблицы;
- запрос значений указанных записей;
- установка значений указанных записей;
- удаление записей.

API подсистемы “БД” предполагает доступ к содержимому таблицы по значению ключевого(ых) полей. Так, операция запроса записи подразумевает предварительную установку ключевых колонок объекта TConfig, по которым будет выполнен запрос. Создание новой записи(строки) производится операцией установки значений записи, которая отсутствует.

Модуль позволяет динамически менять структуру таблиц БД SQLite, путём создания новой БД с требуемой структурой и копирование в неё данных из старой. Так, в случае несоответствия структуры таблицы и структуры устанавливаемой записи, структура таблицы будет приведена к требуемой структуре записи. В случае запроса значений записи и не соответствия структур записи и

таблицы, будут получены только значения общих элементов записи и таблицы. Модуль не отслеживает порядка расположения элементов записи и структуры таблицы!

Модулем реализуется механизм поддержки многоязыковых текстовых переменных. Для полей с многоязыковой текстовой переменной создаются колонки отдельных языков в формате **<lang>#<FldID>** (en#NAME). При этом базовая колонка содержит значение для базового языка. Колонки отдельных языков создаются по надобности, в момент сохранения в БД и при исполнении OpenSCADA в соответствующей локали. В случае отсутствия значения для конкретного языка будет использоваться значений для базового языка.

Типы элементов БД SQLite следующим образом соответствуют типам элементов системы OpenSCADA:

Типы полей системы OpenSCADA	**Типы полей БД SQLite**
TFld::String	TEXT
TFld::Integer, TFld::Boolean	INTEGER
TFld::Real	DOUBLE

4. Права доступа

Права доступа к БД определяются правами доступа к отдельно взятому файлу БД. Модулем поддерживается работа с файлами БД SQLite в режиме только для чтения, например в демонстрациях.

5. Производительность БД

Замер производительности БД выполнялся тестом «БД», модуля системных тестов "SystemTests", путём выполнения операций над записями структурой: <name char(20), descr char(50), val double(10.2), id int(7), stat bool>.

Операция	**K8–3000+, 256M, 120G, SQLite 3.4.2**
Создание 1000 записей (сек):	0.45
Обновление 1000 записей (сек):	0.50
Получение 1000 записей (сек):	0.2
Удаление 1000 записей (сек):	0.2

Модуль подсистемы “БД” <FireBird>

Модуль:	FireBird
Имя:	БД FireBird
Тип:	БД
Источник:	bd_FireBird.so
Версия:	0.9.5
Автор:	Роман Савоченко
Описание:	Модуль БД. Предоставляет поддержку БД FireBird.
Лицензия:	GPL

Модуль <FireBird> предоставляет в систему OpenSCADA поддержку БД FireBird и InterBase. БД FireBird является небольшой встраиваемой БД с функциями сетевой БД, поддерживающей SQL-запросы. БД FireBird построена на основе коммерческой СУБД Interbase и распространяется по свободной лицензии. Ознакомиться с БД можно на сайте БД – http://www.firebirdsql.org. Модуль основан на библиотеке C/C++ API производителя БД. Модуль позволяет выполнять действия над базами данных, таблицами и содержимым таблиц.

1. Операции над БД

Поддерживаются операции открытия и закрытия БД с возможностью создания новой БД при открытии и удаления существующей при закрытии. В терминах подсистемы «БД» системы OpenSCADA открытием БД является её регистрация для последующего использования в системе. Также поддерживается операция запроса перечня таблиц в БД.

БД FireBird адресуется путём указания имени файла БД, пользователя и пароля. В общем адрес БД записывается таким образом: *[<file>;<user>;<pass>]*. Где:

- *file* — полное имя файла БД;
- *user* — пользователь БД, от имени которого производится доступ;
- *pass* — пароль пользователя, от имени которого производится доступ.

Модуль поддерживает кодирование данных в нужную кодировку. С этой целью для БД в целом можно указать рабочую кодировку. В процессе работы будет выполняться кодирование данных базы данных из кодировки БД в системную кодировку OpenSCADA и обратно.

2. Операции над таблицей

Поддерживаются операции открытия, закрытия таблицы с возможностью создания новой таблицы при открытии и удаления существующей при закрытии, а также запрос структуры таблицы.

3. Операции над содержимым таблицы

- сканирование записей таблицы;
- запрос значений указанных записей;
- установка значений указанных записей;
- удаление записей.

API подсистемы “БД” предполагает доступ к содержимому таблицы по значению ключевого(ых) поля(ей). Так, операция запроса записи подразумевает предварительную установку ключевых колонок объекта TConfig, по которым будет выполнен запрос. Создание новой записи(строки) производится операцией установки значений записи, которая отсутствует.

Модуль позволяет динамически менять структуру таблиц БД FireBird. Так, в случае

несоответствия структуры таблицы и структуры устанавливаемой записи структура таблицы будет приведена к требуемой структуре записи. В случае запроса значений записи и не соответствия структур записи и таблицы, будут получены только значения общих элементов записи и таблицы. Модуль не отслеживает порядка расположения элементов записи и структуры таблицы, кроме этого операция изменения типа колонки не является безопасной и данные в изменяемых колонках будут утеряны!

Модулем реализуется механизм поддержки многоязыковых текстовых переменных. Для полей с многоязыковой текстовой переменной создаются колонки отдельных языков в формате **<lang>#<FldID>** (en#NAME). При этом базовая колонка содержит значение для базового языка. Колонки отдельных языков создаются по надобности, в момент сохранения в БД и при исполнении OpenSCADA в соответствующей локали. В случае отсутствия значения для конкретного языка будет использоваться значений для базового языка.

Типы элементов БД FireBird следующим образом соответствуют типам элементов системы OpenSCADA:

Типы полей системы OpenSCADA	Типы полей БД Fire Bird
TFld::String	VARCHAR, BLOB SUBTYPE TEXT
TFld::Integer	INTEGER
TFld::Real	DOUBLE
TFld::Boolean	SMALLINT

4. Права доступа

Права доступа к БД определяются правами БД.

5. Производительность БД

Замер производительности БД выполнялся тестом «БД» модуля системных тестов "Special.SystemTests" путём выполнения операций над записями структурой: <name char(20), descr char(50), val double(10.2), id int(7), stat bool>.

Операция	K8–3000+, 256M, 120G, FireBird 2.0.3 (Local SuperServer)	FireBird 2.0.3 (Remote SuperServer)
Создание 1000 записей (сек):	1.23	2.76
Обновление 1000 записей (сек):	4.43	6.92
Получение 1000 записей (сек):	2.31	4
Удаление 1000 записей (сек):	1.01	2.39

Модуль подсистемы “БД” <PostgreSQL>

Модуль:	PostgreSQL
Имя:	БД PostgreSQL
Тип:	БД
Источник:	bd_PostgreSQL.so
Версия:	0.9.0
Автор:	Лысенко Максим
Описание:	Модуль БД. Предоставляет поддержку БД PostgreSQL.
Лицензия:	GPL

Модуль <PostgreSQL> предоставляет в систему OpenSCADA поддержку БД PostgreSQL. БД PostgreSQL является мощной реляционной и многоплатформенной БД доступной по свободной лицензии. Разработчиком БД PostgreSQL является сообщество PostgreSQL http://www.postgresql.org/. Модуль основан на библиотеке C API производителя БД PostgreSQL. Модуль позволяет выполнять действия над базами данных, таблицами и содержимым таблиц.

1. Операции над БД

Поддерживаются операции открытия и закрытия БД с возможностью создания новой БД при открытии и удаления существующей при закрытии. В терминах подсистемы «БД» системы OpenSCADA открытием БД является её регистрация для последующего использования в системе. Также поддерживается операция запроса списка таблиц в БД.

БД PostgreSQL адресуется строкой следующего типа: [<host>;<hostaddr>;<user>;<pass>;<bd>;<port>;<connect_timeout>]. Где:

- host - Имя хоста для подключения. Если начинается с косой черты, оно указывает Unix-domain соединение вместо TCP/IP соединения, значение - это имя каталога, в котором хранится файл сокета.
- hostaddr - числовой IP адрес хоста для подключения, на котором работает сервер БД PostgreSQL;
- user - имя пользователя БД;
- pass - пароль пользователя для доступа к БД;
- bd - имя БД;
- port - порт, который слушает сервер БД (по умолчанию 5432);
- connect_timeout - таймаут соединения;

В случае локального доступа к БД в пределах одного хоста строка адреса может выглядеть следующим образом:[;roman;123456;OpenSCADA;;10]

В случае удалённого доступа к БД нужно использовать адрес хоста и порт сервера БД. Например: [server.nm.org;192.168.2.1;roman;123456;OpenSCADA;;10]

2. Операции над таблицей

Поддерживаются операции открытия, закрытия таблицы с возможностью создания новой таблицы при открытии и удаления существующей при закрытии, а также запрос структуры таблицы.

3. Операции над содержимым таблицы

- сканирование записей таблицы;
- запрос значений указанных записей;

- установка значения указанных записей;
- удаление записей.

API подсистемы “БД” предполагает доступ к содержимому таблицы по значению ключевого(ых) поля(ей). Так, операция запроса записи подразумевает предварительную установку ключевых колонок объекта TConfig, по которым будет выполнен запрос. Создание новой записи(строки) производится операцией установки значений записи, которая отсутствует.

Модуль позволяет динамически изменять структуру таблиц БД PostgreSQL. Так, в случае несоответствия структуры таблицы и структуры устанавливаемой записи, структура таблицы будет приведена к требуемой структуре записи. В случае запроса значений записи и несоответствия структур записи и таблицы, будут получены только значения общих элементов записи и таблицы. Модуль не отслеживает порядок расположения элементов в записи и структуре таблицы.

Модулем реализуется механизм поддержки многоязыковых текстовых переменных. Для полей с многоязыковой текстовой переменной создаются колонки отдельных языков в формате **<lang>#<FldID>** (en#NAME). При этом базовая колонка содержит значение для базового языка. Колонки отдельных языков создаются по надобности, в момент сохранения в БД и при исполнении OpenSCADA в соответствующей локали. В случае отсутствия значения для конкретного языка будет использоваться значения для базового языка.

Типы элементов БД PostgreSQL следующим образом соответствуют типам элементов системы OpenSCADA:

Типы полей системы OpenSCADA	**Типы полей БД MySQL**
TFld::String	character(n), character varying(n), text
TFld::Integer	integer, bigint, timestamp with time zone [для полей с флагом TFld::DateTimeDec]
TFld::Real	double precision
TFld::Boolean	smallint

4. Права доступа

БД PostgreSQL содержит некоторый механизм разделения доступа, который заключается в указании привилегий пользователя БД. В таблице ниже перечислены необходимые привилегии для полноценной работы.

Операция	**Привилегия**
Создание БД	CREATEDB
Создание соединения	LOGIN

5. Производительность БД

Замер производительности БД выполнялся тестом «БД» модуля системных тестов "SystemTests" путём выполнения операций над записями структурой: <name char(20), descr char(50), val double(10.2), id int(7), stat bool>. OpenSCADA запускалась с демонстрационной конфигурацией.

Операция	**K8–3000+, 384M, 120G, PostgreSQL 8.3 (local)**	**PostgreSQL 8.3 (remote)**
Создание 1000 записей (сек):	0.89	1.04
Обновление 1000 записей (сек):	1.02	1.1
Получение 1000 записей (сек):	0.61	0.63
Удаление 1000 записей (сек):	0.36	0.4

Модуль подсистемы “Сбор данных” <DiamondBoards>

Модуль:	DiamondBoards
Имя:	Diamond платы сбора данных
Тип:	DAQ
Источник:	daq_DiamondBoards.so
Версия:	1.2.1
Автор:	Роман Савоченко
Описание:	Предоставляет доступ к платам сбора данных от Diamond Systems. Включает поддержку системной платы Athena.
Лицензия:	GPL

Модуль предоставляет в систему OpenSCADA поддержку источников динамических данных, основанных на платах сбора данных фирмы Diamond Systems (http://diamondsystems.com/). Модуль построен на основе универсального драйвера производителя плат. Универсальный драйвер доступен практически для всех известных программных платформ в виде библиотеки. Универсальных драйвер был получен по адресу http://www.diamondsystems.com/support/software. Драйвер был включен в дистрибутив системы OpenSCADA, поэтому для сборки данного модуле не требуются внешние библиотеки.

Платы сбора данных фирмы Diamond Systems представляют из себя модули расширения формата PC/104. Платы могут содержать: аналоговые IO(входы/выходы), дискретные IO и счётчики. Комплектация плат может значительно варьироваться. Могут содержаться только IO одного типа или же всё понемногу. Кроме того, функцией сбора данных могут наделяться и системные платы этой фирмы. Например, системная плата Athena содержит: 16 AI, 4 AO, 24 DIO.

Модуль предоставляет поддержку аналоговых и дискретных IO. Сбор аналоговых входов (AI) поддерживается в двух режимах: прямого сбора и сбора по прерыванию. Метод сбора по прерыванию позволяет достичь максимальной частоты опроса поддерживаемой аппаратурой. В случае с процессорной платой Athena эта частота достигает 100 кГц. В процессе сбора по прерыванию данные получаются секундными кадрами и помещаются в буфера архивов значений.

В случае опроса аналоговых каналов по прерыванию настроить индивидуально каждый канал нельзя. Такая возможность предоставляется только при прямом опросе.

Дискретные каналы обычно являются двунаправленными и группируются по 8 каналов. Каждой группе каналов можно отдельно назначить направленность. Модуль предоставляет возможность конфигурировать группы дискретных параметров.

Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

1. Контроллер данных плат фирмы Diamond

Плата фирмы Diamond Systems конфигурируется путем создания контроллера в системе OpenSCADA и его конфигурации. Пример вкладки конфигурации контроллера платы приведен на рис.1.

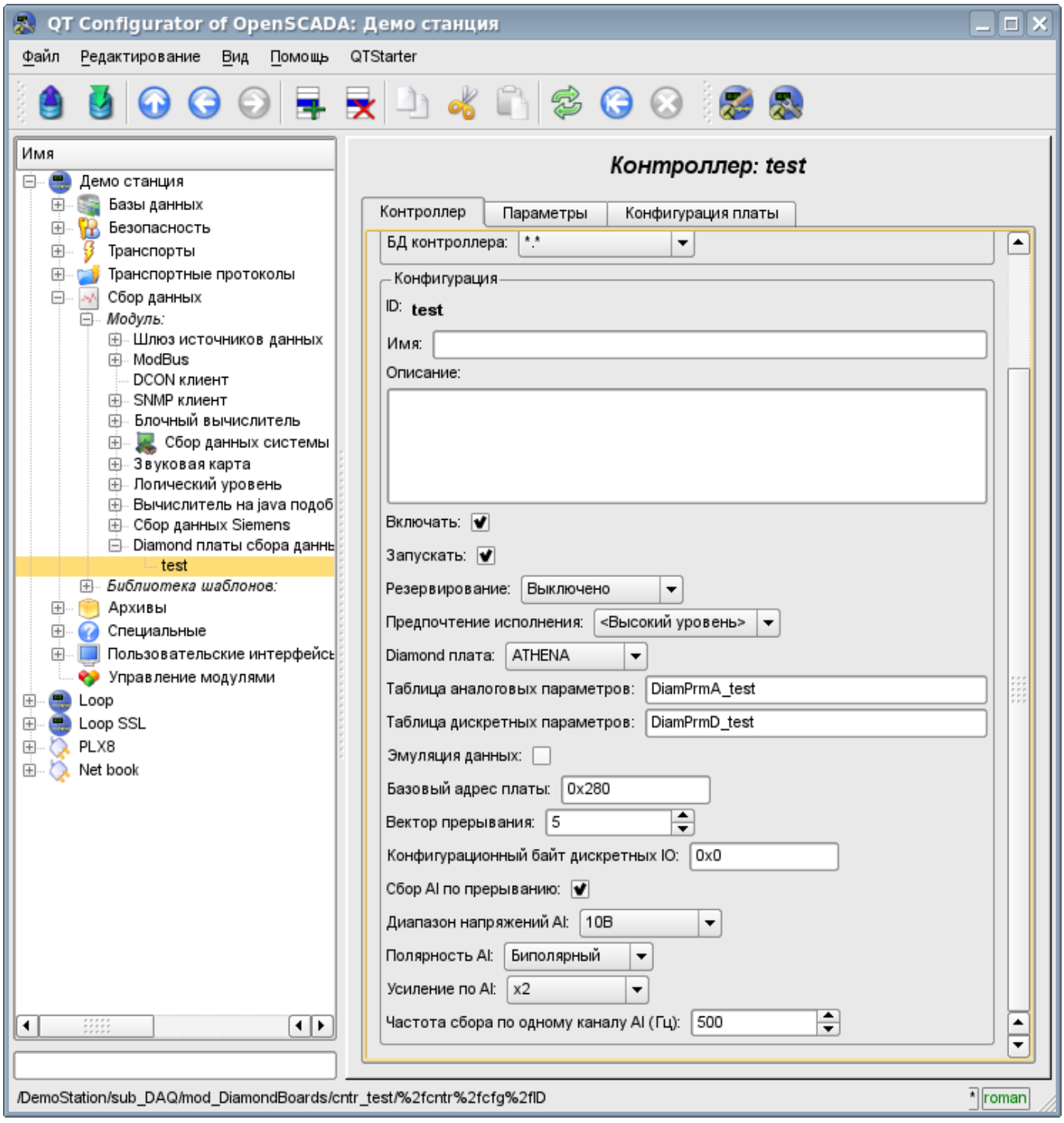


Рис.1. Вкладка конфигурации контроллера/платы фирмы Diamond Systems.

С помощью этой формы можно установить:

- Состояние контроллера(платы), а именно: Статус, «Включен», «Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера(платы).
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Тип платы фирмы Diamond Systems.

- Имена таблиц для хранения конфигурации аналоговых и дискретных параметров данного контроллера.
- Включение режима эмуляции высокоскоростного источника данных.
- Базовый адрес и аппаратное прерывание платы (для сбора по прерыванию).
- Признак сбора аналоговых входов по прерыванию и частоту сбора данных по одному каналу.
- Общую конфигурацию преобразователя аналоговых входов в составе: диапазона входных напряжений, полярности и усиления каналов.

В режиме прямого опроса аналоговых входов аппаратное прерывание платы, частота опроса аналоговых входов и усиление аналогового преобразователя недоступны.

Для конфигурации портов цифровых входов/выходов на странице контроллера содержится вкладка их конфигурации (рис.2).

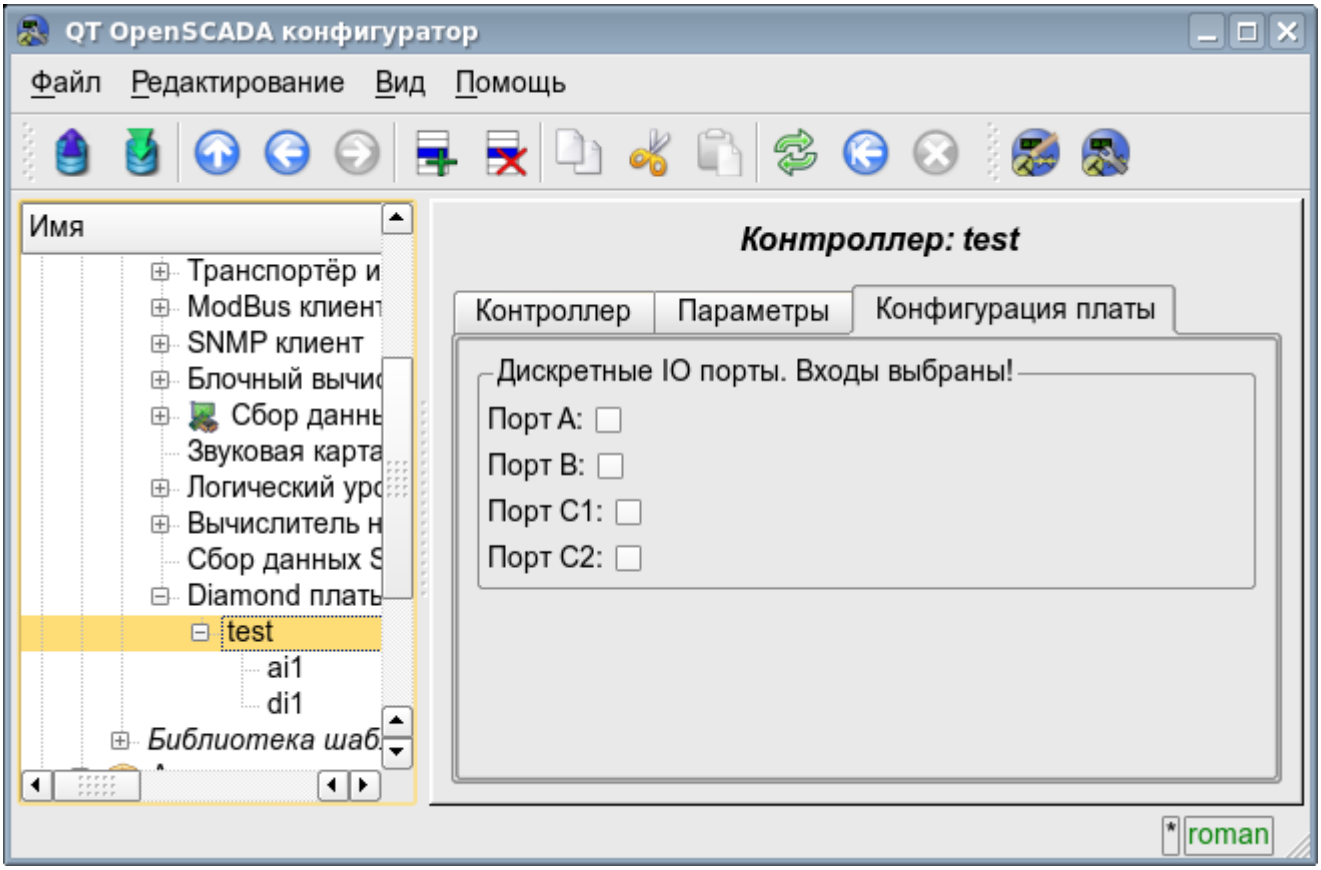


Рис.2. Вкладка конфигурации портов цифровых входов/выходов.

2. Параметры контроллера Diamond

Модулем предоставляется информация о двух типах параметров: цифровом и аналоговом. Каждый тип параметра хранится в отдельной БД и, как следствие, содержит собственную вкладку конфигурации. Вкладка конфигурации аналоговых параметров представлена на рис.3. Вкладка конфигурации цифровых параметров представлена на рис.4.

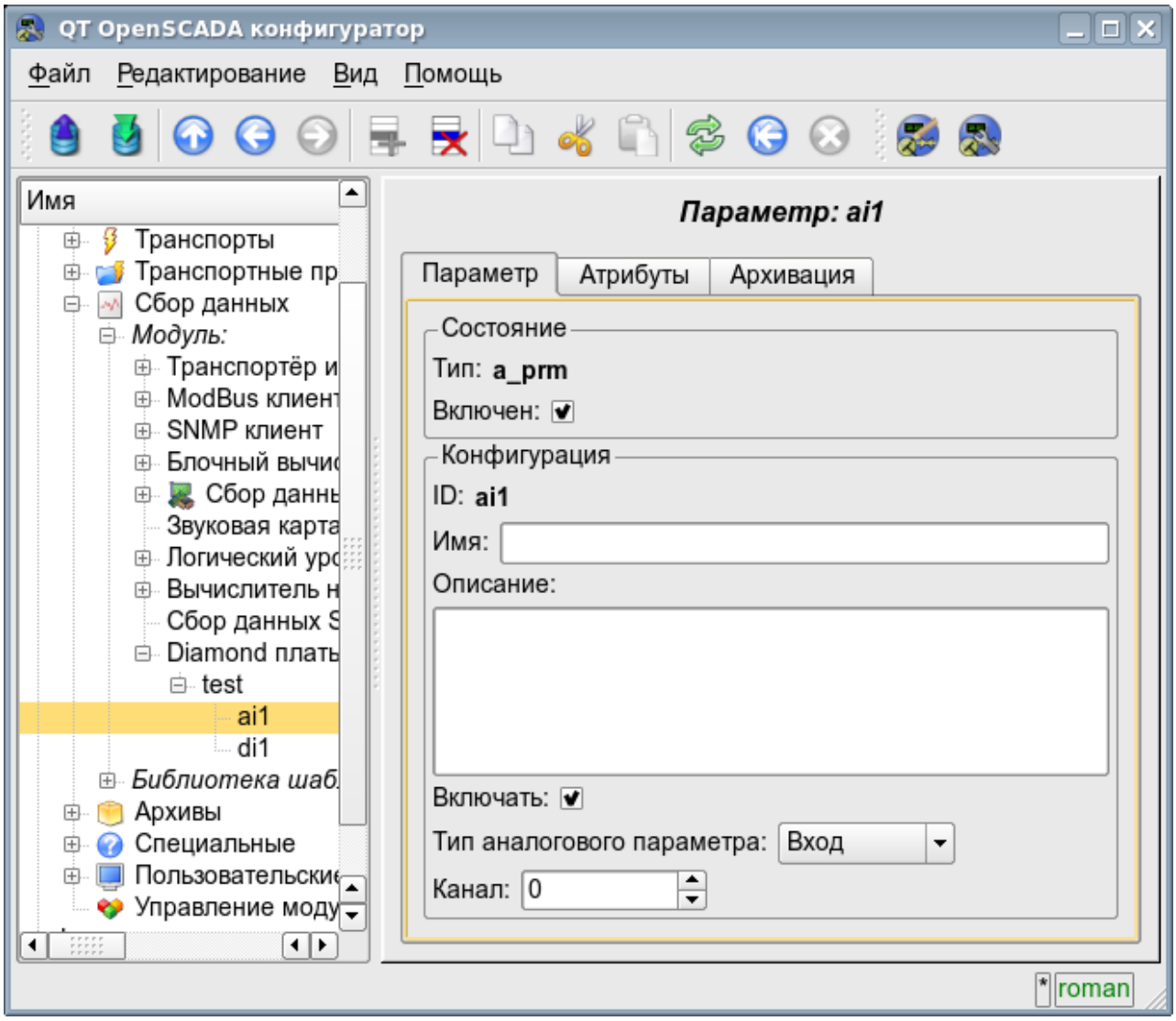


Рис.3. Вкладка конфигурации аналоговых параметров.

С помощью формы конфигурации аналоговых параметров можно установить:

- Режим параметра, а именно «Включен» и тип параметра.
- Идентификатор, имя и описание параметра.
- Состояние, в которое переводить параметр при загрузке: «Включен».
- Направленность параметра – «Вход» или «Выход».
- Физический канал параметра.
- Усиление канала в случае входа (для прямого опроса).

Для доступа к значениям аналоговых параметров формируются атрибуты. Для аналоговых входов:

- значение в процентах (value);
- входное напряжение (voltage);
- код АЦП (code).

Для аналоговых выходов устанавливаются:

- значение в процентах (value);
- выходное напряжение (voltage).

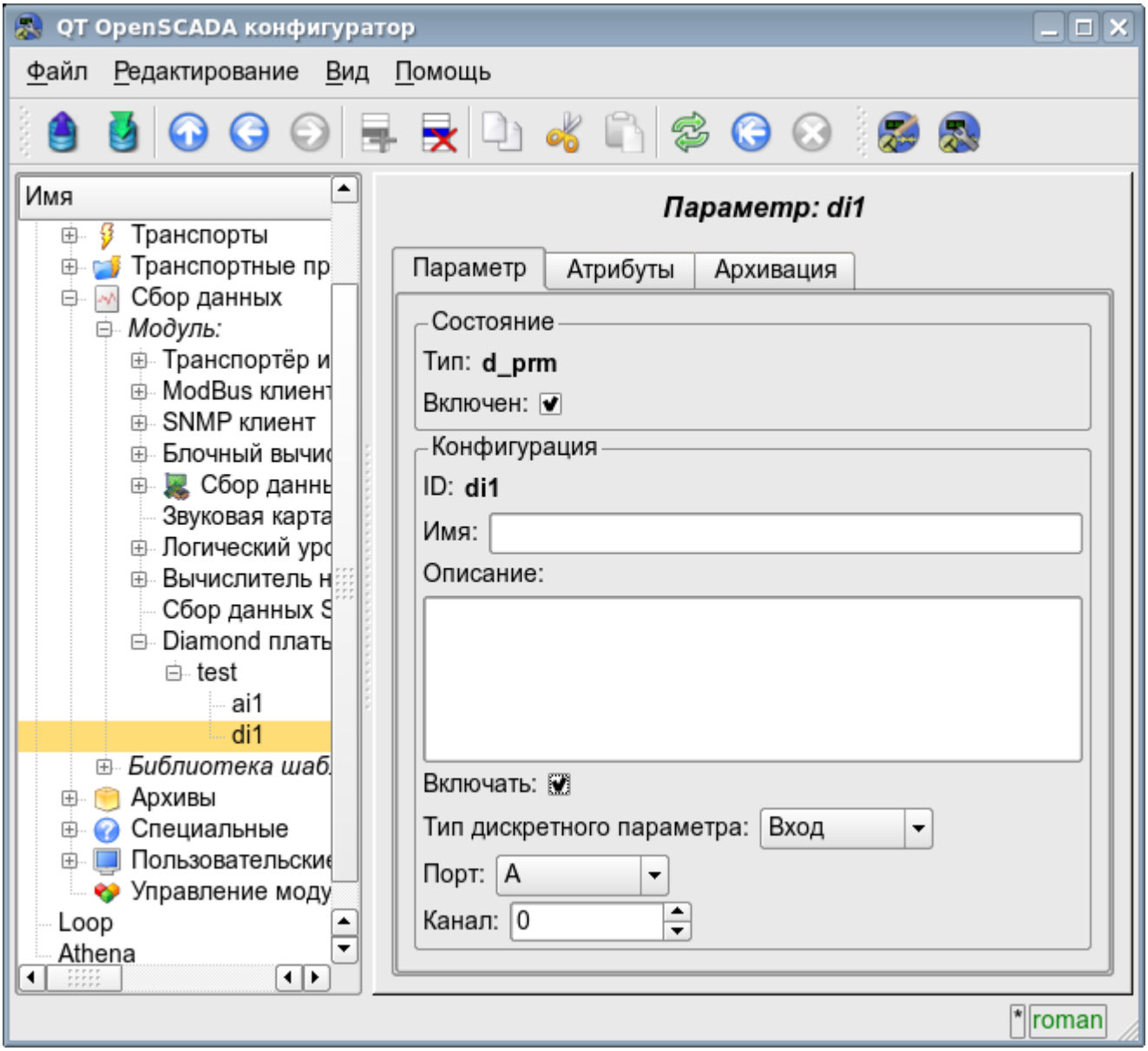


Рис.4. Вкладка конфигурации цифровых параметров.

С помощью вкладки конфигурации цифровых параметров можно установить:

- Режим параметра, а именно «Включен» и тип параметра.
- Идентификатор, имя и описание параметра.
- Состояние, в которое переводить параметр при загрузке: «Включен».
- Направленность параметра – «Вход» или «Выход».
- Физический порт и номер канала.

Для доступа к значениям цифровых параметров формируются атрибут *<value>*, предоставляющий входное значение или выставляющий выходное.

Модуль подсистемы “Сбор данных” <System>

Модуль:	System
Имя:	Сбор данных ОС
Тип:	DAQ
Источник:	daq_System.so
Версия:	1.7.2
Автор:	Роман Савоченко
Описание:	Предоставляет сбор данных из ОС. Поддерживаются источники данных ОС Linux: HDDTemp, LMSensors, Uptime, Memory, CPU и т.д.
Лицензия:	GPL

Модуль является, своего рода, шлюзом между системой OpenSCADA и ОС(операционной системой). Модуль получает данные из различных источников данных ОС и позволяет управлять компонентами ОС (в будущем).

Модуль предоставляет возможность автоматического поиска поддерживаемых и активных источников данных с созданием параметров для доступа к ним, а также реализация функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

1. Контроллер данных

Для добавления источника данных ОС создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

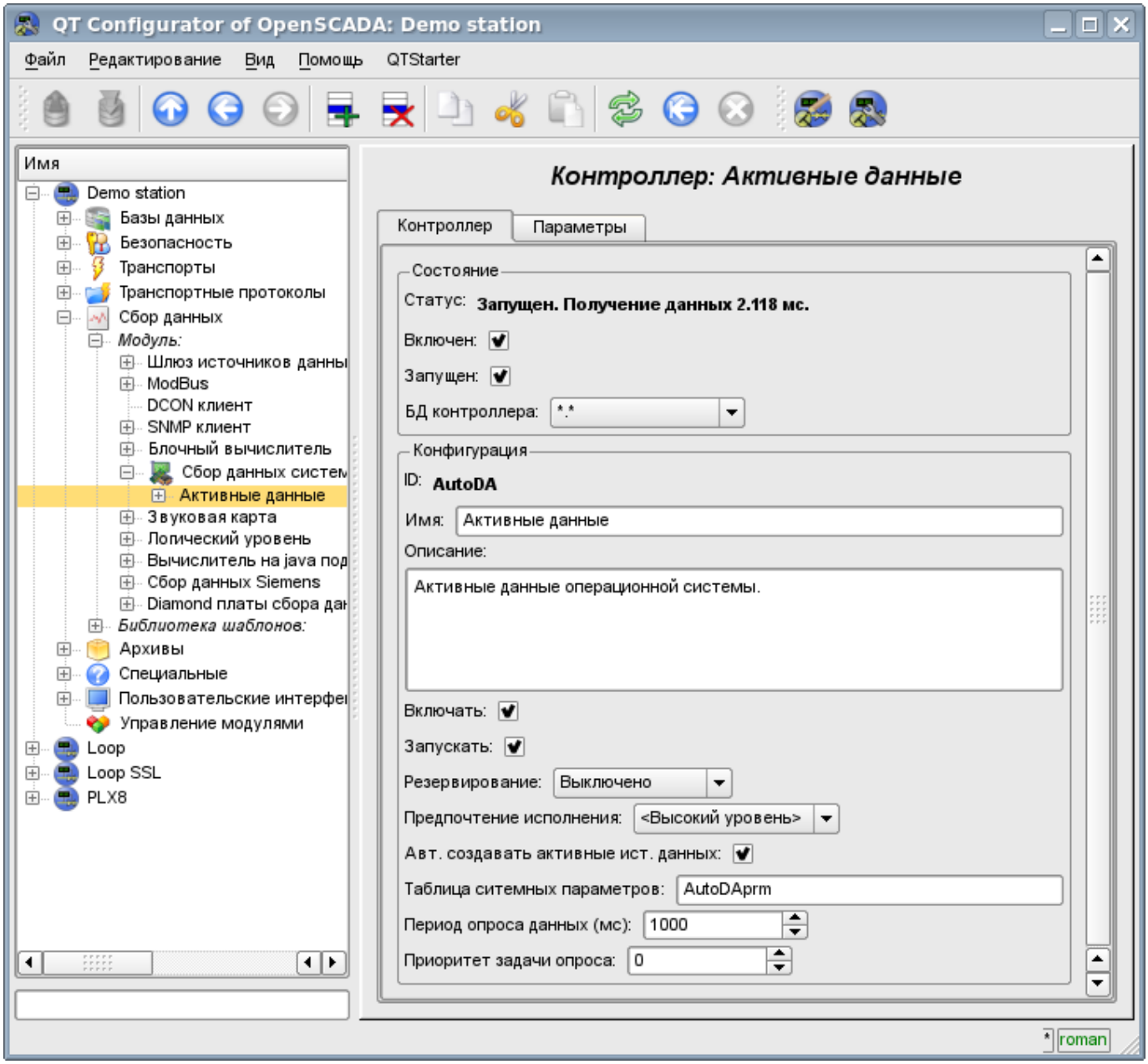


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Признак «Автоматический поиск активных источников данных и создание параметров для них».
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи опроса источников данных.

2. Параметры

Модуль *System* предоставляет только один тип параметров – “Все параметры”. Дополнительными конфигурационными полями параметров данного модуля (рис.2) являются:

- часть системы;
- дополнительный (зависит от источника данных).

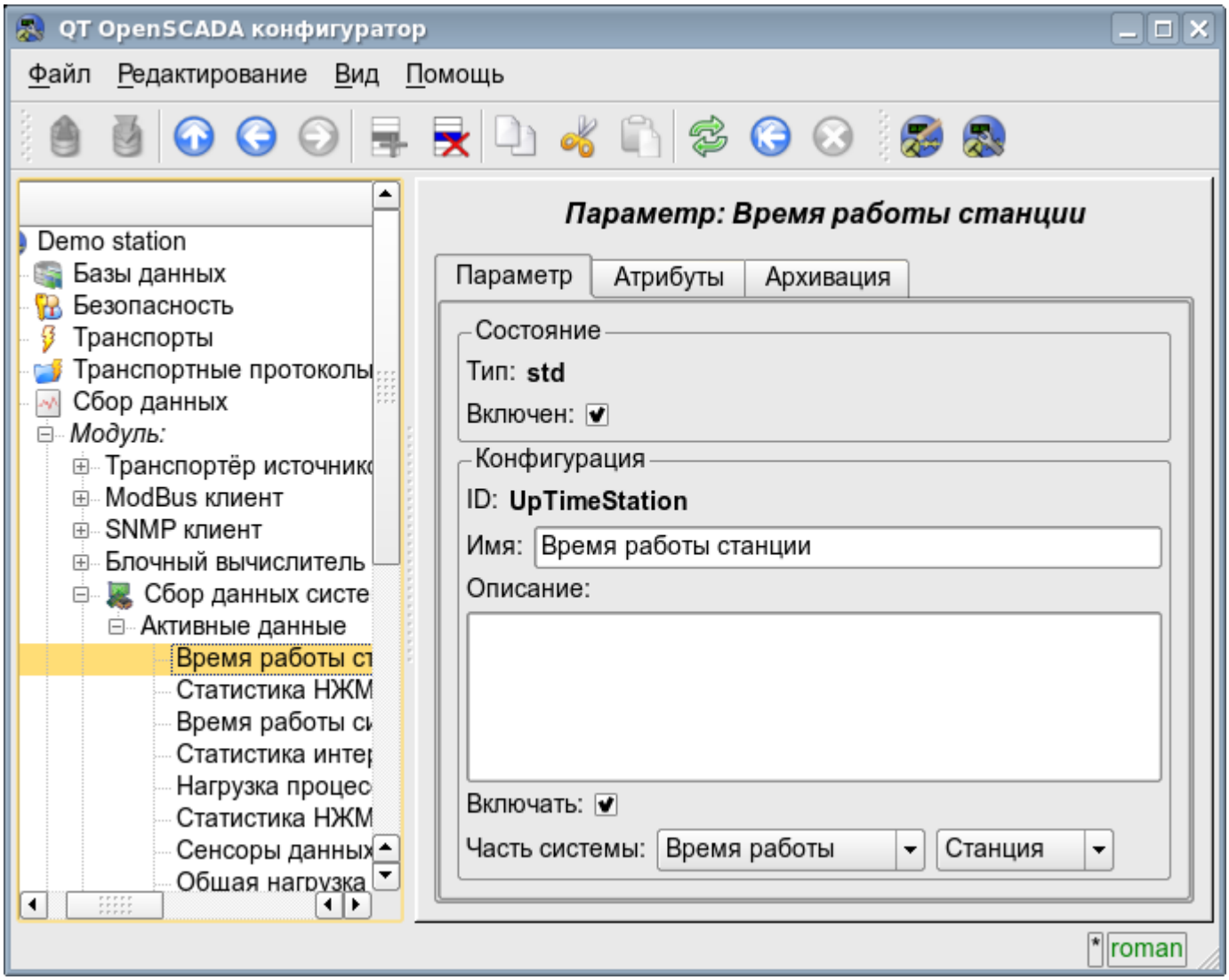


Рис.2. Вкладка конфигурации параметра.

В таблице ниже приведен список поддерживаемых источников данных ОС, значение дополнительного конфигурационного поля и атрибуты параметров.

Ист. данных	Значение доп. конфигурац. поля	Атрибуты параметра	Требования
Процессор (CPU)	Имя/номер процессора. Может иметь значение номера процессора или быть «в общем» по всем процессорам <gen>.	• [real] load:Нагрузка (%) • [real] sys:Система (%) • [real] user:Пользователь (%) • [real] idle:Простой (%)	

Ист. данных	Значение доп. конфигурац. поля	Атрибуты параметра	Требования
Память (MEM)	Не используется	• [dec] free:Свободно (кБ); • [dec] total:Всего (кБ); • [dec] use:Использовано (кБ); • [dec] buff:Буфера (кБ); • [dec] cache:Кеш (кБ); • [dec] sw_free:Своп, свободно (кБ); • [dec] sw_total:Своп, всего (кБ); • [dec] sw_use:Своп, использовано (кБ).	
Сенсоры (sensors)	Не используется	Атрибуты определяются сенсорами, доступными на материнской плате. Для каждого сенсора создаётся отдельный атрибут.	Для работы используется библиотека libsensors или программа mbmon. Более приоритетным в использовании является библиотека libsensors, поскольку mbmon имеет проблемы на многоядерных архитектурах.
Температура HDD (hddtemp)	Диск. Доступные в системе диски.	• [string] disk:Имя; • [string] ed:Единица измерения; • [real] t:Температура.	Должена быть установлена, сконфигурированна и запущена как сервис программа hddtemp
Время работы (uptime)	Время работы: • Система; • Станция.	• [dec] full:Секунды полностью; • [dec] sec:Секунды; • [dec] min:Минуты; • [dec] hour:Часы; • [dec] day:Дни.	
HDD Smart (hddsmart)	Диск. Доступные в системе диски.	Атрибуты определяются SMART-полями доступными для указанного диска. Для каждого поля создаётся отдельный атрибут.	Должна быть установлена и доступна утилита smartctl.
Статистика HDD (hddstat)	Диск или раздел. Доступные в системе диски и разделы.	Атрибуты: • [dec] rd:Прочитано (Кб); • [dec] wr:Записано (Кб).	
Статистика сети (netstat)	Сетевой интерфейс. Сетевые интерфейсы доступные в системе.	Атрибуты: • [dec] rcv:Принято (Кб); • [dec] trns:Передано (Кб).	

Модуль подсистемы “Сбор данных” <BlockCalc>

Модуль:	BlockCalc
Имя:	Блочный вычислитель.
Тип:	DAQ
Источник:	daq_BlockCalc.so
Версия:	1.4.0
Автор:	Роман Савоченко
Описание:	Предоставляет блочный вычислитель.
Лицензия:	GPL

Модуль подсистемы «DAQ» BlockCalc предоставляет в систему OpenSCADA механизм создания пользовательских вычислений. Механизм вычислений основывается на формальном языке блочных схем(функциональных блоков).

Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня. Кроме синхронизации значений и архивов атрибутов параметров модулем осуществляется синхронизация значений блоков блочной схемы, с целью безударного подхвата алгоритмов.

Языки блочного программирования основываются на понятии блочных схем (функциональных блоков). При чем в зависимости от сущности блока блочные схемы могут быть: логическими схемами, схемами релейной логики, моделью технологического процесса и другое. Суть блочной схемы состоит в том, что она содержит список блоков и связи между ними.

С формальной точки зрения блок это элемент (функция), который имеет входы, выходы и алгоритм вычисления. Исходя из концепции среды программирования, блок – это кадр значений, ассоциированный с объектом функции.

Разумеется, входы и выходы блоков нужно соединять для получения цельной блочной схемы. Предусмотрены следующие типы связей:

- межблочные, подключение входа одного блока к выходу другого, входа одного блока к входу другого и выход одного блока ко входу другого;
- дальние межблочные, соединение блоков контроллеров разных блочных схем данного модуля;
- коэффициенты, преобразование входа в постоянную, все входы/выходы по умолчанию инициируются как постоянные;
- внешний атрибут параметра.

Условно соединения блоков можно изобразить как связи между блоками в целом (рис. 1) или детализация связей (рис. 2). В процессе связывания параметров блоков допустимо соединение параметров любого типа. При этом, в процессе вычисления будет выполняться автоматическое приведение типов.

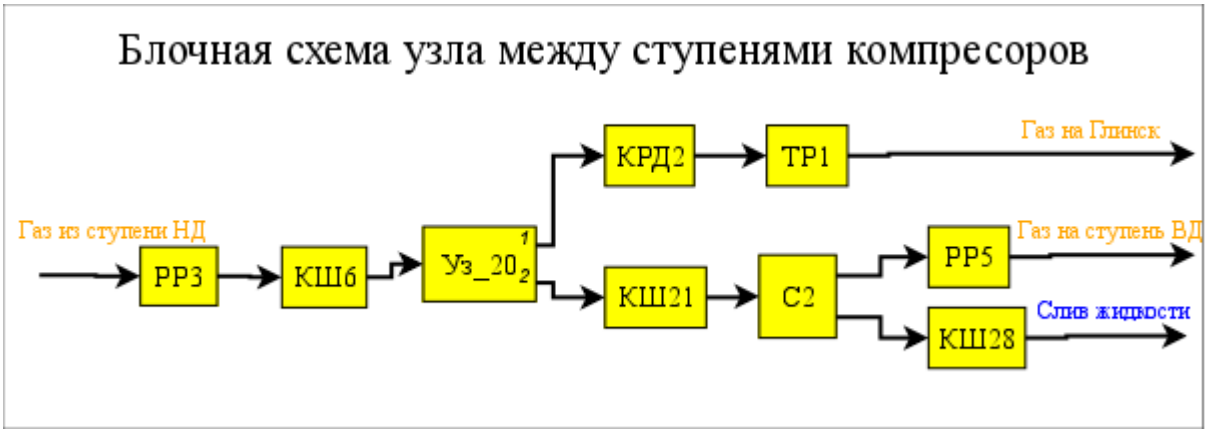


Рис. 1. Общие связи между блоками блочной схемы

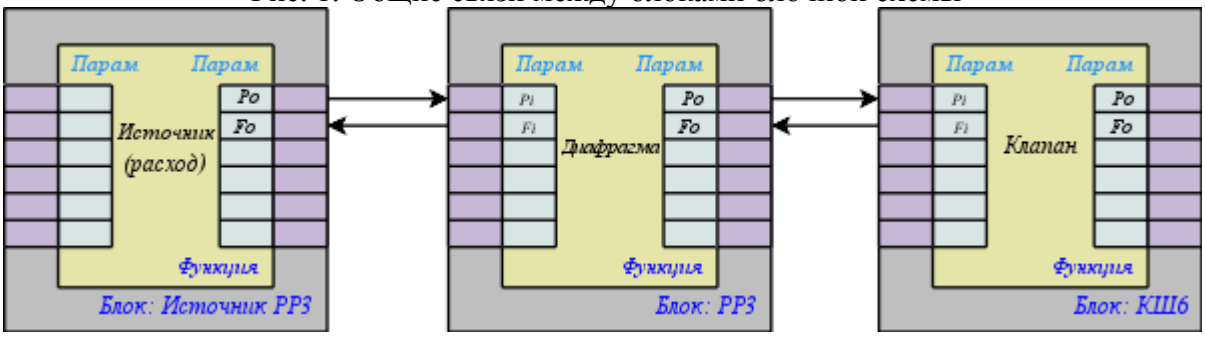


Рис. 2. Детализированные связи между блоками

1. Контроллер модуля

Каждый контроллер этого модуля содержит блочную схему, которую он обсчитывает с указанным периодом. Для предоставления вычисленных данных в систему OpenSCADA в контроллере могут создаваться параметры. Пример вкладки конфигурации контроллера данного типа изображен на рис.3.

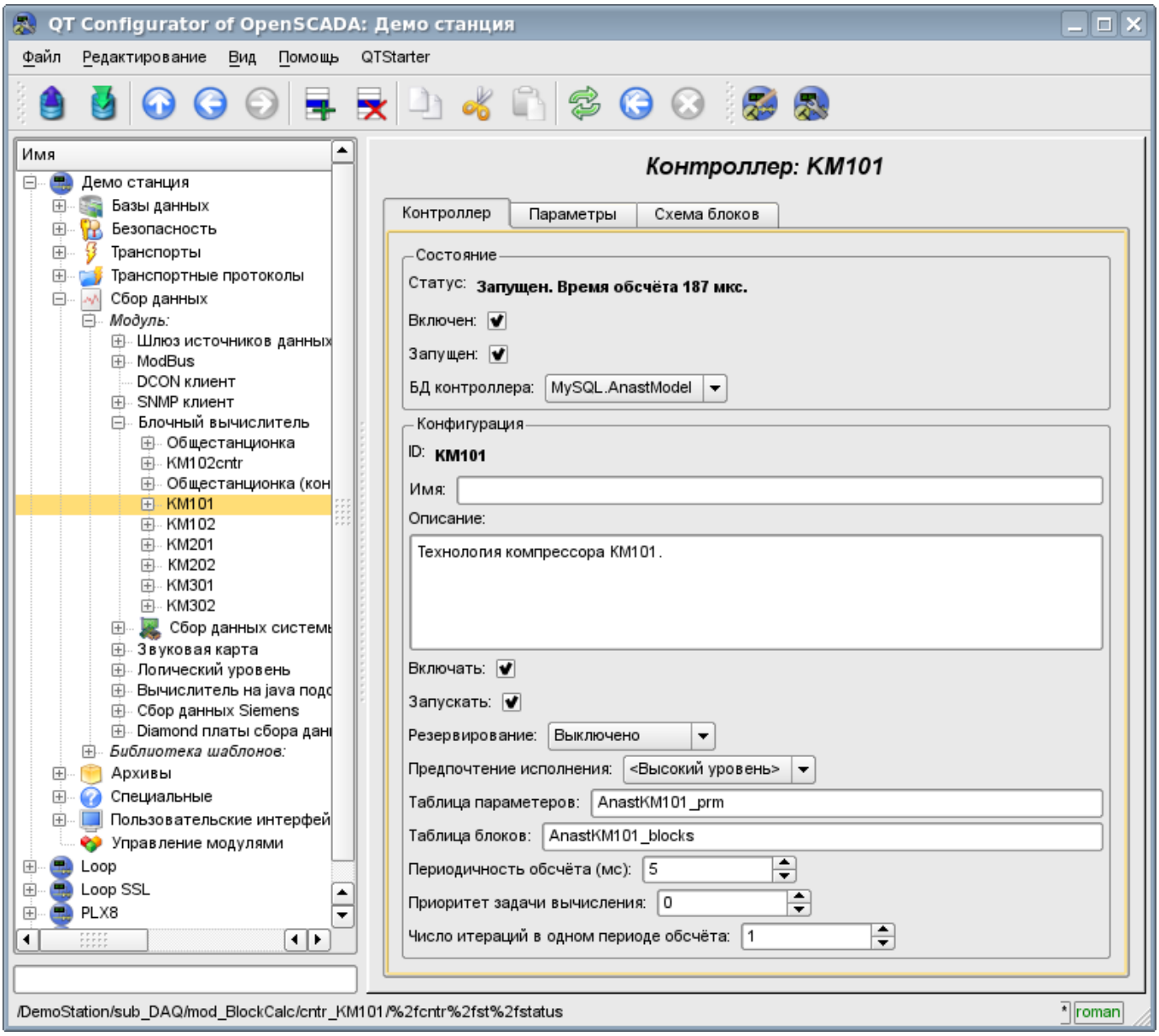


Рис. 3. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», «Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имена таблиц для хранения параметров и блоков контроллера.
- Период, приоритет и число итераций в одном цикле задачи вычисления блочной схемы контроллера.

2. Блочная схема контроллера

Блочная схема формируется посредством вкладки блоков контроллера, конфигурации блока (Рис.4) и его связей (Рис.5).

Блоки блочной схемы могут связываться как между собой, так и подключаться к атрибутам параметров. Сами блоки при этом не содержат структуры входов/выходов(IO), а содержат значения, исходя из структуры IO связанной функции. Функции для связывания с блоком используются из объектной модели системы OpenSCADA.

Любой блок может в любой момент быть исключён из обработки и переконфигурирован после чего может быть опять включен в обработку. Связи между блоками могут конфигурироваться без исключения блоков из обработки и остановки контроллера. Значения всех IO, не охваченных связями могут быть изменены в процессе обработки.

С помощью вкладки блоков можно:

- Добавить/удалить блок в блочную схему.
- Проконтролировать общее количество, количество включенных и количество обрабатываемых блоков.

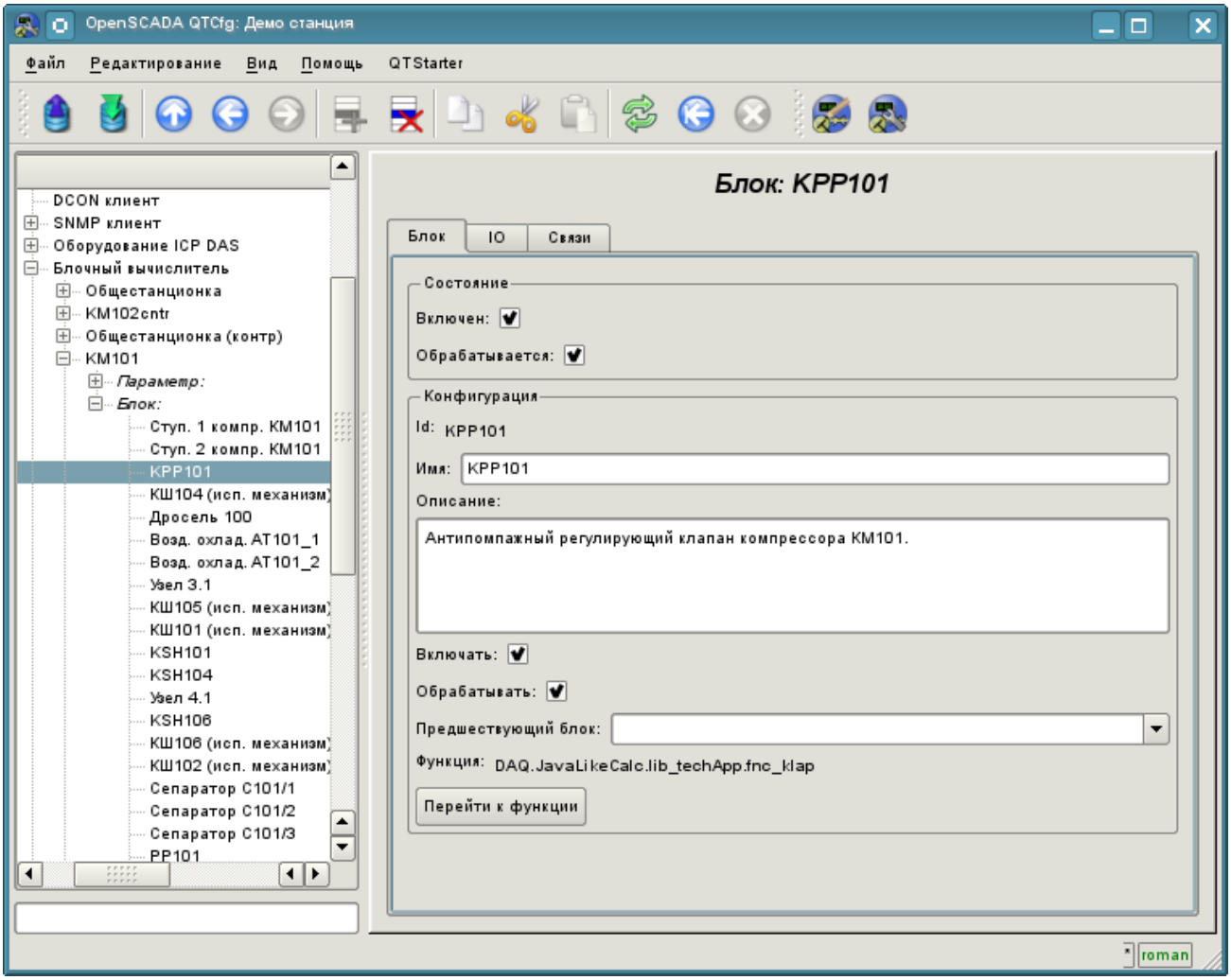


Рис. 4. Вкладка конфигурации блока блочной схемы.

С помощью формы конфигурации блока можно установить:

- Состояние блока, а именно: «Включен» и «Обрабатывается».
- Идентификатор, имя и описание блока.
- Состояние, в которое переводить блок при загрузке: «Включен» и «Запущен».
- Указать блок, который должен обязательно выполяться перед данным.
- Назначить рабочую функцию из объектной модели. Перейти к функции для ознакомления.

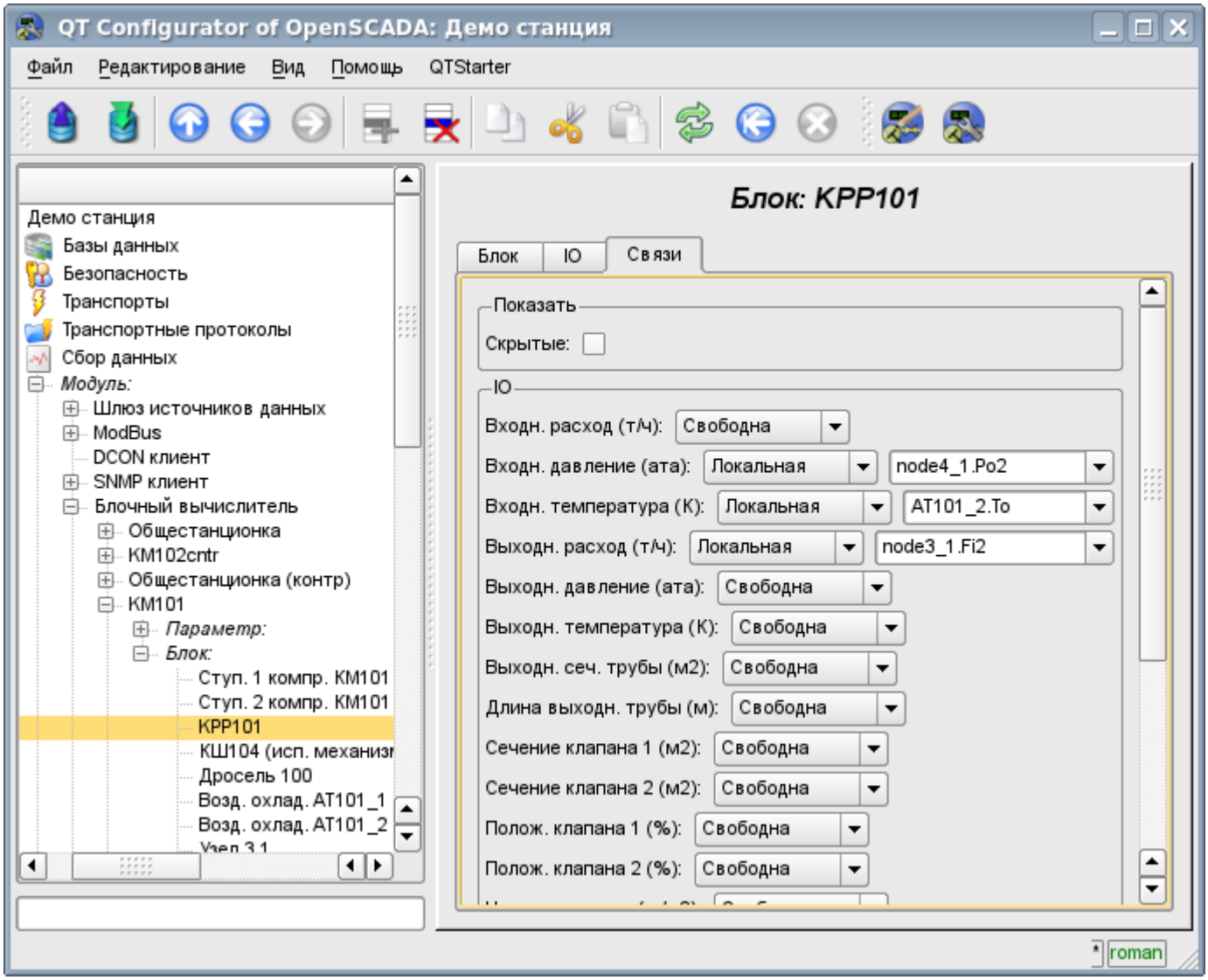


Рис. 5. Вкладка конфигурации связей блока блочной схемы.

С помощью вкладки конфигурации связей блока блочной схемы можно установить связи для каждого параметра блока отдельно.

Поддерживаются следующие типы связей:

- Межблочные. Подключение входа блока к выходу другого блока, входа одного блока к входу другого и выхода одного блока ко входу другого.
- Дальние межблочные. Соединение блоков из различных контроллеров данного модуля.
- Коэффициент. Превращение входа в константу. Все входы/выходы по умолчанию инициированы как константы.
- Внешний атрибут параметра.

Для установки значений параметров блока предназначена соответствующая вкладка (Рис.6).

В соответствии с реализацией пользовательских функций в системе OpenSCADA поддерживаются четыре основных типа IO: целое, вещественное, логическое и строка.

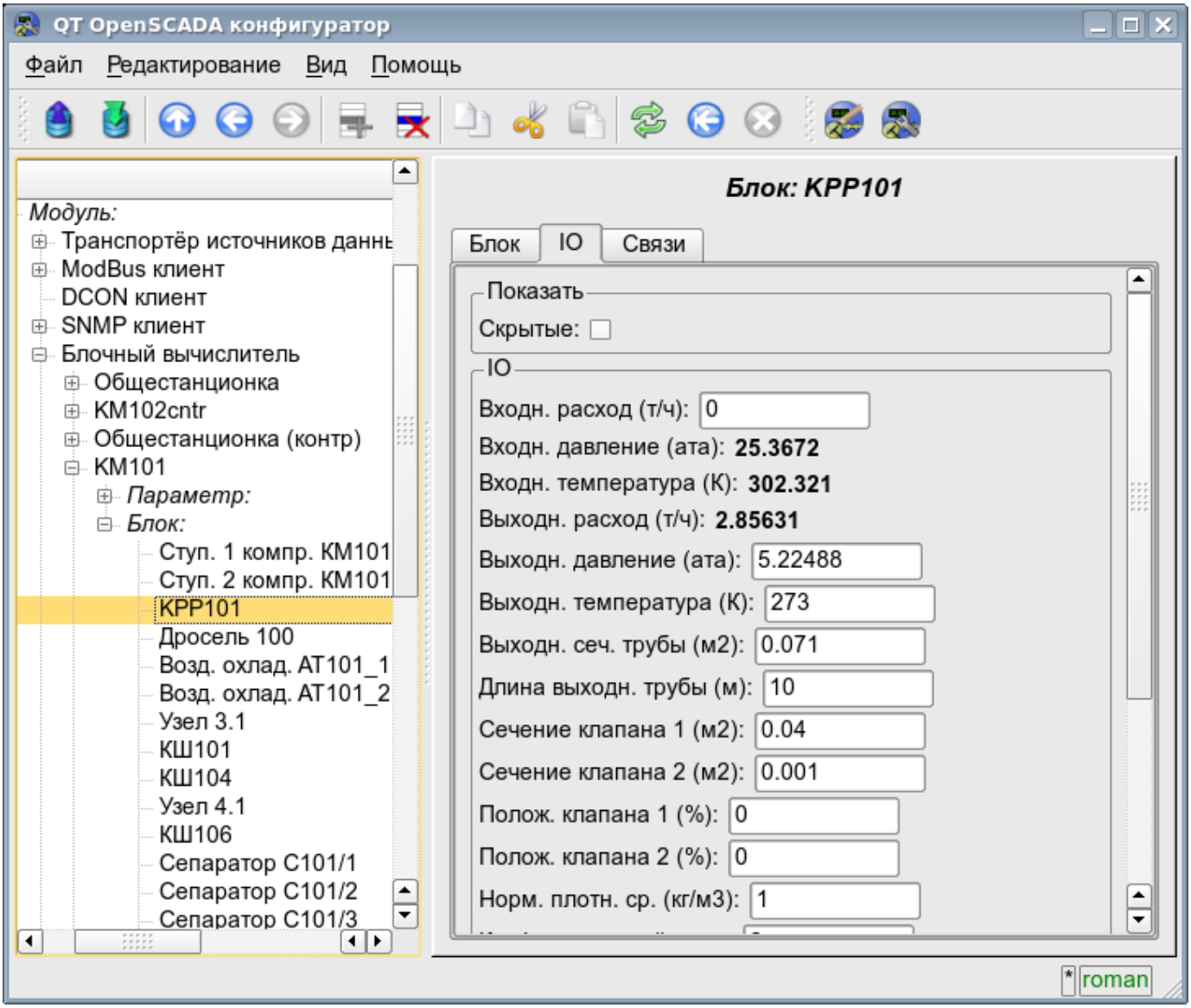


Рис. 6. Вкладка конфигурации значений параметров блока блочной схемы.

3. Параметры контроллера

Модуль предоставляет только один тип параметров “Стандартный”. Параметр служит для отражения вычисленных в блоках данных на атрибуты параметров контроллера. Пример вкладки конфигурации параметра приведен на Рис.7.

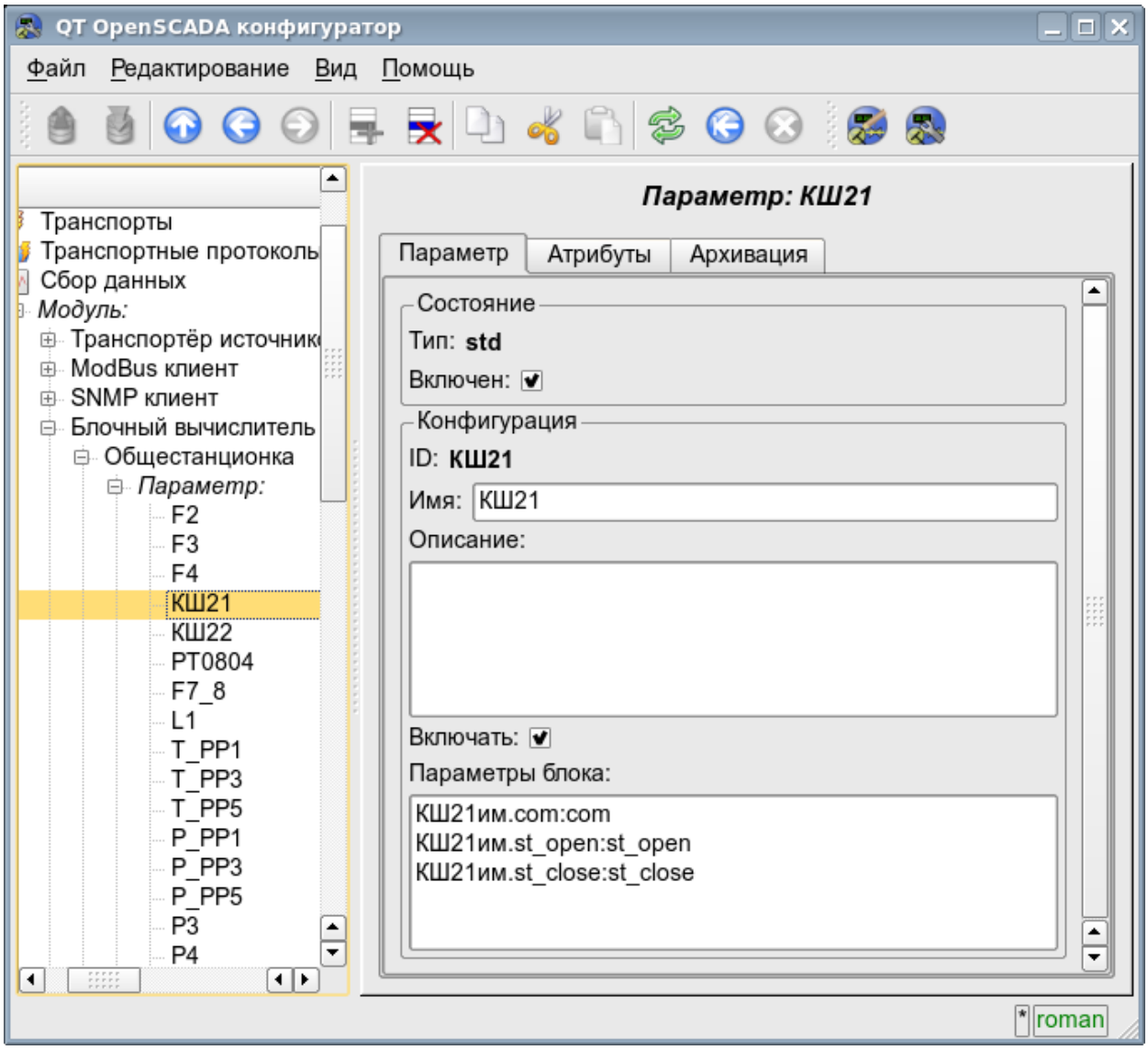


Рис. 7. Вкладка конфигурации значений параметров контроллера.

С помощью этой вкладки можно установить:

- Состояние параметра, а именно: «Включен» и тип параметра.
- Идентификатор, имя и описание параметра.
- Состояние, в которое переводить параметр при загрузке: «Включен».
- Перечень атрибутов, отражённых на параметры блоков. Формируется в виде списка элементов в формате: *<BLK>.<BLK_IO>:<AID>:<ANM>*. Где:
  - <BLK> - идентификатор блока, блочной схемы; для постоянной значение устанавливается:
    '*s' - строковый тип;
    '*i' - целочисленный тип;
    '*r' - вещественный тип;
    '*b' - логический тип.
  - <BLK_IO> - параметр блока, блочной схемы; для постоянной значений устанавливается в значение атрибута;
  - *<AID>* — идентификатор атрибута параметра;
  - *<ANM>* — имя атрибута параметра.

4. Копирование блочных схем

Для упрощения и ускорения процедуры разработки сложных и повторяющихся блочных схем предусмотрен механизм копирования элементов блочной схемы как по отдельности, так и блочных схем целиком. Механизм копирования интегрирован в ядро OpenSCADA и работает прозрачно.

Модуль подсистемы “Сбор данных” <JavaLikeCalc>

Модуль:	JavaLikeCalc
Имя:	Вычислитель на Java-подобном языке.
Тип:	DAQ
Источник:	daq_JavaLikeCalc.so
Версия:	1.8.0
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

Модуль контроллера *JavaLikeCalc* предоставляет в систему OpenSCADA 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.

Модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня. Кроме синхронизации значений и архивов атрибутов параметров модулем осуществляется синхронизация значений вычислительной функции, с целью безударного подхвата алгоритмов.

Параметры функции могут свободно создаваться, удаляться или модифицироваться. Текущая версия модуля поддерживает до 65535 параметров функции в сумме с внутренними переменными. Вид редактора функций показан на рис.1.

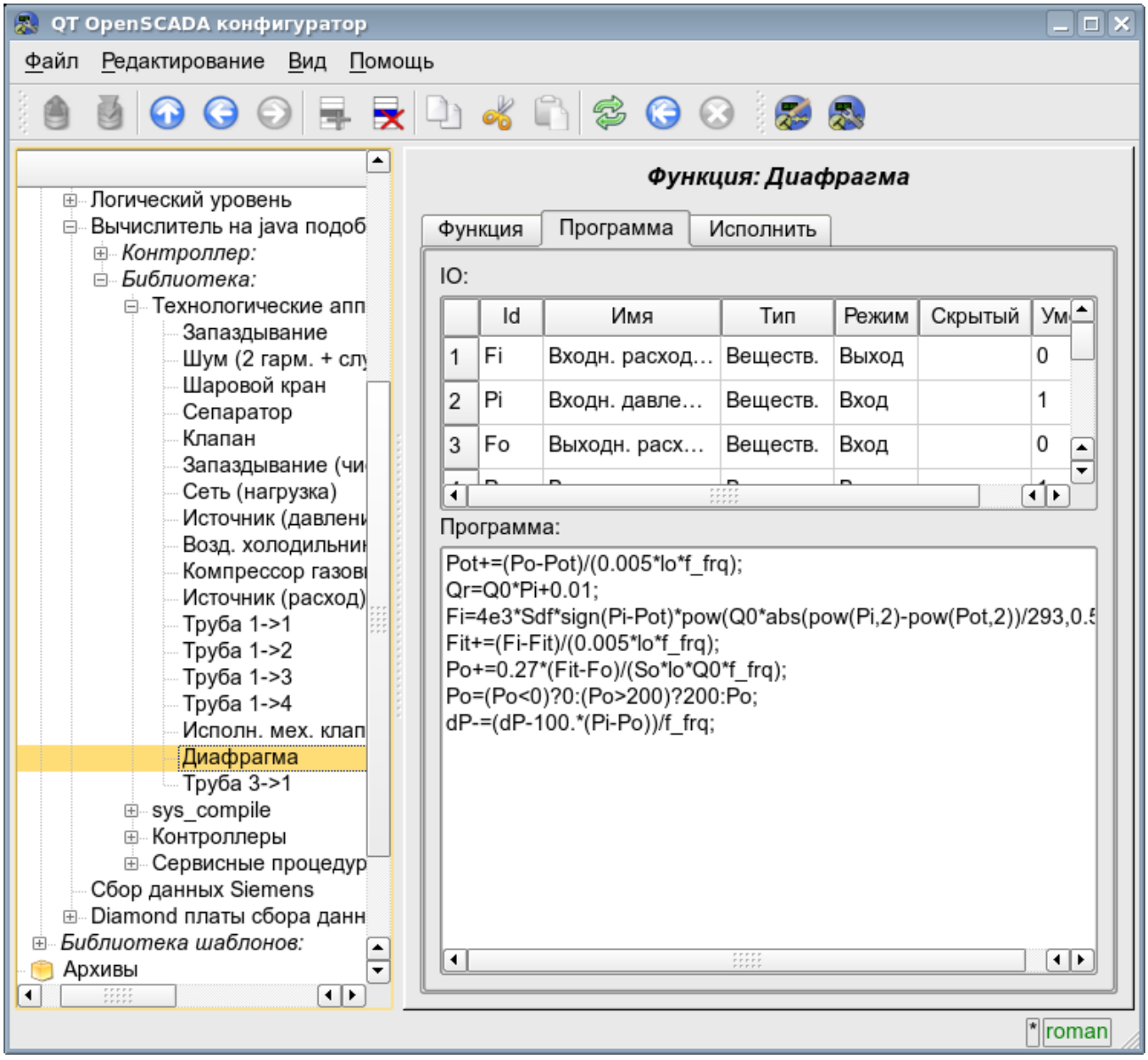


Рис.1. Вид редактора функций.

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, функции объекта и системных узлов OpenSCADA (DOM);
- уже зарегистрированные символы переменных, атрибуты объектов и иерархия объектов DOM;
- 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 до 255 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 сверху 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 со 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:

- *var* - оператор инициализации переменной;
- *if* - оператор условия "Если";
- *else* - оператор условия "Иначе";
- *while* - описание цикла while;
- *for* - описание цикла for;
- *in* - разделитель цикла for для перебора свойств объекта;
- *break* - прерывание выполнения 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;);
- *new* - создание объекта, реализованы объект "Object" и массив "Array".

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 три типа циклов: while, for и for-in. Синтаксис циклов соответствует 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>;*

Цикл **for-in** записывается следующим образом: *for(<переменная> in <объект>) <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;
- *<переменная>* - переменная, которая будет содержать имя свойства объекта при переборе;
- *<объект>* - объект для которого осуществляется перебор свойств.

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. Объект

Языком предоставляется поддержка типа данных объект “Object”. Объект представляет собой ассоциативный контейнер свойств и функций. Свойства могут содержать как данные четырёх базовых типов, так и другие объекты. Доступ к свойствам объекта может осуществляться посредством записи имён свойств через точку к объекту *<obj.prop>*, а также посредством заключения имени свойства в квадратные скобки *<obj["prop”]>*. Очевидно, что первый механизм статичен, а второй позволяет указывать имя свойства через переменную. Создание объекта осуществляется посредством ключевого слова *<new>*: *<varO = new Object()>*. Базовое определение объекта не содержит функций. Операции копирования объекта на самом деле делают ссылку на исходный объект. При удалении объекта осуществляется уменьшения счётчика ссылок, а при достижении счётчика ссылок нуля объект удаляется физически.

Разные компоненты могут доопределять базовый объект особыми свойствами и функциями. Стандартным расширением объекта является массив “Array”, который создаётся командой *<varO = new Array(prm1,prm2,prm3,...,prmN)>*. Перечисленные через запятую параметры помещаются в массив в исходном порядке. Если параметр только один то массив инициируется указанным количеством пустых элементов. Особенностью массива является то, что он работает со свойствами как с индексами и полное их именование бессмысленно, а значит доступен механизм обращения только заключением индекса в квадратные скобки *<arr[1]>*. Массив хранит свойства в собственном контейнере одномерного массива.

Массив предоставляет специальное свойство “length” для получения размера массива *<var = arr.length;>*. Также массив предоставляет следующие специальные функции:

- *string join(string sep = ",")*, *string toString(string sep = ",")*, *string valueOf(string sep = ",")* - Возвращает строку с элементами массива разделёнными *<sep>* или символом ','.
- *Array concat(Array arr);* - Добавляет к исходному массиву элементы массива *<arr>*. Возвращает исходный массив с изменениями.
- *int push(ElTp var, ...);* - Помещает элемент(ы) *<var>* в конец массива, как в стек. Возвращает новый размер массива.

- *ElTp pop( );* - Удаление последнего элемента массива и возврат его значения, как из стека.
- *Array reverse( );* - Изменение порядка расположения элементов массива. Возвращается исходный массив с изменениями.
- *ElTp shift( );* - Сдвиг массива в верх. При этом первый элемент массива удаляется, а его значение возвращается.
- *int unshift(ElTp var, ...);* - Задвигает элемент(ы) <*var*> в массив. Первый элемент в 0, второй в 1 и т.д.
- *Array slice(int beg, int end);* - Возвращает фрагмент массива от <*beg*> к <*end*>. Если значение начала или конца отрицательно, то отсчёт ведётся с конца массива. Если конец не указан, то концом является конец массива.
- *Array splice(int beg, int remN, ElTp val1, ElTp val2, ...);* - Вставляет, удаляет или заменяет елементы массива. Возвращает исходный массив с изменениями. В первую очередь осуществляется удаление элементов с позиции <*beg*> и количеством <*remN*>, а затем вставляются значения <*val1*> и т.д. начиная с позиции <*beg*>.
- *Array sort( );* - Сортировка элементов массива в лексикографическом порядке.

Частичными свойствами объекта обладают и базовые типы. Свойства и функции базовых типов приведены ниже:

- Логический тип, функции:
  - *bool isEVal( );* - Проверка значения на "EVAL".
  - *string toString( );* - Представление значения в виде строки "true" или "false".
- Целое и вещественное число:
  Свойства:
  - *MAX_VALUE* - максимальное значение;
  - *MIN_VALUE* - минимальное значение;
  - *NaN* - недостоверное значение.

  Функции:
  - *bool isEVal( );* - Проверка значения на "EVAL".
  - *string toExponential(int numbs);* - Возврат строки отформатированного числа в экспоненциальной нотации и количеством значащих цифр <*numbs*>. Если <*numbs*> отсутствует то цифр будет столько сколько необходимо.
  - *string toFixed(int numbs);* - Возврат строки отформатированного числа в нотации с фиксированной точкой и количеством цифр после десятичной точки <*numbs*>. Если <*numbs*> отсутствует то количество цифр после десятичной точки равно нулю.
  - *string toPrecision(int prec);* - Возврат строки отформатированного числа с количеством значащих цифр <*prec*>.
  - *string toString(int base);* - Возврат строки отформатированного числа целого типа с базой представления 8-восьмеричное, 10-десятичное и 16-шестнадцатеричное.
- Строка:
  Свойства:
  - *int length* - длина строки.

  Функции:
  - *bool isEVal( );* - Проверка значения на "EVAL".
  - *string charAt(int symb);* - Извлекает из строки символ <*symb*>.
  - *int charCodeAt(int symb);* - Извлекает из строки код символа <*symb*>.
  - *string concat(string val1, string val2, ...);* - Возвращает новую строку сформированную путём присоединения значений <*val1*> и т.д. к исходной.
  - *int indexOf(string substr, int start);* - Возвращает позицию искомой строки <*substr*> в исходной строке начиная с позиции <*start*>. Если исходная позиция не указана то поиск начинается с начала. Если искомой строки не найдено то возвращается -1.
  - *int lastIndexOf(string substr, int start);* - Возвращает позицию искомой строки <*substr*> в исходной строке начиная с позиции <*start*> при поиске с конца. Если исходная позиция не указана то поиск начинается с конца. Если искомой строки не найдено то возвращается -1.
  - *string slice(int beg, int end); string substring(int beg, int end);* - Возврат подстроки

извлечённой из исходной начиная с позиции <*beg*> и заканчивая <*end*>. Если значение начала или конца отрицательно, то отсчёт ведётся с конца строки. Если конец не указан, то концом является конец строки.

- *Array split(string sep, int limit);* - Возврат массива элементов строки разделённых <*sep*> с ограничением количества элементов <*limit*>.
- *string insert(int pos, string substr);* - Вставка в позицию <*pos*> текущей строки подстроку <*substr*>.
- *string replace(int pos, int n, string substr);* - Замена подстроки с позиции <*pos*> и длиной <*n*> в текущей строке на подстроку <*substr*>.
- *real toReal();* - преобразование текущей строки в вещественное число.
- *int toInt(int base = 0);* - преобразование текущей строки в целое число, в соответствии с основанием <base> (от 2 до 36). Если основание равно 0 то будет учитываться префиксная запись для определения основания (123-десятичное; 0123-восьмеричное; 0x123-шестнадцатиричное).
- *string parse(int pos, string sep = ".", int off = 0);* - выделение из исходной строки элемента <*pos*> для разделителя элементов <*sep*> от смещения <*off*>. Результирующее смещение помещается назад в <*off*>.
- *string parsePath(int pos, int off = 0);* - выделение из исходного пути элемента <*pos*> от смещения <*off*>. Результирующее смещение помещается назад в <*off*>.
- *string path2sep(string sep = ".");* - преобразование пути в текущей строке в строку с разделителем <*sep*>.

Для доступа к системным объектам(узлам) OpenSCADA предусмотрен соответствующий объект, который создаётся путём простого указания точки входа "SYS" корневого объекта OpenSCADA, а затем, через точку указываются вложенные объекты в соответствии с иерархией. Например, вызов функции запроса через исходящий транспорт осуществляется следующим образом: *SYS.Transport.Sockets.out_testModBus.messIO(strEnc2Bin("15 01 00 00 00 06 01 03 00 00 00 05"));*.

1.6. 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.

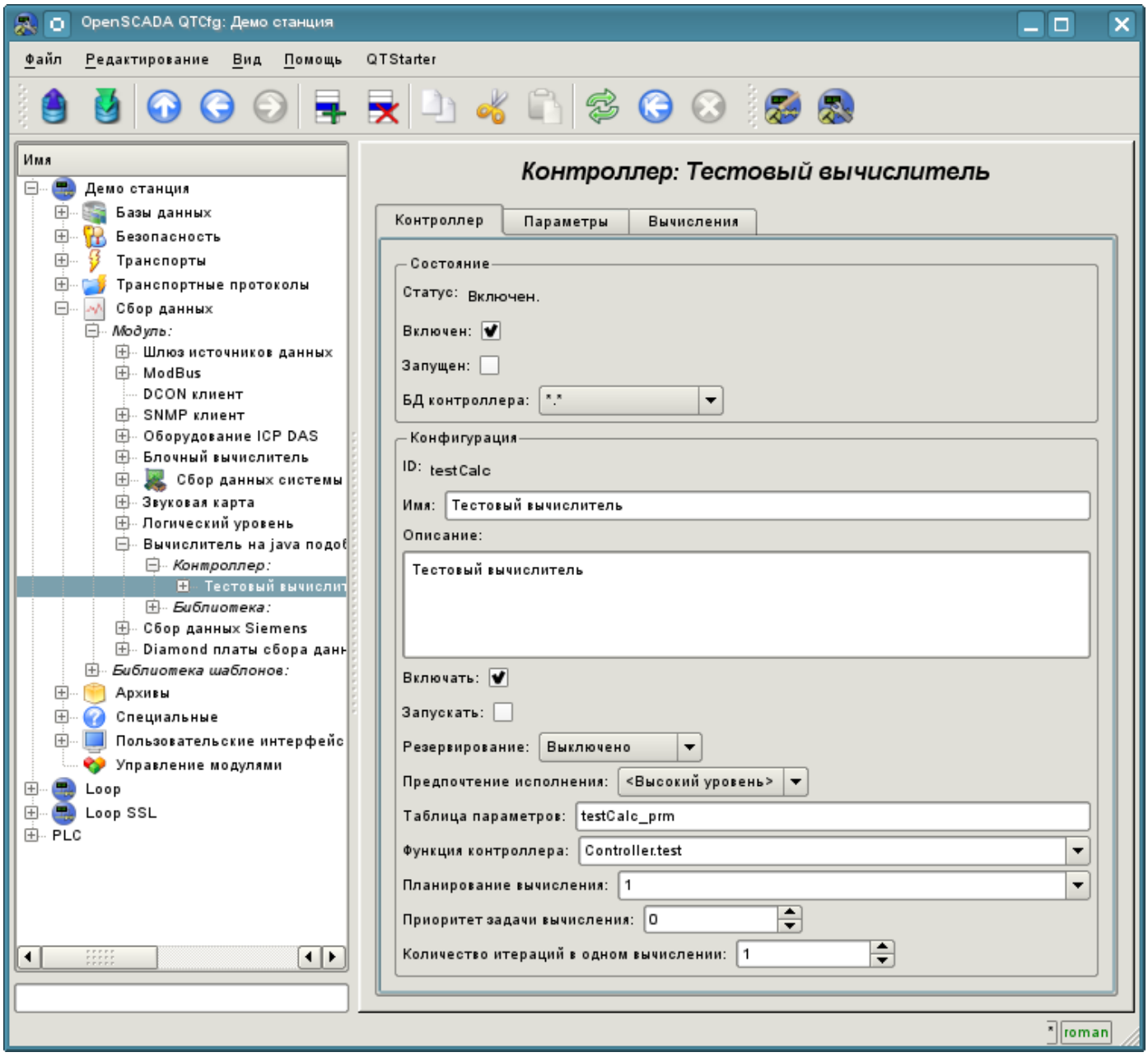


Рис.2. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», «Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- 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.

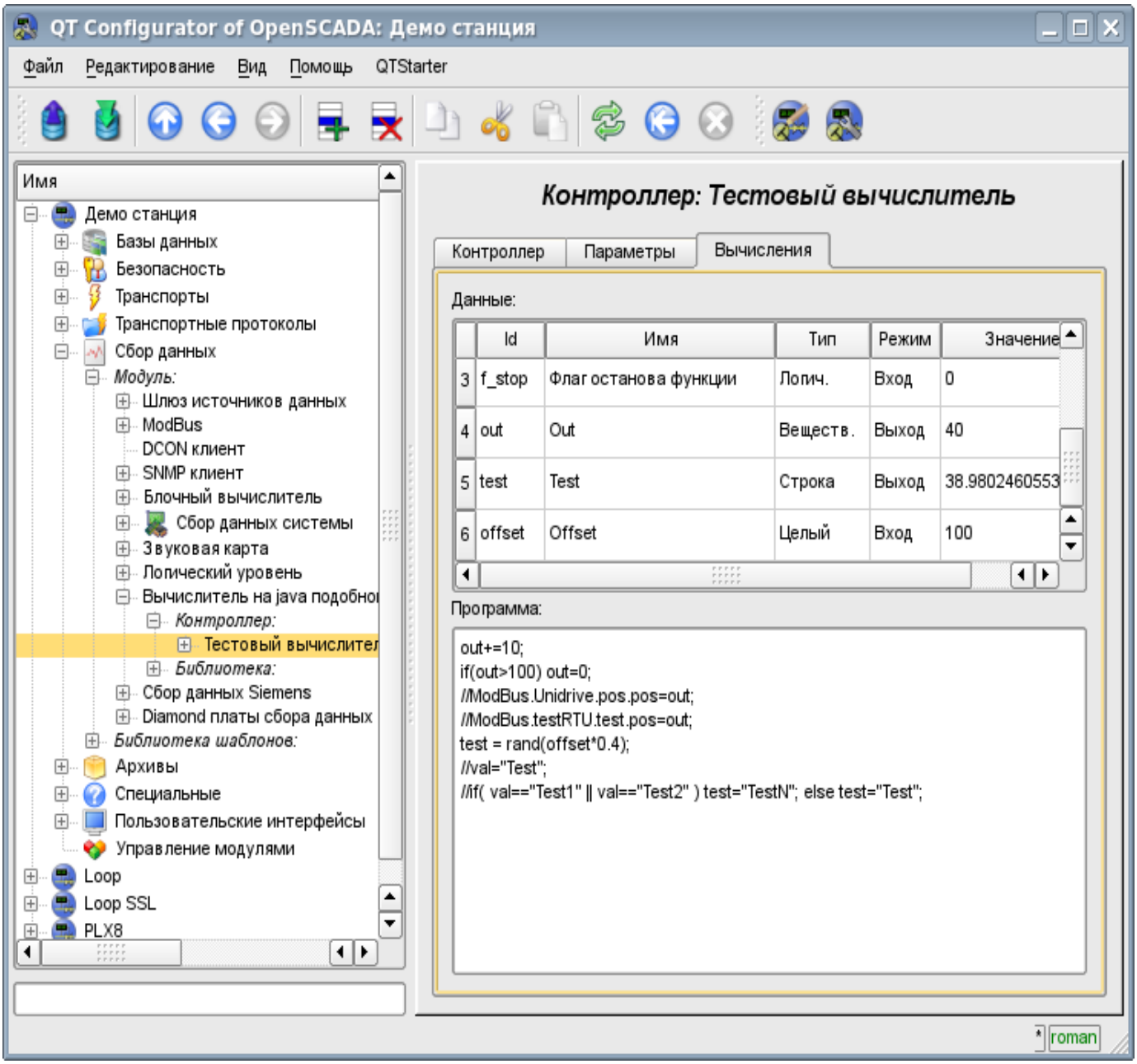


Рис.3. Вкладка «Вычисления» контроллера.

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.

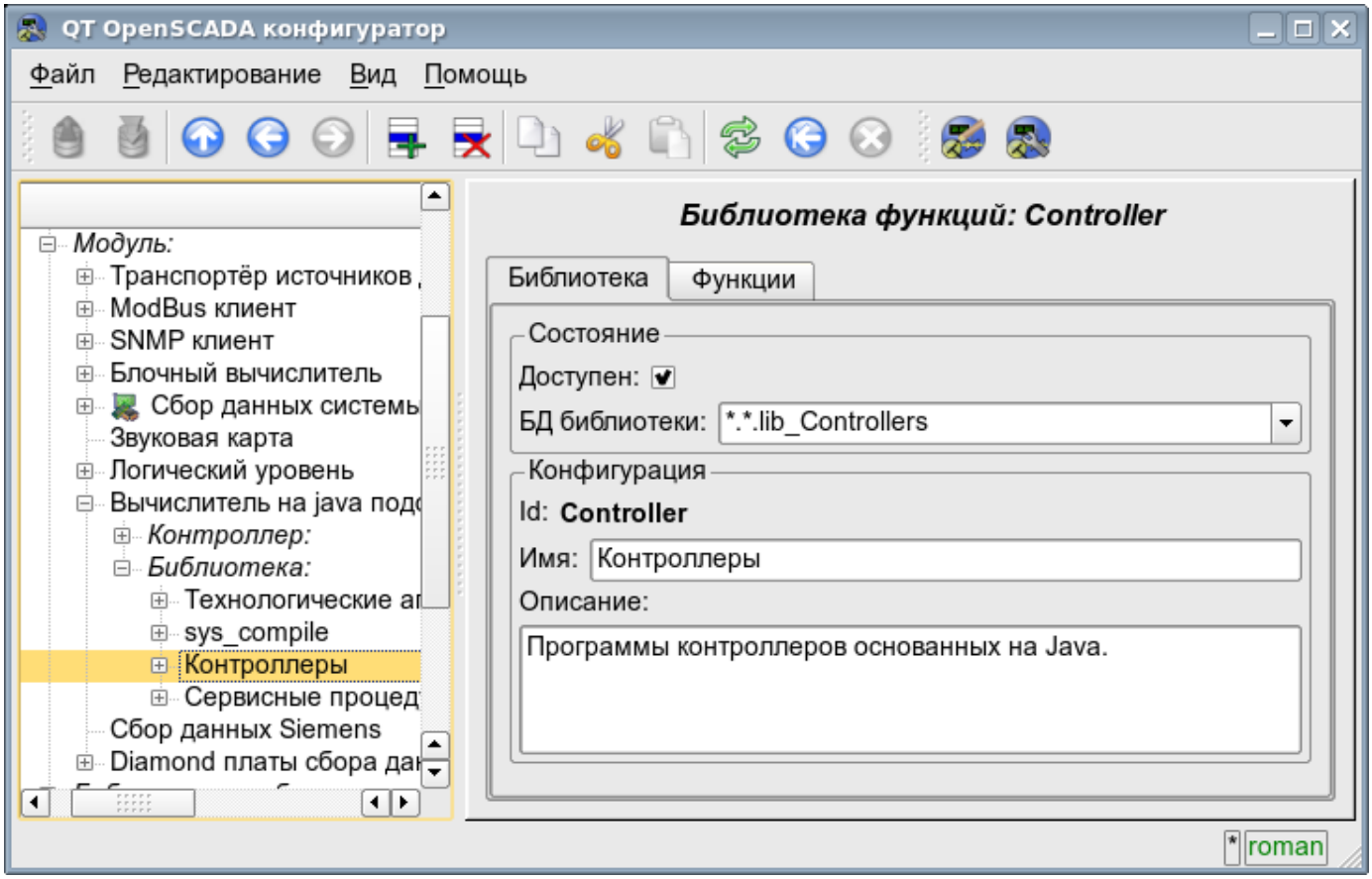


Рис.4. Вкладка конфигурации библиотеки.

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.

Модуль подсистемы “Сбор данных” <LogicLev>

Модуль:	LogicLev
Имя:	Логический уровень
Тип:	DAQ
Источник:	daq_LogicLev.so
Версия:	1.1.2
Автор:	Роман Савоченко
Описание:	Предоставляет логический уровень параметров.
Лицензия:	GPL

Модуль является чистой реализацией механизма логического уровня, основанного на шаблонах параметров подсистемы «Сбор данных – DAQ». Реализация модуля основана на проекте «Логический уровень параметров системы OpenSCADA" /Doc/LogParmUrov. Практически, данный модуль является реализацией подсистемы «Параметры» указанного проекта без шаблонов и вынесенная в модуль.

Модуль предоставляет механизм формирования параметров подсистемы “DAQ” на основе других источников этой подсистемы на уровне пользователя. Фактически, модулем используются шаблоны подсистемы “DAQ” и специфический формат описания ссылок на атрибуты параметров подсистемы “DAQ”.

Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня. Кроме синхронизации значений и архивов атрибутов параметров модулем осуществляется синхронизация значений вычислительных шаблонов, с целью безударного подхвата алгоритмов.

1. Контроллер данных

Для добавления источника данных параметров логического уровня создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

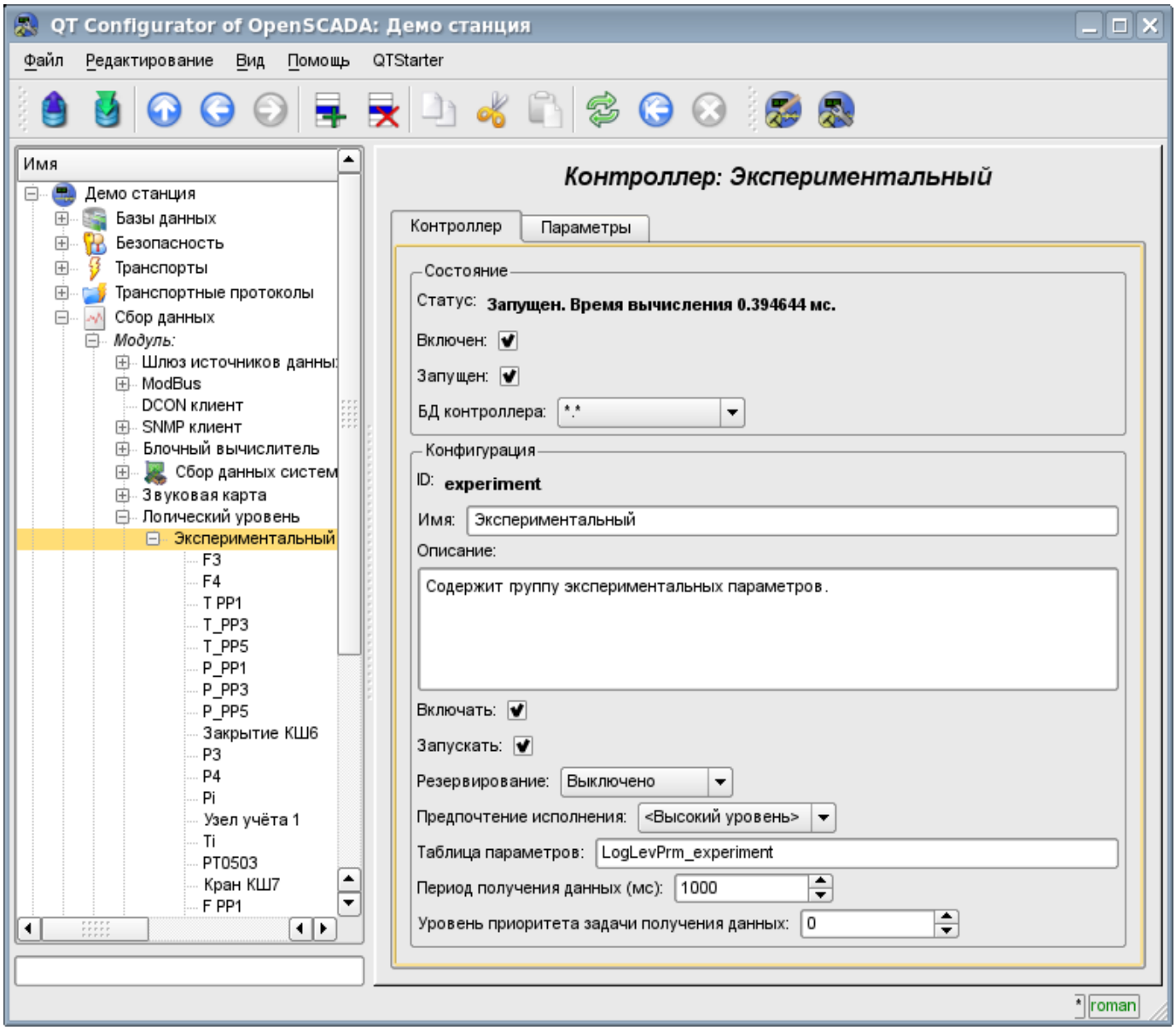


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи опроса источников данных.

2. Параметры

Модуль *LogicLev* предоставляет только один тип параметров – «Стандартный». Дополнительными конфигурационными полями параметров данного модуля (рис.2) являются:

- режим параметра;
- адрес; в случае шаблона – это адрес шаблона, а в случае прямого отражения – это адрес параметра.

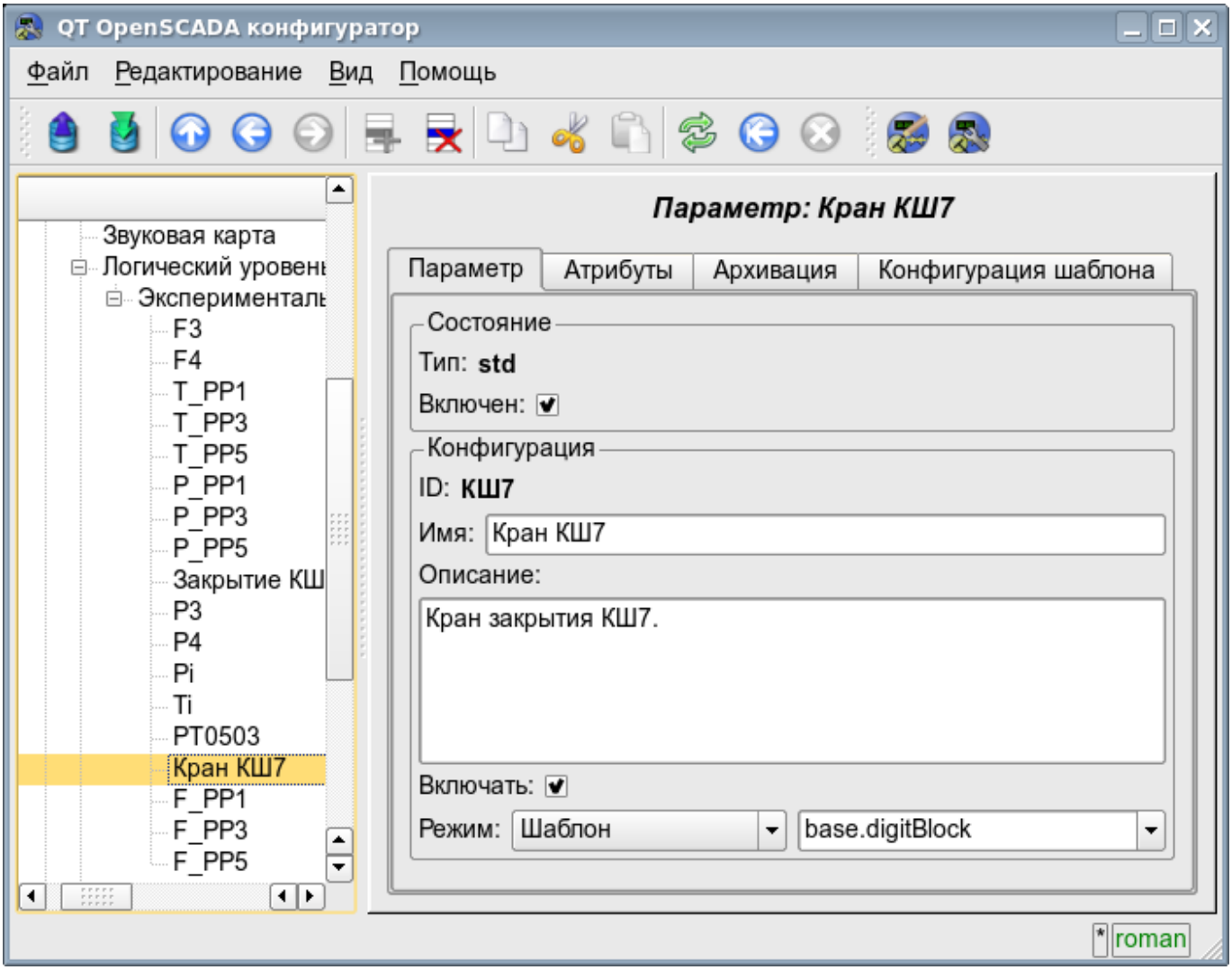


Рис.2. Вкладка конфигурации параметра.

При формировании шаблона для данного контроллера нужно учитывать особенность формата ссылки шаблона. Ссылка должна записываться в виде: **<Параметр>|<Идентификатор>**, где:

<Параметр> — строка, характеризующая параметр;

<Идентификатор> — идентификатор атрибута параметра.

Подобная запись позволяет группировать несколько атрибутов одного исходного параметра и назначать их выбором только параметра. Т.е. в диалоге конфигурации шаблона (рис.3) будет указываться только параметр. Это не исключает возможности назначать атрибуты параметров отдельно каждый, кроме того, если опустить в конфигурации шаблона описание ссылки в указанном формате, то назначаться будет атрибут параметра (рис.4).

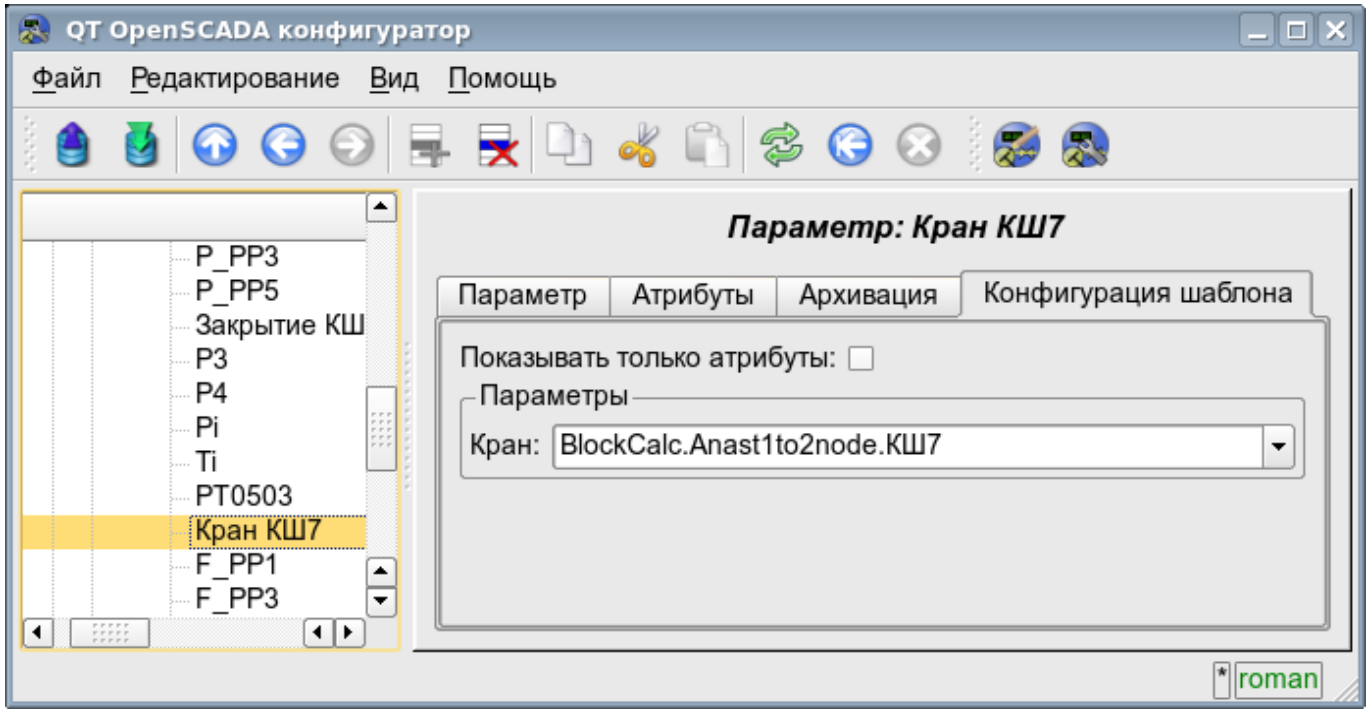


Рис.3. Вкладка конфигурации шаблона параметра.

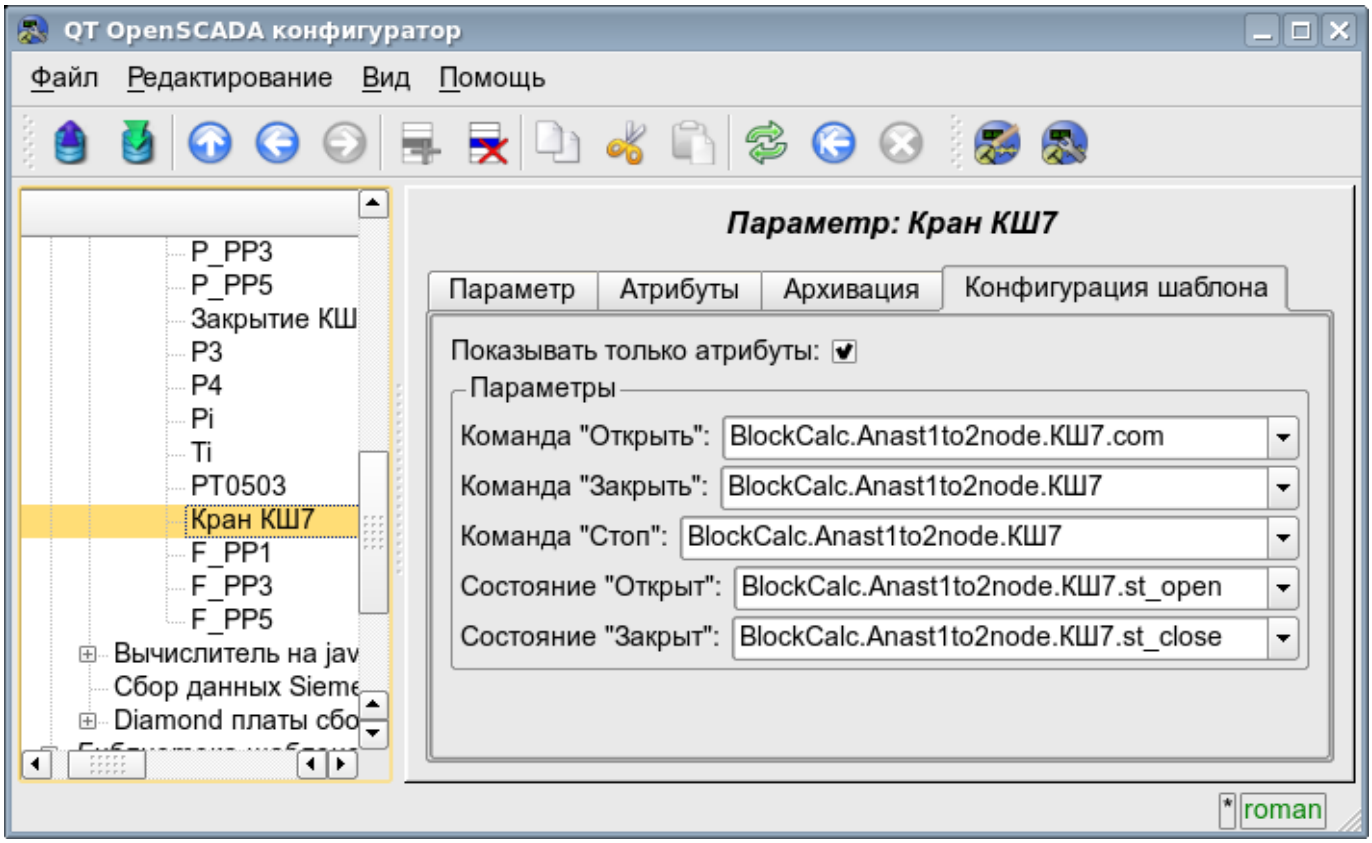


Рис.4. Вкладка конфигурации шаблона параметра. Показывать только атрибуты

В соответствии с шаблоном, лежащим в основе параметра, мы получаем набор атрибутов параметра рис.5.

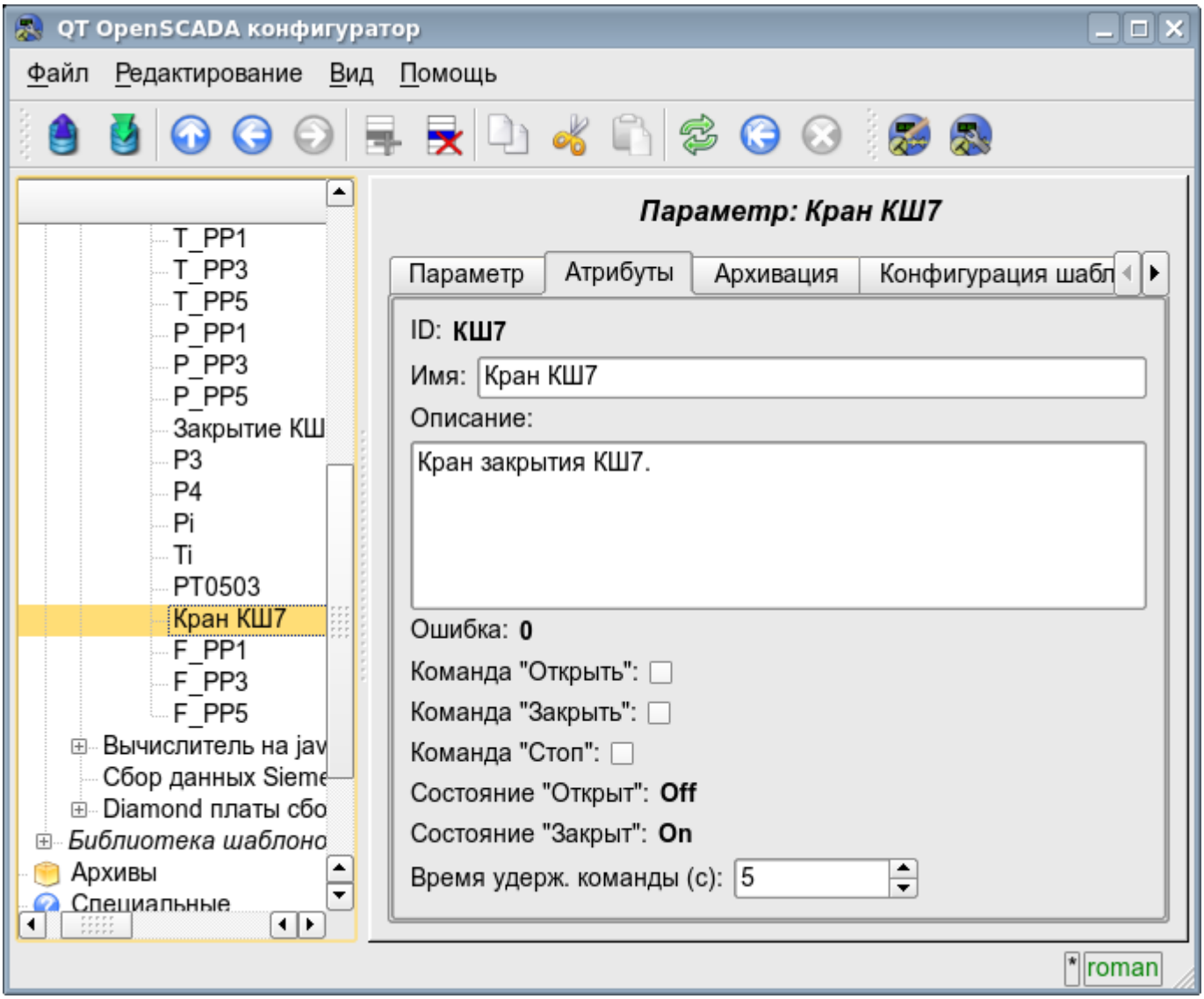


Рис.5. Вкладка атрибутов параметра.

Модуль подсистемы “Сбор данных” <SNMP>

Модуль:	SNMP
Имя:	SNMP клиент
Тип:	DAQ
Источник:	daq_SNMP.so
Версия:	0.4.1
Автор:	Роман Савоченко
Описание:	Предоставляет реализацию клиента SNMP-сервиса.
Лицензия:	GPL

Протокол SNMP был разработан с целью проверки функционирования сетевых маршрутизаторов и мостов в 1988 году. Впоследствии сфера действия протокола охватила и другие сетевые устройства, такие как хабы, шлюзы, терминальные сервера и даже устройства, имеющие отдалённое отношение к сети: принтера, источники бесперебойного питания, PLC и т.д. Кроме того, протокол допускает возможность внесения изменений в функционирование указанных устройств. На данное время протокол SNMP стандартизирован как RFC-1157, -1215, -1187, -1089.

Данный модуль предоставляет возможность собирать информацию у различных устройств по SNMP протоколу. Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

1. SNMP

Основными взаимодействующими «лицами» протокола являются агенты и системы управления. Если рассматривать эти два понятия на языке «клиент-сервер», то роль сервера выполняют агенты, то есть те самые устройства, для опроса состояния которых и был разработан протокол. Соответственно, роль клиентов отводится системам управления – сетевым приложениям, необходимым для сбора информации о функционировании агентов. Помимо этих двух субъектов в модели протокола можно выделить также еще два: управляющую информацию и сам протокол обмена данными.

Вся информация об объектах системы-агента содержится в так называемой MIB (management information base) – базе управляющей информации, другими словами MIB представляет собой совокупность объектов (MIB-переменные), доступных для операций записи-чтения.

На данный момент существует четыре базы MIB:

1. Internet MIB – база данных объектов для обеспечения диагностики ошибок и конфигураций. Включает в себя 171 объект (в том числе и объекты MIB I).
2. LAN manager MIB – база из 90 объектов – пароли, сессии, пользователи, общие ресурсы.
3. WINS MIB – база объектов, необходимых для функционирования WINS сервера.
4. DHCP MIB – база объектов, необходимых для функционирования DHCP сервера, служащего для динамического выделения IP адресов в сети.

1.1. MIB

Все имена MIB имеют иерархическую структуру. Существует десять корневых алиасов:

1. System — данная группа MIB II содержит в себе семь объектов, каждый из которых служит для хранения информации о системе (версия ОС, время работы и т.д.).
2. Interfaces — содержит 23 объекта, необходимых для ведения статистики сетевых интерфейсов агентов (количество интерфейсов, размер MTU, скорость передачи, физические адреса и т.д.).
3. AT (3 объекта) — отвечают за трансляцию адресов. Более не используется. Была включена

в MIB I. В SNMP v2 эта информация была перенесена в MIB для соответствующих протоколов.

4. IP (42 объекта) — данные о проходящих IP пакетах (количество запросов, ответов, отброшенных пакетов).
5. ICMP (26 объектов) — информация о контрольных сообщениях (входящие/исходящие сообщения, ошибки и т.д.).
6. TCP (19) — все, что касается одноименного транспортного протокола (алгоритмы, константы, соединения, открытые порты и т.п.).
7. UDP (6) — аналогично, только для UDP протокола (входящие/исходящие датаграммы, порты, ошибки).
8. EGP (20) — данные о трафике Exterior Gateway Protocol (используется маршрутизаторами, объекты хранят информацию о принятых/отосланных/отброшенных кадрах).
9. Transmission — зарезервирована для специфических MIB.
10. SNMP (29) — статистика по SNMP – входящие/исходящие пакеты, ограничения пакетов по размеру, ошибки, данные об обработанных запросах и многое другое.

1.2. Адресация

Каждый из корневых алиасов представляется в виде дерева, растущего вниз. Например, к адресу администратора можно обратиться посредством пути: system.sysContact.0, ко времени работы системы system.sysUpTime.0, к описанию системы (версия, ядро и другая информация об ОС) : system.sysDescr.0. С другой стороны, те же данные могут задаваться и в точечной нотации. Так system.sysUpTime.0 соответствует значение 1.3.0, так как system имеет индекс “1” в группах MIB II, а sysUpTime – 3 в иерархии группы system. Ноль в конце пути говорит о скалярном типе хранимых данных. В процессе работы символьные имена объектов не используются, то есть если менеджер запрашивает у агента содержимое параметра system.sysDescr.0, то в строке запроса ссылка на объект будет преобразована в “1.1.0”, а не будет передана «как есть».

В общем, существует несколько способов записи адреса к MIB-переменной:

- 1 Прямая кодовая адресация – “.1.3.6.1.2.1.1” (корневой алиас System). При такой адресации каждая MIB переменная кодируется идентификатором, а полный адрес записывается в виде последовательности идентификаторов разделённых точкой, слева на право. Данная запись адреса является основной и все другие способы записи приводятся к ней.
- 2 Полная символьная, в соответствии с предыдущей кодовой – “.iso.org.dod.internet.mgmt.mib-2.system”.
- 3 Адресация от корневого алиаса – “system.sysDescr”. 4 Адресация от базы MIB – «SNMPv2-MIB::sysDescr».

1.3. Взаимодействие

В SNMP клиент взаимодействует с сервером по принципу запрос-ответ. Сам по себе агент способен инициировать только одно действие, называемое ловушкой прерыванием. Помимо этого, все действия агентов сводятся к ответам на запросы, посылаемые менеджерами.

Существует 3 основные версии протокола SNMP (v1 & v2 & v3), которые не являются совместимыми. В SNMP v3 включена поддержка шифрования трафика, для чего, в зависимости от реализации, используются алгоритмы DES, MD5. Это ведет к тому, что при передаче наиболее важные данные недоступны для прослушивания. В качестве транспортного протокола в SNMP обычно используется протокол UDP, Хотя, на самом деле, SNMP поддерживает множество других транспортных протоколов нижнего уровня.

1.4. Авторизация

Одним из ключевых понятий в SNMP является понятие group (группа). Процедура авторизации менеджера представляет собой простую проверку на принадлежность его к определенной группе, из списка, находящегося у агента. Если агент не находит группы менеджера в своем списке, их

дальнейшее взаимодействие невозможно. По умолчанию используются группы: private и public.

2. Модуль

Данный модуль поддерживает работу с протоколом SNMP версии 1. На текущий момент поддерживается только чтение MIB-параметров. Кроме того, перечень типов MIB-параметров ограничен списком: ASN_OCTET_STR, ASN_INTEGER и ASN_COUNTER. Поддержка остальных типов и запись планируется в следующих версиях модуля.

2.1. Контроллер данных

Для добавления источника данных SNMP создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

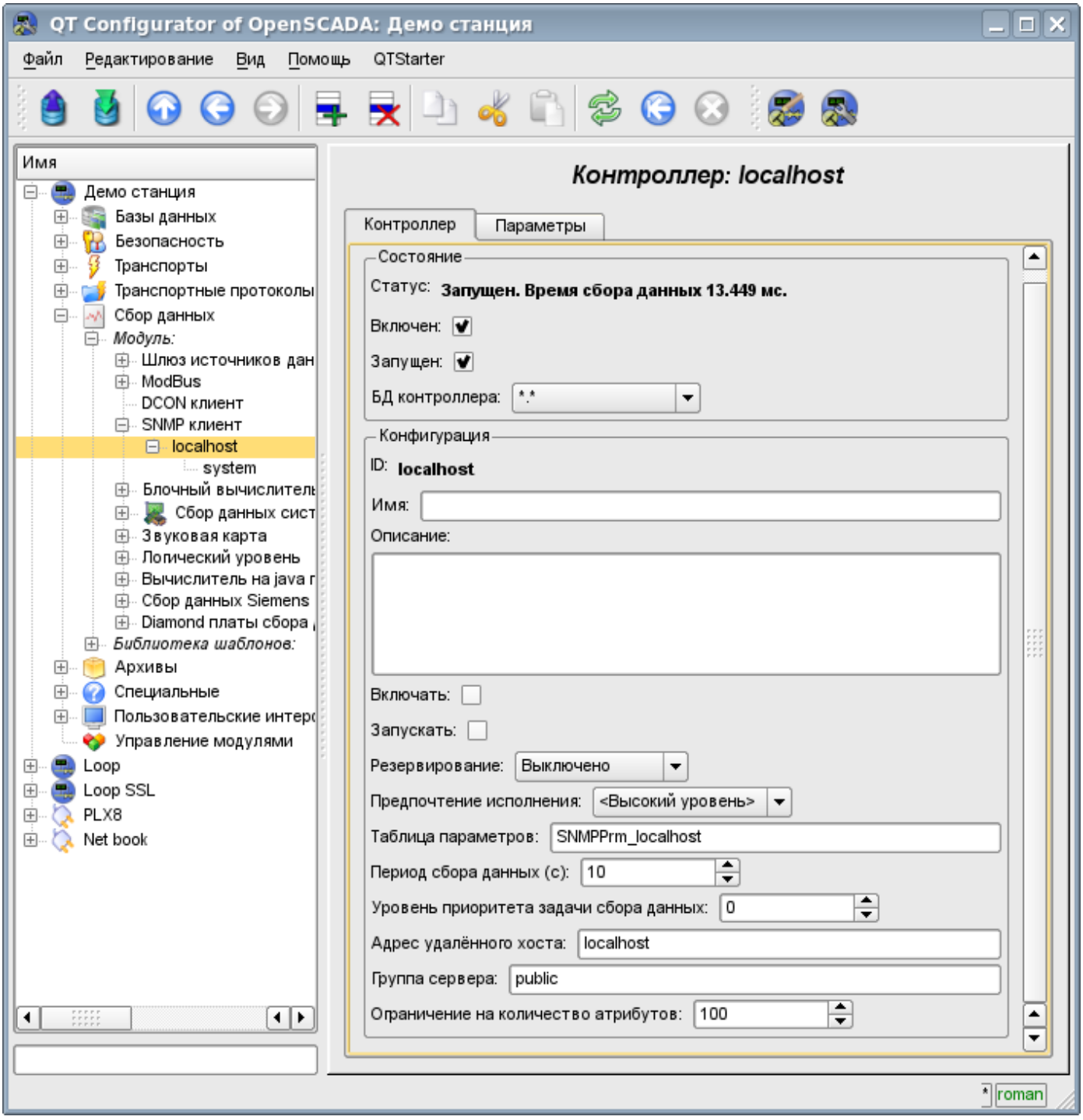


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи сбора данных.
- Адрес удалённого хоста, группу доступа и ограничение на количество атрибутов в одном параметре.

2.2. Параметры

Модуль *SNMP* предоставляет только один тип параметров – “Стандарт”. Дополнительным конфигурационным полем параметра данного модуля (рис.2) является список MIB-параметров, ветки которых подлежат считыванию.

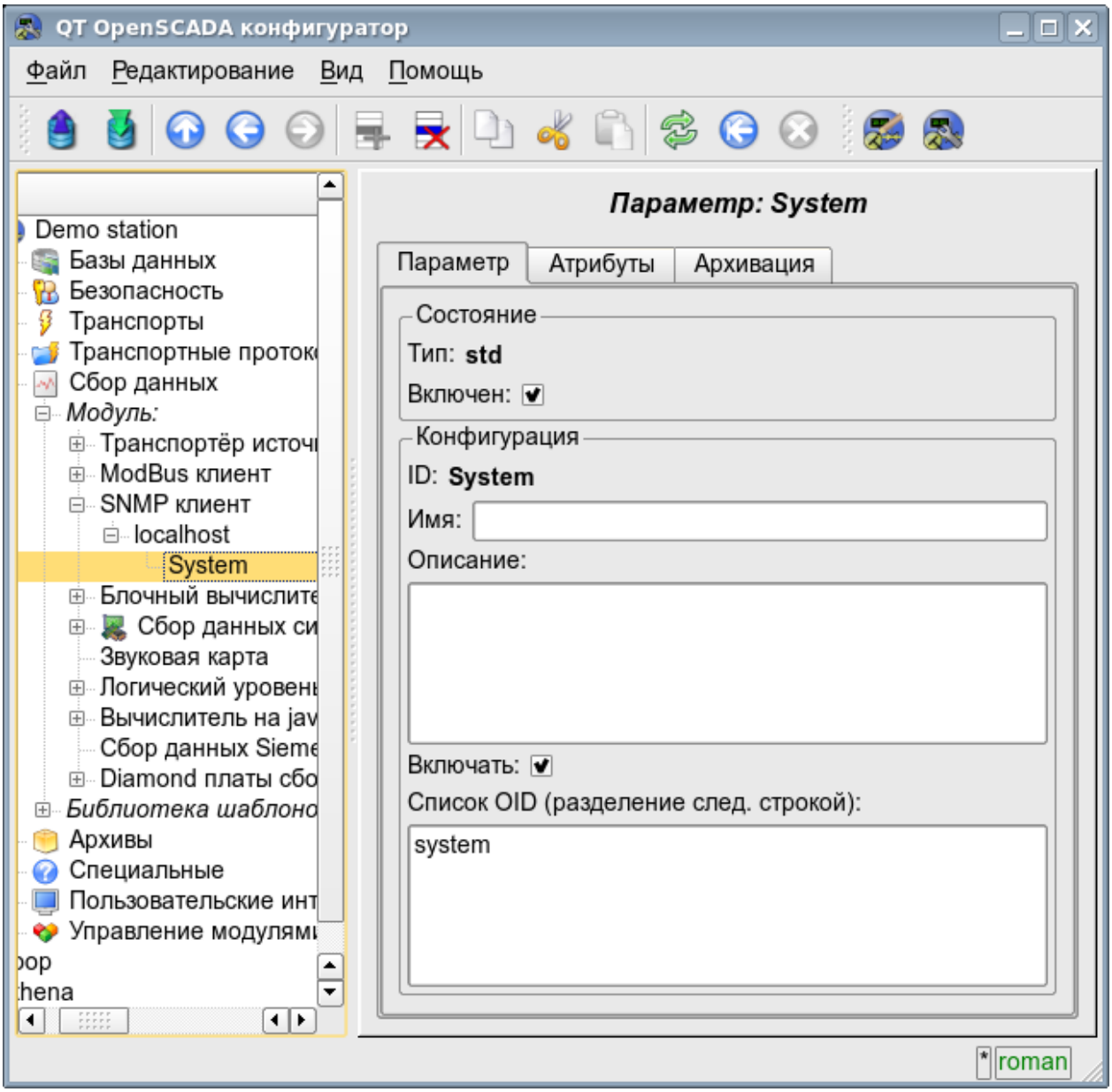


Рис.2. Вкладка конфигурации параметра.

В соответствии с указанным списком MIB-параметров выполняется опрос их ветвей и создание атрибутов параметра. В дальнейшем выполняется обновление значений параметров. Атрибуты именуются в соответствии с кодовой адресацией MIB-параметров, в качестве идентификатора, и адресации от базы MIB объектов, в имени атрибута (рис.3).

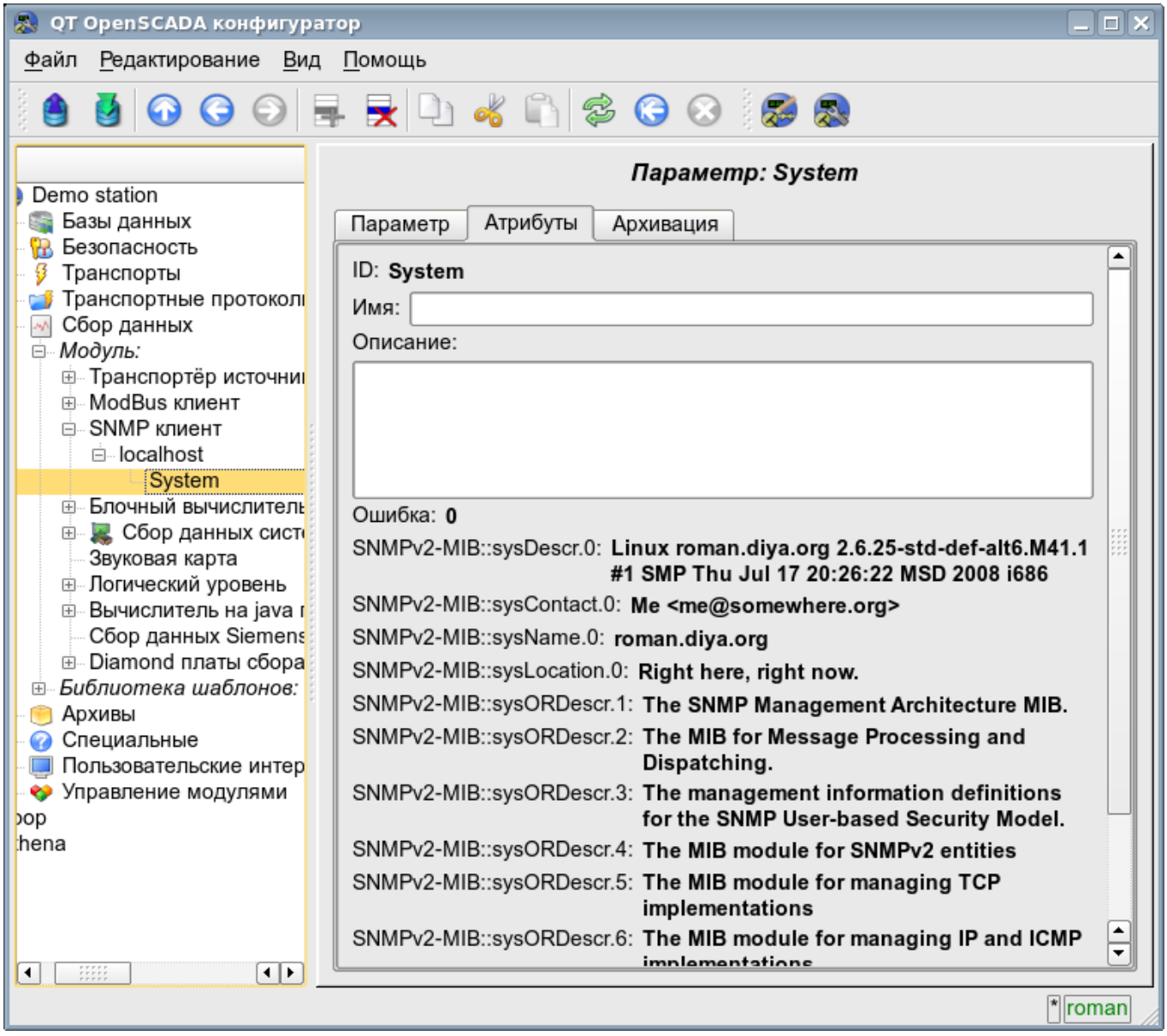


Рис.3. Вкладка атрибутов параметра.

Модуль подсистемы “Сбор данных” <Siemens>

Модуль:	Siemens
Имя:	Siemens DAQ
Тип:	DAQ
Источник:	daq_Siemens.so
Версия:	1.2.3
Автор:	Роман Савоченко
Описание:	Предоставляет источник данных ПЛК Siemens посредством карт Hilscher CIF с использованием протокола MPI и библиотеки Libnodave для остального.
Лицензия:	GPL

Первоочередной целью создания модуля является обеспечение поддержки промышленных контроллеров фирмы Siemens серии S7 (S7–300, S7–400). Исторически сложилось, что доступ к контроллерам указанной фирмы в сети Profibus производился только посредством собственных коммуникационных процессоров (CP5412, CP5613 и т.д.) и собственного протокола S7. Указанные коммуникационные процессоры и API к протоколу S7 достаточно дорогостоящие, кроме того драйвера к коммуникационным процессорам и S7 API закрыты и доступны только для платформы Intel+Windows (встречалась информация о возможности купить для Linux).

В качестве альтернативы данным решением от фирмы Siemens, которое позволяет полноценно работать с контроллерами фирмы Siemens, является спектр коммуникационных продуктов фирмы Hilscher (http://hilscher.com) в лице коммуникационных процессоров CIF серии PB(Profibus) и библиотека Libnodave(http://libnodave.sourceforge.net).

Особенностью продуктов Hilscher является полностью открытая спецификация протокола обмена с коммуникационным процессором, унифицированный драйвер для всех плат CIF, наличие драйвера для многих распространённых операционных систем(ОС) и отрытость драйвера для ОС Linux (GPL).

В основу данного модуля положен драйвер версии 2.620 фирмы Hilsher, любезно предоставленный фирмой Hilsher в лице Devid Tsaava для ядер серии 2.6 ОС Linux. Все необходимые для сборки файлы включены в модуль и не требуется удовлетворения ни каких специальных зависимостей. Драйвер версии 2.620 для плат CIF можно загрузить здесь.

Спектр плат фирмы Hilsher семейства CIF и унифицированный драйвер поддерживают широчайший спектр оборудования. Заложить поддержку всех этих возможностей в данном модуле, не имея всего этого оборудования на руках, не представляется возможным. Поэтому поддержка того или иного оборудования будет добавляться по потребности и наличию оборудования. По состоянию на версию 1.1.0 модулем предоставляется поддержка источников данных на сети Profibus или MPI посредством протокола MPI на скоростях сети от 9600Бод до 12МБод. В частности, поддерживаются и выполнена проверка на контроллерах фирмы Siemens семейства S7 (S7–300, S7–400).

Библиотека Libnodave является реализацией путём реверсинжениринга протоколов MPI, S7, ISO-TSAP и других, которые используются при взаимодействии с контроллерами фирмы Siemens. Библиотекой поддерживаются многие MPI и USB адаптеры, а также ProfiNet. Коммуникационные процессоры фирмы Siemens, на платформах отличных от Windows, библиотекой не поддерживаются. На данном этапе модулем обеспечена поддержка протокола ISO-TSAP(ProfiNet) посредством библиотеки Libnodave. Библиотека Libnodave полностью включена в данный модуль и не требует разрешения особых зависимостей как при сборке, так и при исполнении.

Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня. Кроме синхронизации значений и архивов атрибутов параметров модулем осуществляется синхронизация значений вычислительных шаблонов, с целью безударного подхвата алгоритмов.

1. Коммуникационные контроллеры CIF

Драйвером плат семейства CIF поддерживается возможность установки до 4 CIF плат. С целью контроля за наличием плат в системе и возможности их конфигурации, модуль содержит форму контроля и конфигурации CIF-плат (рис.1).

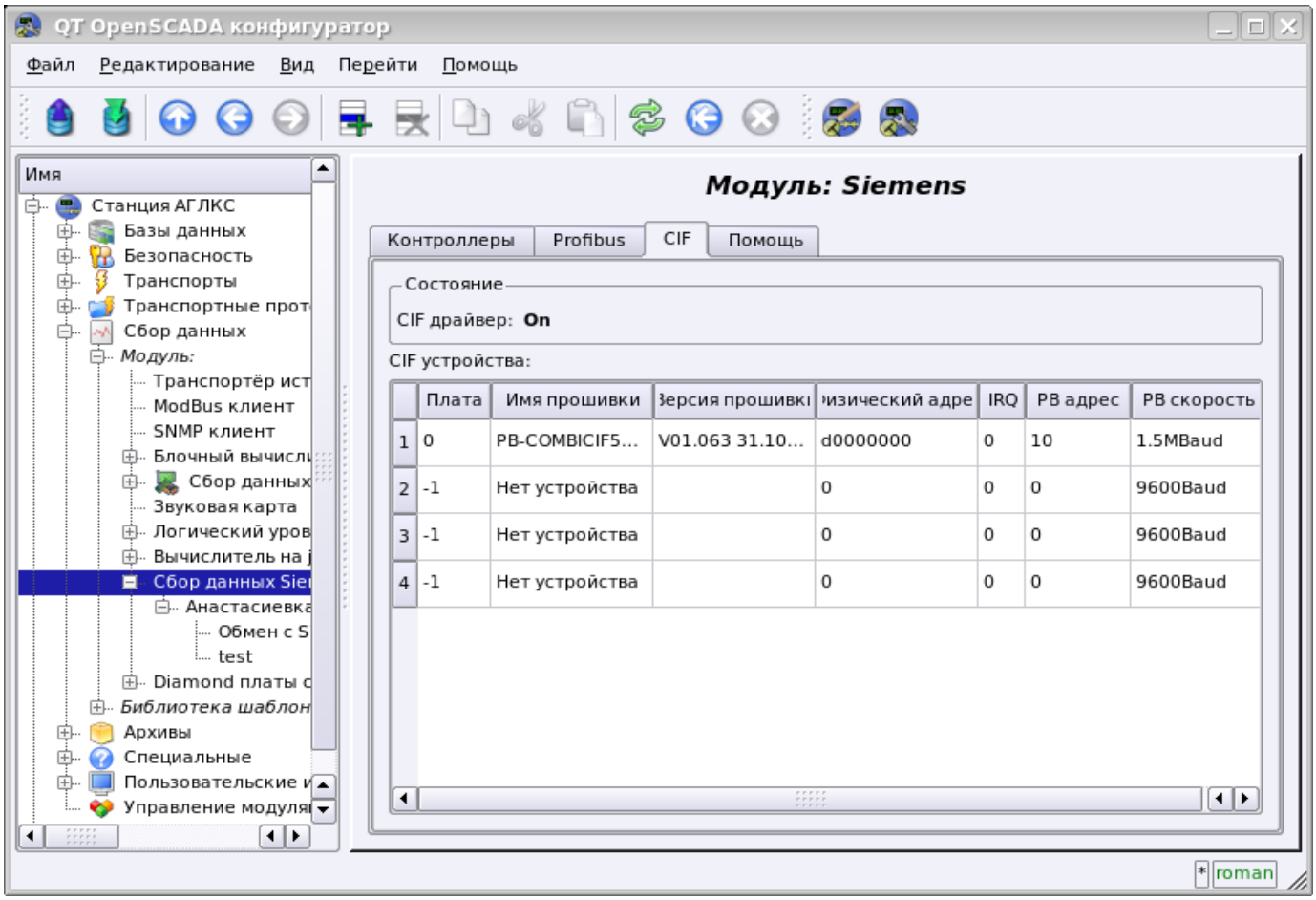


Рис.1. Вкладка конфигурации CIF-плат.

С помощью этой формы можно проконтролировать наличие коммуникационных процессоров, их конфигурацию, а также настроить параметры сети Profibus в виде PB адреса коммуникационного процессора и скорости шины Profibus. В другой вкладке модуля (рис.2) можно проконтролировать наличие различных станций в сети Profibus.

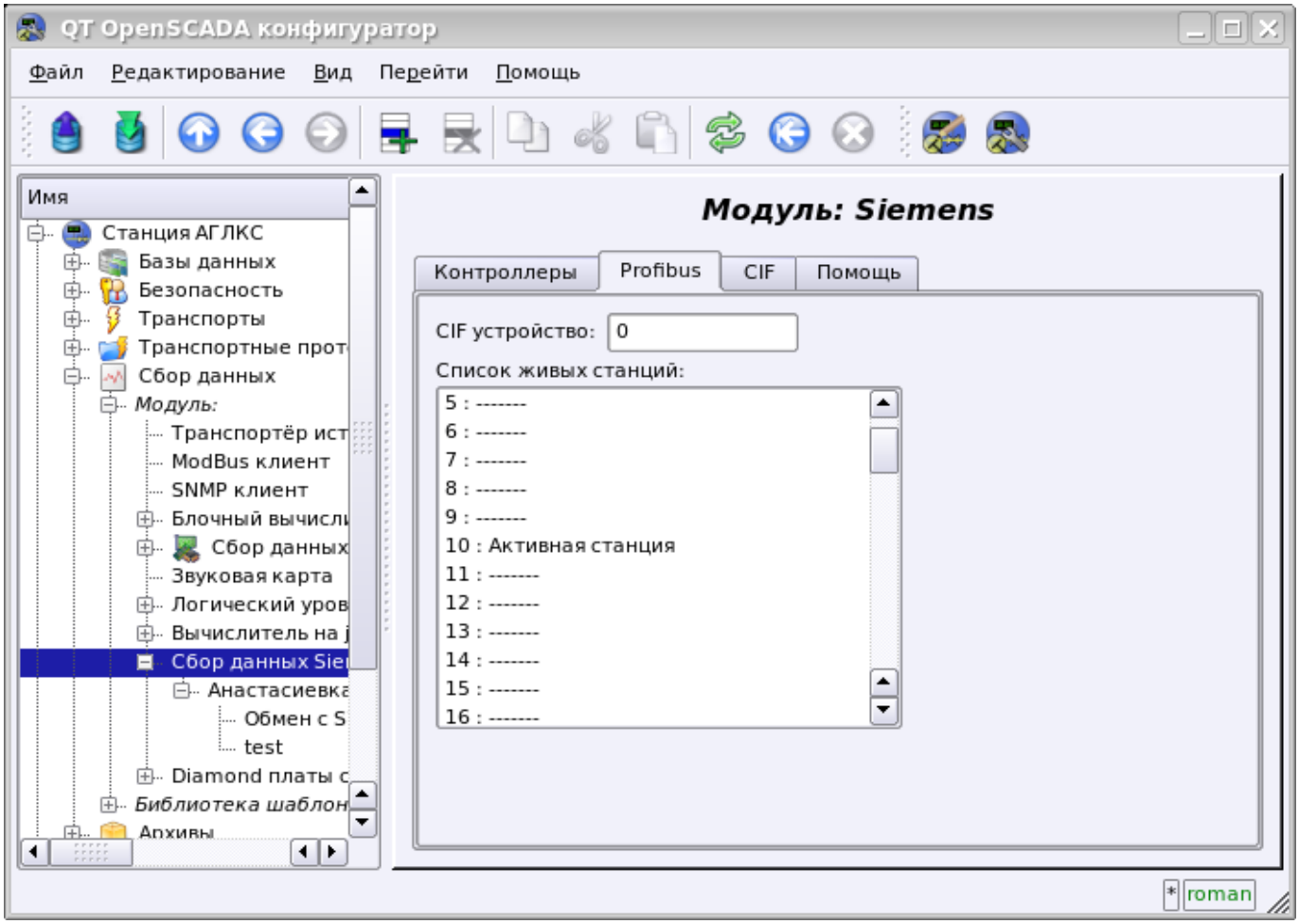


Рис.2. Вкладка мониторинга сети Profibus.

2. Контроллер источника данных

Для добавления источника данных создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.3.

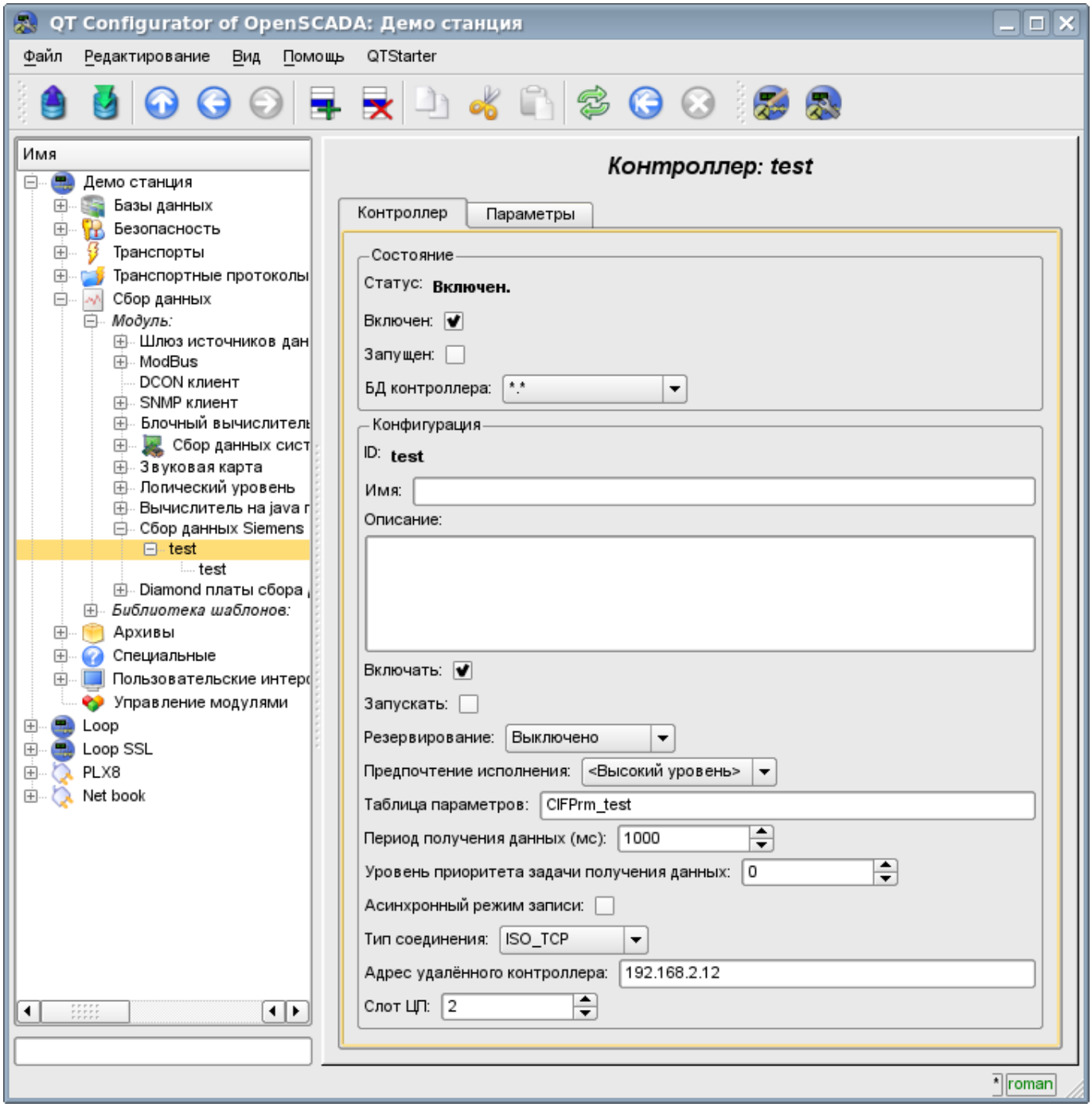


Рис.3. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи опроса источников данных.
- Режим асинхронной записи в удалённые контроллеры.
- Тип соединения. Поддерживаются CIF_PB и ISO_TCP

- Адрес промышленного контроллера. В случае типа соединения CIF это адрес в сети Profibus, а в случае ISO_TCP это IP-адрес в сети Ethernet.
- Слот ЦП в который установлен центральный процессор контроллера.
- CIF плату, используемую для доступа к промышленному контроллеру посредством CIF коммуникационных процессоров.

3. Параметры источника данных

Учитывая высокую интеллектуальность источников данных в лице промышленных контроллеров фирмы Siemens S7–300 и S7–400, параметры выполняются на основе шаблонов. Данный подход позволяет не ограничиваться жёстким перечнем типов параметров, что ограничивает и возможности контроллеров, а предоставить возможность пользователю формировать нужные ему типы параметров самостоятельно или использовать библиотеки уже разработанных ранее типов параметров (шаблонов).

Исходя из этого, модуль предоставляет только один тип параметров – «Логический». Дополнительными конфигурационными полями параметров данного модуля (рис.4) является поле выбора шаблона параметра.

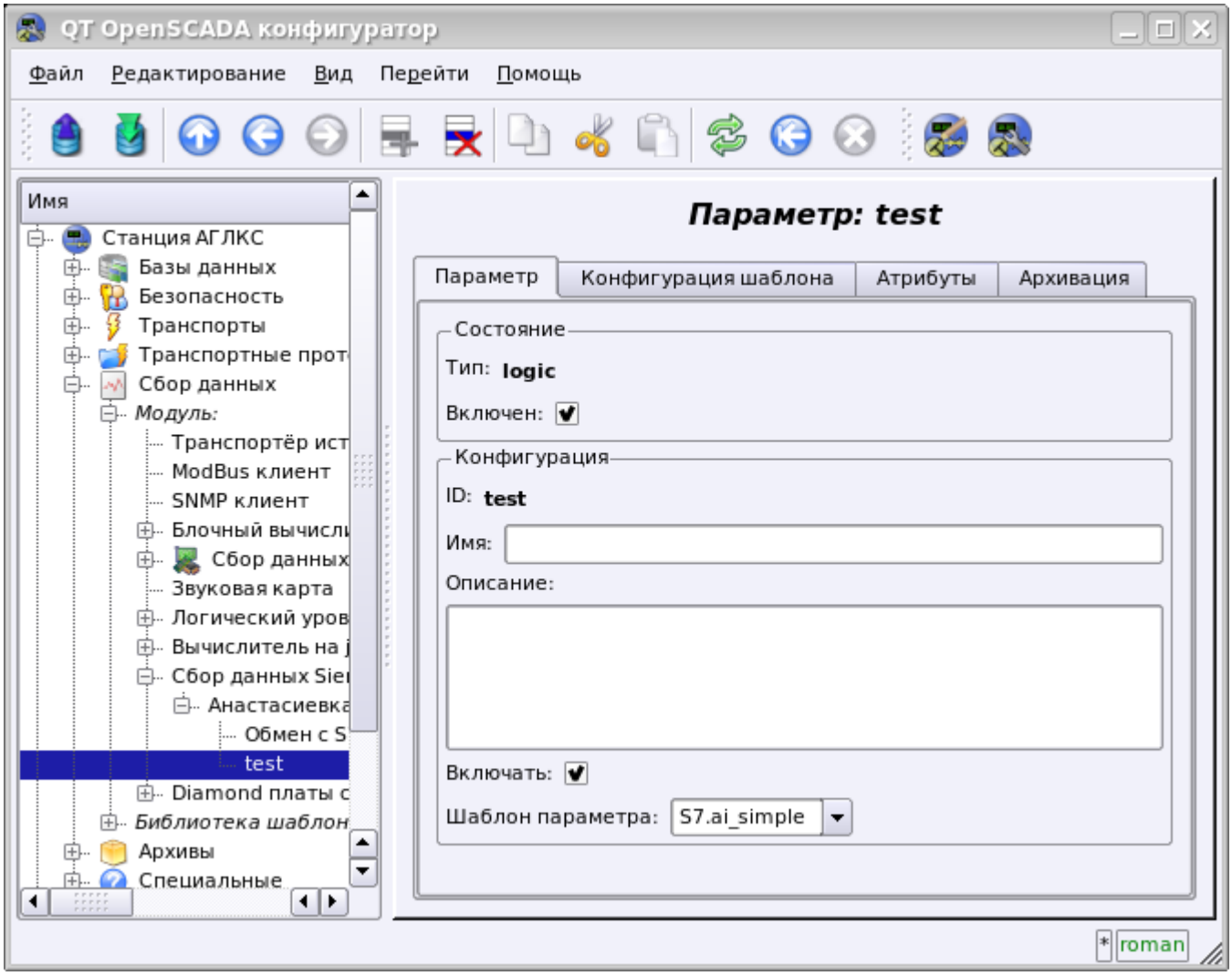


Рис.4. Вкладка конфигурации параметра.

Для конфигурации шаблона параметра предоставляется соответствующая вкладка. Содержимое этой вкладки определяется конфигурацией шаблона, т.е формируются соответствующие ссылочные поля и поля установки постоянных.

Типы ссылок определяются типом параметра в шаблоне (Логический, Целый, Вещественный и Строковый) и определением значения ссылки (для групповых ссылок). Определение групповой ссылки в шаблоне записывается в формате: "<Имя ссылки>|<Смещение в БД>|<Размер значения>",

где:

- *<Имя ссылки>* — Имя групповой ссылки. Все ссылки с одинаковым именем группируются и указываются как одна ссылка на БД или БД с указанным смещением.
- *<Смещение в БД>* — Имя смещения в блоке данных (БД). При указании только БД при конфигурации шаблона это смещение будет указанно для параметра, если же при конфигурации шаблона будет указано и смещение, то оба смещения будут суммироваться вместе. Такой подход позволяет обращаться к множеству структур в одном БД.
- *<Размер значения>* — Необязательное поле, которое определяет нестандартный размер значения в контроллере. Предусмотрены следующие размеры типов значений:
  - *Целое:* — 1 байт(знаковый), 2 байта(знаковый, по умолчанию) и 4 байта(знаковый).
  - *Вещественное:* — 4 байта(float, по умолчанию), 8 байт(double).
  - *Логический:* — всегда один байт (с указанием бита через точку – DB1.10.1).
  - *Строка:* — 10 байт(по умолчанию) и 1–200 можно указывать.

Наглядный пример общего процесса конфигурации параметра от шаблона и до значений приведен в рисунках от 5 до 8.

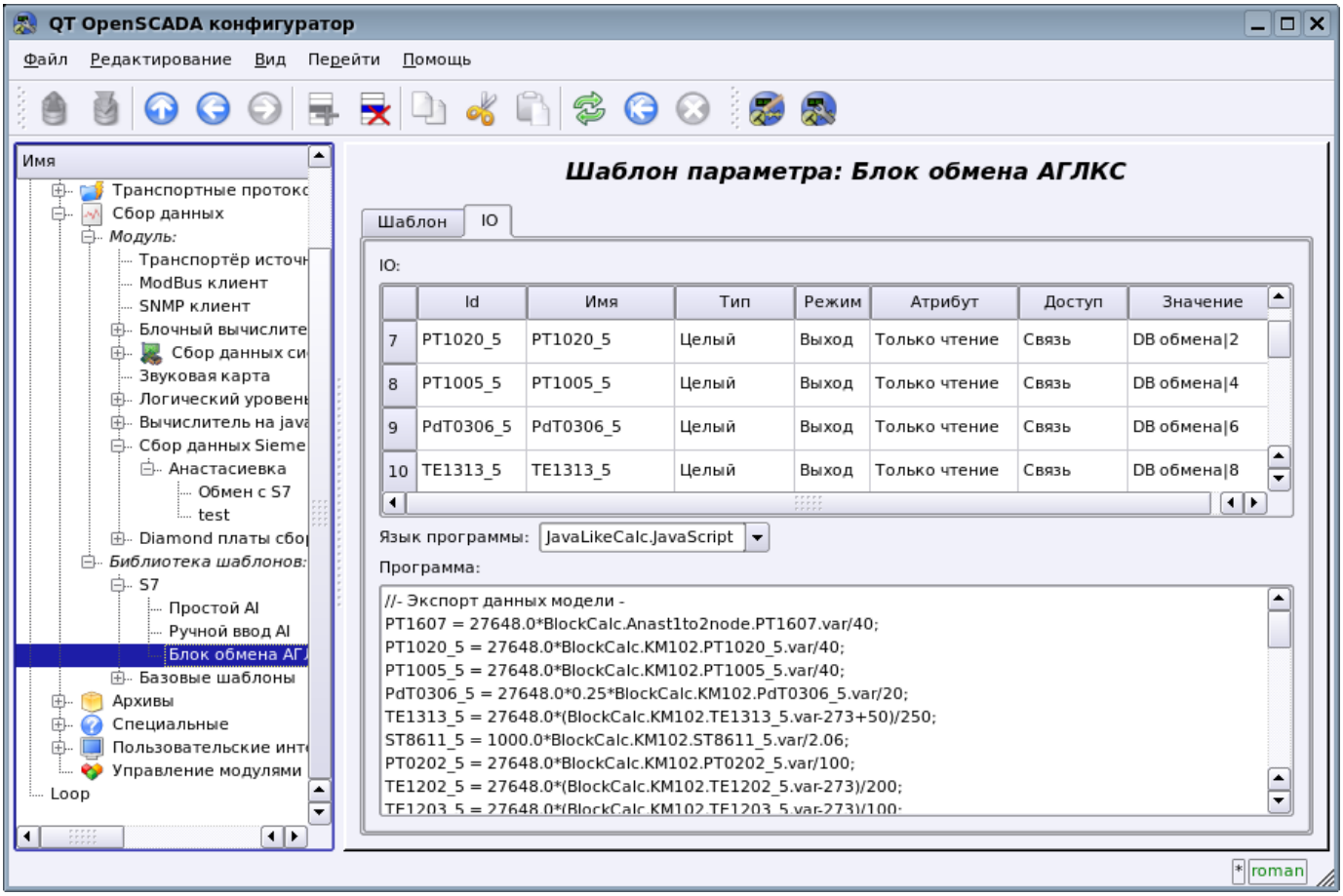


Рис.5. Пример шаблона с группированием.

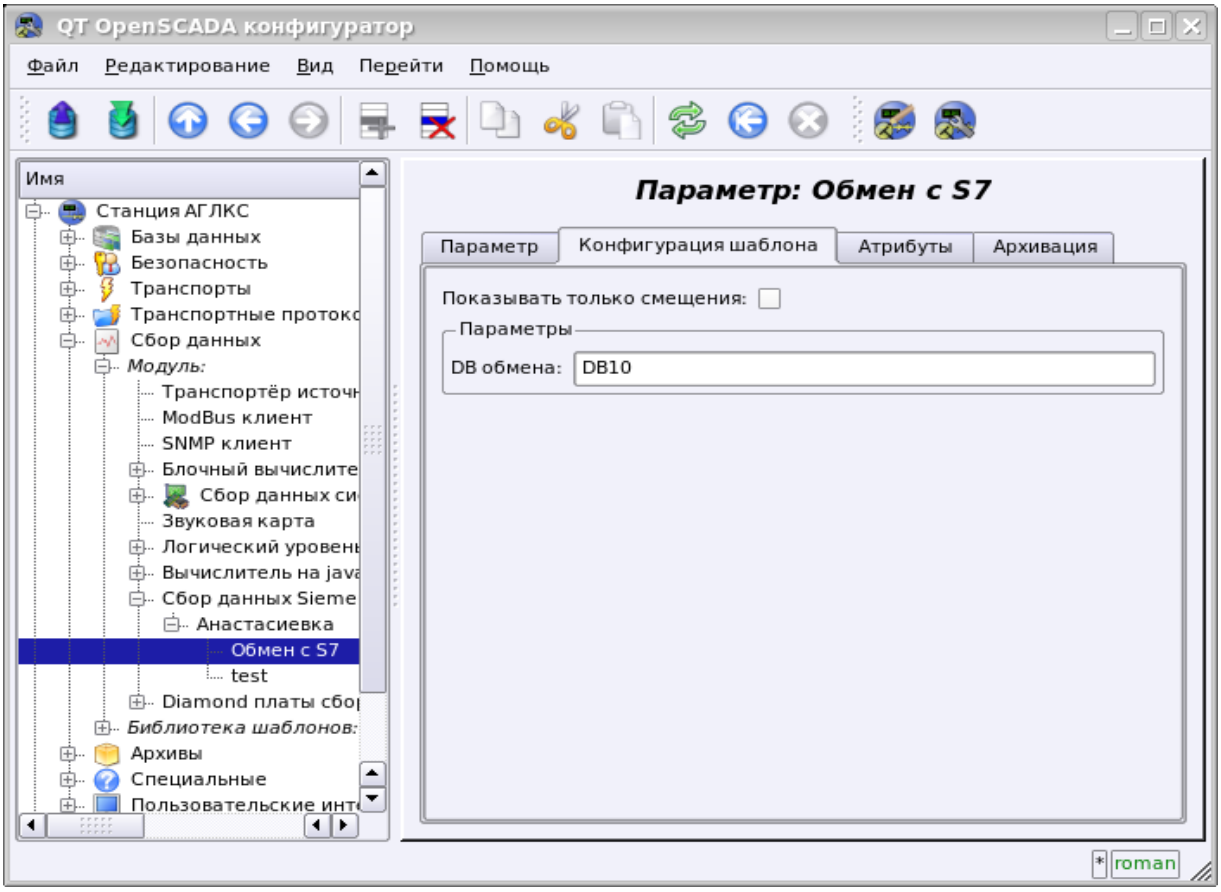


Рис.6. Вкладка конфигурации шаблона параметра

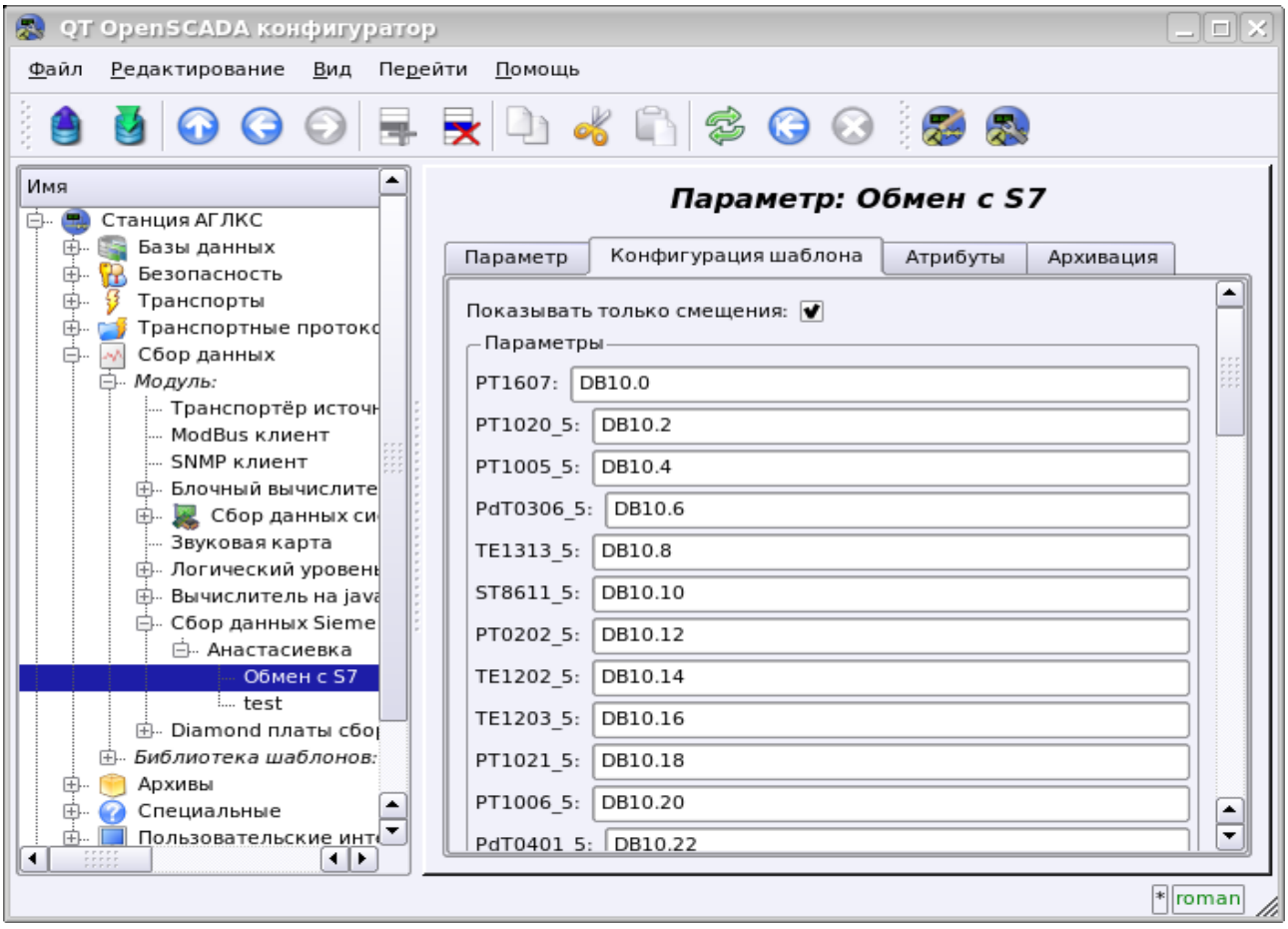


Рис.7. Вкладка конфигурации шаблона параметра с указанием параметров по отдельности.

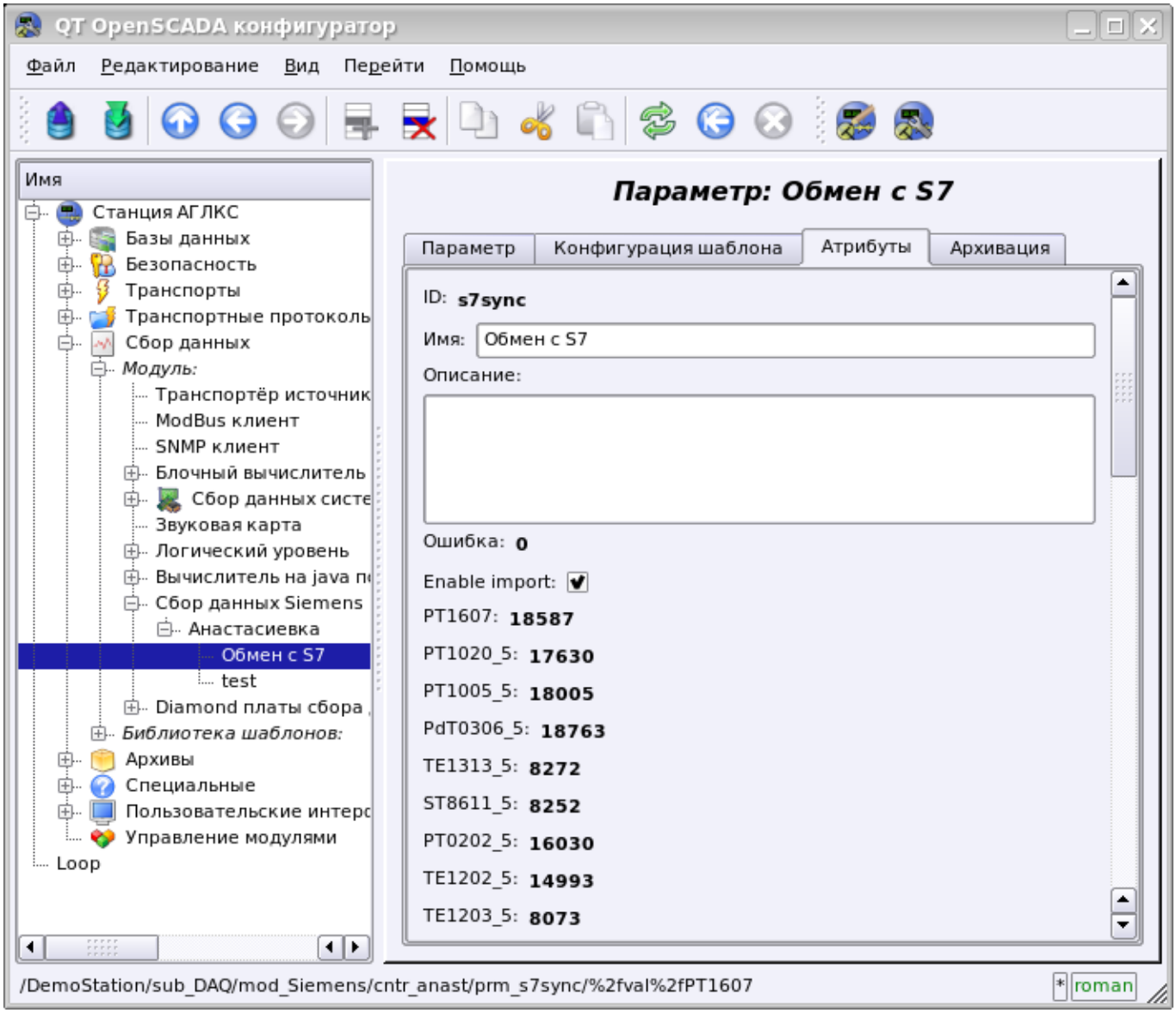


Рис.8. Значения параметра.

Модулем поддерживается адресация только к блокам данных (БД) контроллеров!

4. Асинхронный режим записи

Стандартным режимом записи для SCADA-систем, взаимодействующих с PLC, является синхронный, поскольку позволяет контролировать корректность завершения операции записи. Однако в случаях записи множества параметров сразу и часто такой подход не оправдан в виду отправки множества мелких запросов в контроллер, что перегружает PLC и занимает большой интервал времени. Решением этой проблемы является асинхронная запись смежных значений одним блоком. Такой режим поддерживается этим модулем и позволяет записывать все параметры сразу смежными блоками по 240байт. Чтение и запись в таком режиме производится смежными блоками с периодичностью опроса контроллера.

Модули <ModBus> подсистемы “Сбор данных” и подсистемы «Транспортные протоколы»

Параметр	Модуль 1	Модуль 2
ID:	ModBus	ModBus
Имя:	ModBus	ModBus
Тип:	DAQ	Протокол
Источник:	daq_ModBus.so	daq_ModBus.so
Версия:	1.1.1	0.6.1
Автор:	Роман Савоченко	Роман Савоченко
Описание:	Предоставляет реализацию клиентского сервиса протокола ModBus. Поддерживаются Modbus/TCP, Modbus/RTU и Modbus/ASCII протоколы.	Предоставляет реализацию протоколов ModBus. Поддерживаются Modbus/TCP, Modbus/RTU и Modbus/ASCII протоколы.
Лицензия:	GPL	GPL

ModBus — коммуникационный протокол, основанный на клиент-серверной архитектуре. Разработан фирмой Modicon для использования в контроллерах с программируемой логикой (PLC). Стал стандартом де-факто в промышленности и широко применяется для организации связи промышленного электронного оборудования. Использует для передачи данных через последовательные линии связи RS-485, RS-422, RS-232, а также сети TCP/IP. В настоящее время поддерживается некоммерческой организацией ModBus-IDA.

Существуют три режима протокола: ModBus/RTU, ModBus/ASCII и ModBus/TCP. Первые два используют последовательные линии связи (в основном RS-485, реже RS-422/RS-232), последний использует для передачи данных сети TCP/IP.

Модуль сбора данных предоставляет возможность собирать информацию у различных устройств по протоколу ModBus во всех режимах. Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня. В то же время модуль протокола позволяет сформировать и выдать данные по протоколу ModBus в различных режимах и через интерфейсы, поддерживаемые модулями подсистемы «Транспорты».

1. Общее описание протокола ModBus

Протокол ModBus/RTU предполагает одно ведущее (запрашивающее) устройство в линии (master), которое может передавать команды одному или нескольким ведомым устройствам (slave), обращаясь к ним по уникальному в линии адресу. Синтаксис команд протокола позволяет адресовать 247 устройств на одной линии связи стандарта RS-485 (реже RS-422 или RS-232). В случае с режимом TCP адресация исключена из протокола, поскольку выполняется на уровне TCP/IP стека.

Инициатива проведения обмена всегда исходит от ведущего устройства. Ведомые устройства прослушивают линию связи. Мастер подаёт запрос (посылка, последовательность байт) в линию и переходит в состояние прослушивания линии связи. Ведомое устройство отвечает на запрос, пришедший в его адрес. Окончание ответной посылки мастер определяет по временному интервалу между окончанием приёма предыдущего байта и началом приёма следующего. Если этот интервал превысил время, необходимое для приёма полтора байта на заданной скорости передачи, приём фрейма ответа считается завершённым. Фреймы запроса и ответа по протоколу ModBus имеют фиксированный формат.

1.1. Адресация

Все операции с данными привязаны к нулю, каждый вид данных (регистр, бит, регистр входа или бита входа) начинаются с адреса 0000 и заканчиваются 65535.

1.2. Стандартные коды функций

В протоколе ModBus можно выделить несколько подмножеств команд (Таблица 1).

Таблица 1: Подмножество команд протокола ModBus

Подмножество	**Диапазон кодов**
Стандартные	1–21
Резерв для расширенных функций	22–64
Пользовательские	65–119
Резерв для внутренних нужд	120–255

Модулем сбора данных используются команды 0x03 и 0x06 для чтения и записи регистров, 0x01 и 0x05 для чтения и записи битов, 0x02 и 0x04 для чтения регистра и бита входа соответственно.

Модуль протокола обрабатывает запросы командами 0x03 и 0x06 для чтения и записи регистров, 0x01 и 0x05 для чтения и записи битов.

2. Модуль реализации протокола

Модуль протокола ModBus содержит код реализации протокольной части ModBus, а именно особенности вариантов протоколов ModBus/TCP, ModBus/RTU и ModBus/ASCII. Модуль протокола, совместно с выбранным транспортом, активно используется модулем сбора данных для осуществления непосредственных запросов. Поскольку модуль протокола является автономным, то, используя его, можно создавать дополнительные модули сбора данных посредством нестандартных функций расширения ModBus различного оборудования автоматизации.

2.1. API функции исходящих запросов

API функции исходящих запросов оперируют обменом блоками PDU, завёрнутыми в XML-пакеты со следующей структурой:

<prt id="sId" reqTm="reqTm" node="node" reqTry="reqTry">[pdu]</prt>

Где:

- *prt* - имя тега запроса с названием используемого варианта протокола (TCP, RTU или ASCII).
- *sId* - идентификатор источника запроса. Используется для помещения в протокол выходного протокола.
- *reqTm* - время запроса, а именно время, в течение которого ожидать ответа.
- *node* - номер узла назначения или идентификатор юнита ModBus/TCP.
- *reqTry* - количество попыток повторения запроса с ошибкой в ответе. Только для вариантов ModBus/RTU и ModBus/ASCII.
- *pdu* - непосредственно блок юнита данных протокола (PDU) ModBus.

Результирующий pdu заменяет pdu запроса в XML-пакете, а также устанавливается атрибут "err" с кодом и текстом ошибки, если такова имела место.

2.2. Обслуживание запросов по протоколу ModBus

Входная часть обслуживания запросов к модулю протокола осуществляет проверку и обработку запросов посредством объектов узлов, предусмотренных модулем (рис.1). Фактически реализуется механизм, позволяющий выполнять системой OpenSCADA роль сервера ModBus/TCP или подчинённого устройства ModBus/RTU и ModBus/ASCII. Таким образом система OpenSCADA получает возможность использоваться в роли любого участника сетей ModBus.

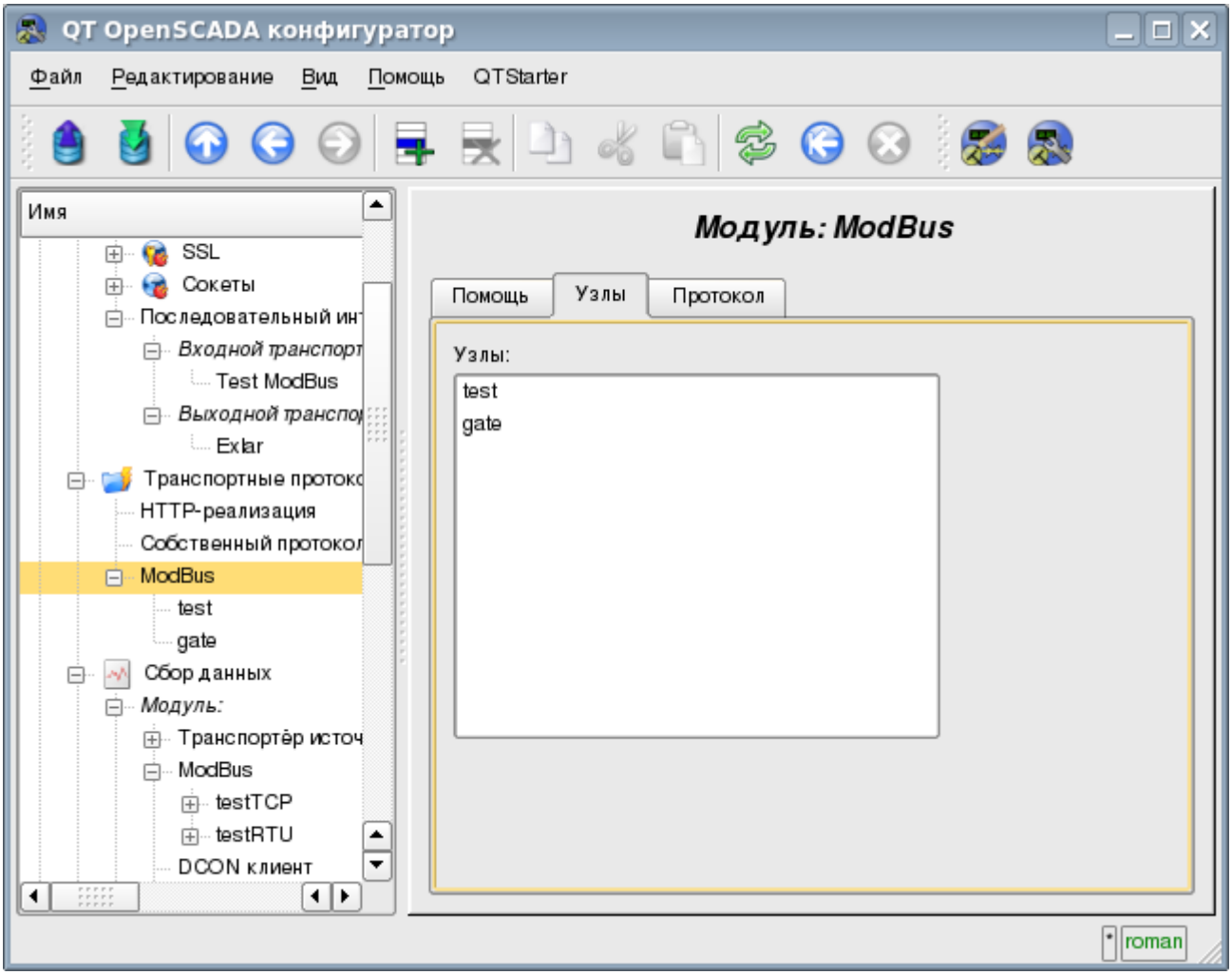


Рис.1. Вкладка перечня узлов обслуживания входящих запросов протокола.

Узел протокола эквивалентен физическому узлу устройства сети ModBus. Узел протокола может работать в трёх режимах:

- "Данные" - режим отражения данных системы OpenSCADA на массивы регистров и битов ModBus с передачей их по запросу клиентского узла или мастера.
- "Шлюз узла" - режим перенаправления (шлюзования) запросов к узлу в другой сети ModBus через данный узел.
- "Шлюз сети" - режим перенаправления запросов к любому узлу в другую сеть ModBus, фактически выполняя интеграцию нескольких сетей ModBus в одну.

Поскольку узлов протокола может быть создано множество, то получается, что на одном интерфейсе, т.е. в одной сети, одна станция на основе OpenSCADA может прозрачно представлять несколько узлов сети ModBus с различными данными.

Рассмотрим особенности конфигурации узла протокола в различных режимах.

Режим узла протокола «Данные»

Режим используется для отражения данных системы OpenSCADA на массивы регистров и битов ModBus. Общая конфигурация узла осуществляется во вкладке «Узел» (рис.2) параметрами:

- Состояние узла, а именно: статус, «Включен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание узла.
- Состояние, в которое переводить узел при загрузке: «Включен».
- Адрес узла сети ModBus от 1 до 247.
- Входящий транспорт, к сети которого относится узел. Выбирается из перечня входных транспортов подсистемы «Транспорты» OpenSCADA. Указание в качестве транспорта символа "*" делает данный узел участником любой сети ModBus с обработкой запросов от любого транспорта.
- Вариант протокола ModBus, запросы в котором должен обрабатывать узел из списка: Все, RTU, ASCII, TCP/IP.
- Выбор режима, в данном случае этот режим «Данные».
- Период обсчёта данных в секундах. Указывает периодичность обработки формируемых для запросов данных, а именно таблицы данных ModBus, программы обсчёта данных и обслуживание ссылок на данные OpenSCADA.

Узлом в этом режиме обрабатываются следующие стандартные команды протокола ModBus:

- *01* - чтение группы битов;
- *03* - чтение группы регистров;
- *05* - установка одного бита;
- *06* - установка одного регистра.

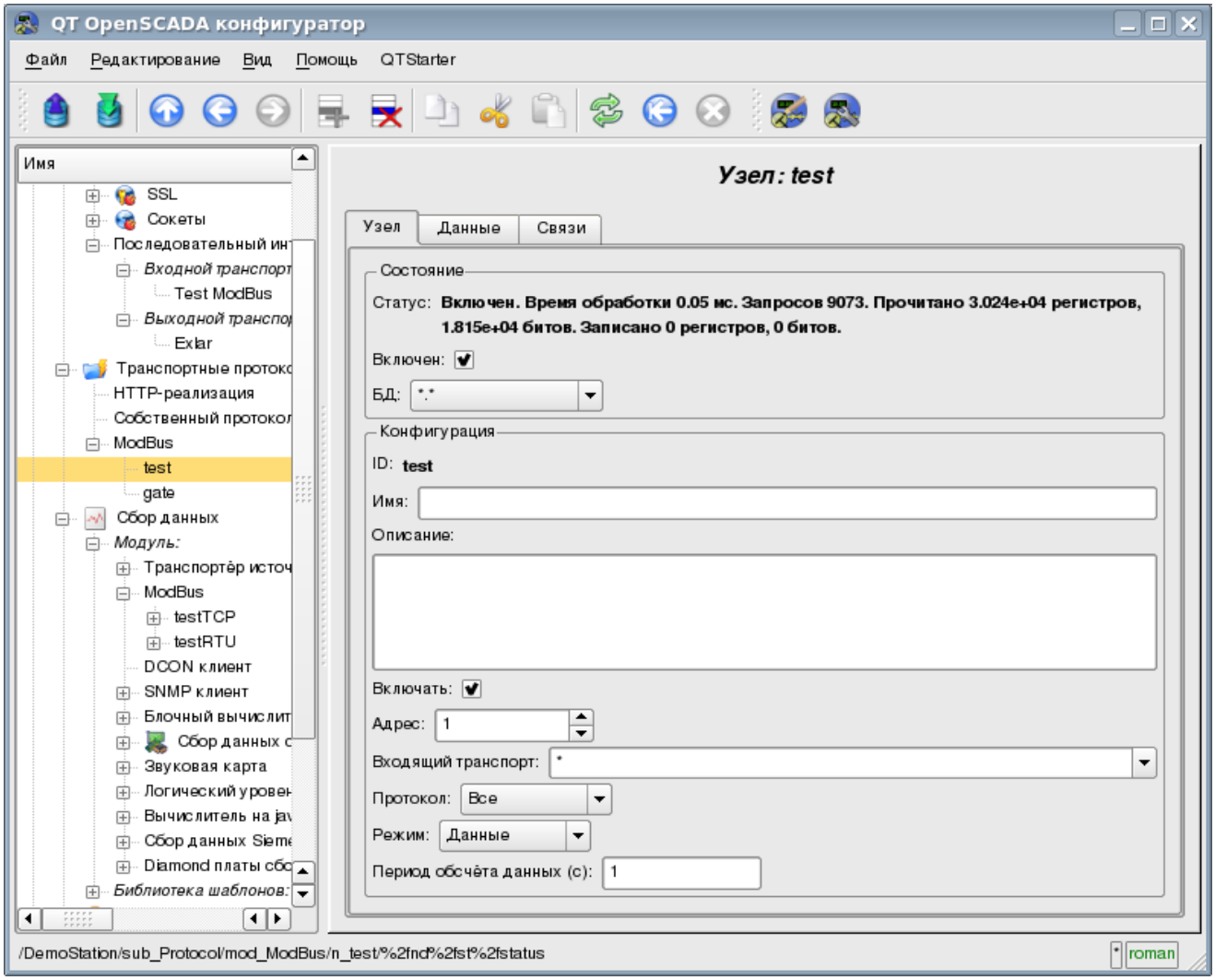


Рис.2. Вкладка «Узел» страницы конфигурации узла протокола в режиме «Данные».

Для формирования таблицы отражения данных сети ModBus, а именно регистров и битов предусматривается вкладка "Данные" (рис.3). Вкладка "Данные" содержит таблицу параметров и программу обработки параметров с указанным языком программирования доступным в системе OpenSCADA. Таблица содержит параметры со свойствами:

- *Id* - идентификатор параметра. Является ключевым для формирования таблиц регистров и битов ModBus. Для указания того, что данный параметр является регистром ModBus, необходимо идентификатор записать в виде "R[N]w", где N - число номера регистра от 0 до 65535, а w - необязательный символ, указывающий на возможность установки его запросом ModBus, например: R23, R456, R239w. Для указания бита ModBus необходимо идентификатор записать в виде "C[N]w", где N - число номера бита от 0 до 65535, а w - необязательный символ, указывающий на возможность установки его запросом ModBus, например: C437, C0, C39w. Все остальные параметры, не попавшие под вышеуказанные правила, являются внутренними и используются для различных промежуточных вычислений, обработки и преобразования.
- *Имя* - Имя параметра, используется для именования связи.
- *Tun* - Тип параметра из списка: "Вещественный", "Целый", "Логический" и "Строка". Для регистров и битов ModBus имеет смысл устанавливать "Целый" и "Логический" тип соответственно.
- *Связь* - Признак того, что данный параметр должен связываться с атрибутом параметра подсистемы "Сбор данных". Указанные этим флагом связи устанавливаются во вкладке "Связи".
- *Значение* - Исходное или текущее, если узел включен, значение параметра.

В таблице по умолчанию определяются несколько параметров специального значения:

- *f_frq* - частота вычисления таблицы программой;
- *f_start* - признак первого исполнения, запуска, программы.
- *f_stop* - признак последнего исполнения, останова, программы.

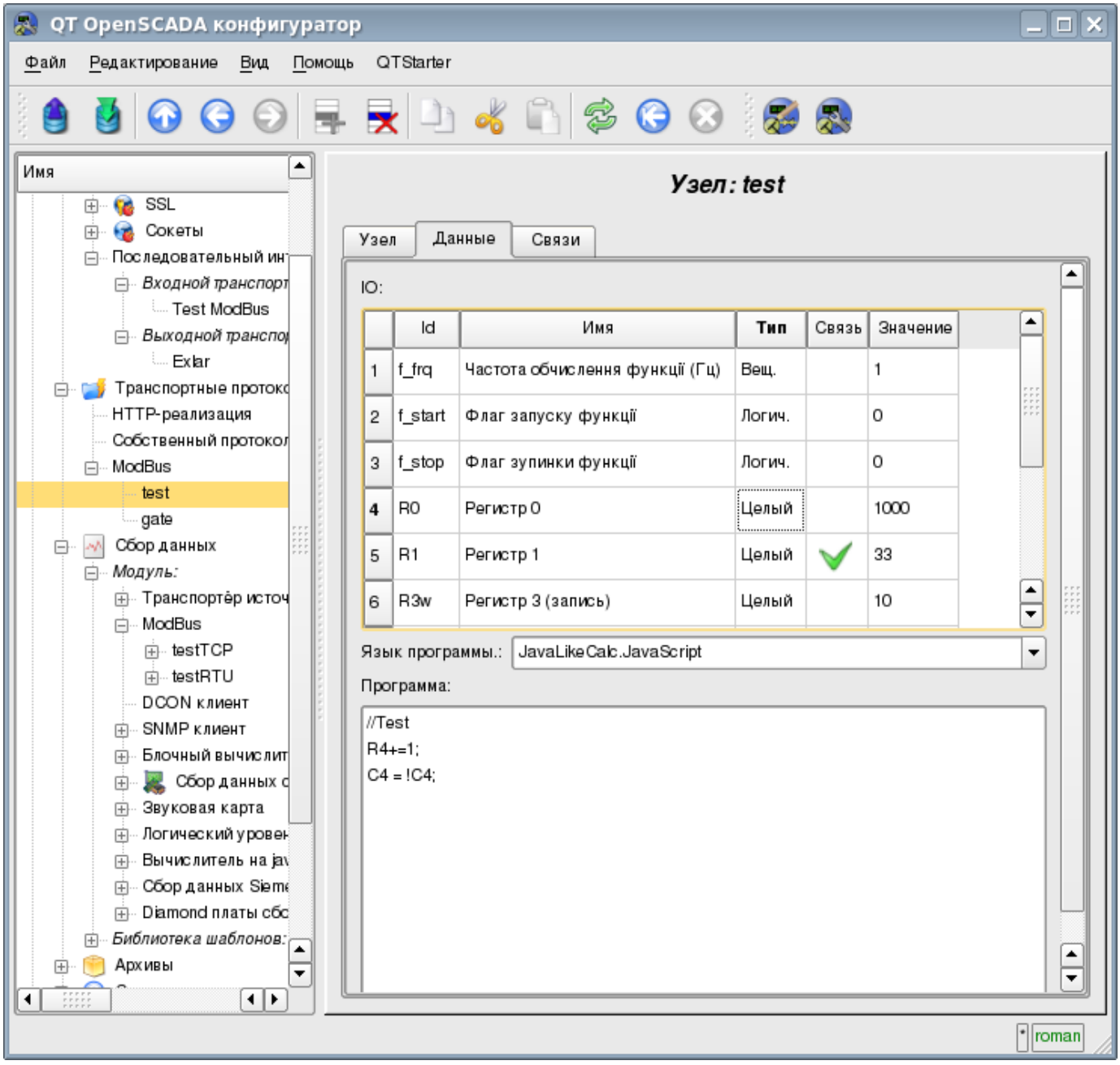


Рис.3. Вкладка «Данные» страницы конфигурации узла протокола в режиме «Данные».

Для указанных в качестве связей параметров можно установить связи только для выключенного узла протокола во вкладке «Связи» (рис.4).

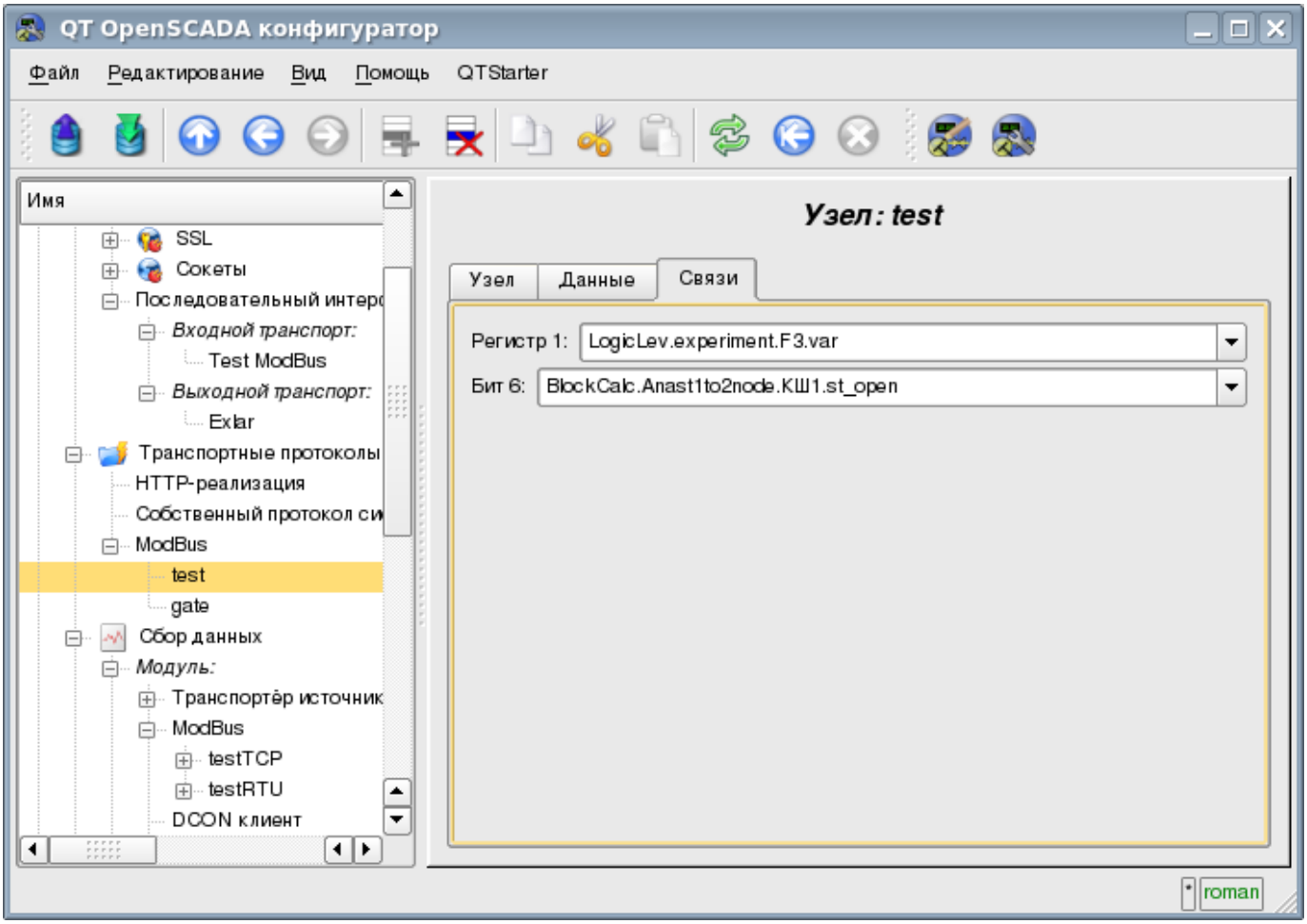


Рис.4. Вкладка «Связи» страницы конфигурации узла протокола в режиме «Данные».

Режим узла протокола «Шлюз узла»

Режим используется для проброса запросов к отдельному устройству в другой сети ModBus в сеть ModBus, для которой сконфигурирован данный узел. Общая конфигурация узла осуществляется во вкладке «Узел» (рис.5) параметрами:

- Состояние узла, а именно: статус, «Включен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание узла.
- Состояние, в которое переводить узел при загрузке: «Включен».
- Адрес узла сети ModBus от 1 до 247.
- Входящий транспорт, к сети которого относится узел. Выбирается из перечня входных транспортов подсистемы «Транспорты» OpenSCADA. Указание в качестве транспорта символа "*" делает данный узел участником любой сети ModBus с обработкой запросов от любого транспорта.
- Вариант протокола ModBus запросы в котором должен обрабатывать узел из списка: Все, RTU, ASCII, TCP/IP.
- Выбор режима, в данном случае этот режим «Шлюз узла».
- Транспорт, в который перенаправлять запрос, из перечня исходящих транспортов подсистемы «Транспорты».
- Протокол в котором перенаправлять запрос.
- Адрес узла сети ModBus, от 1 до 247, в которую перенаправляется запрос.

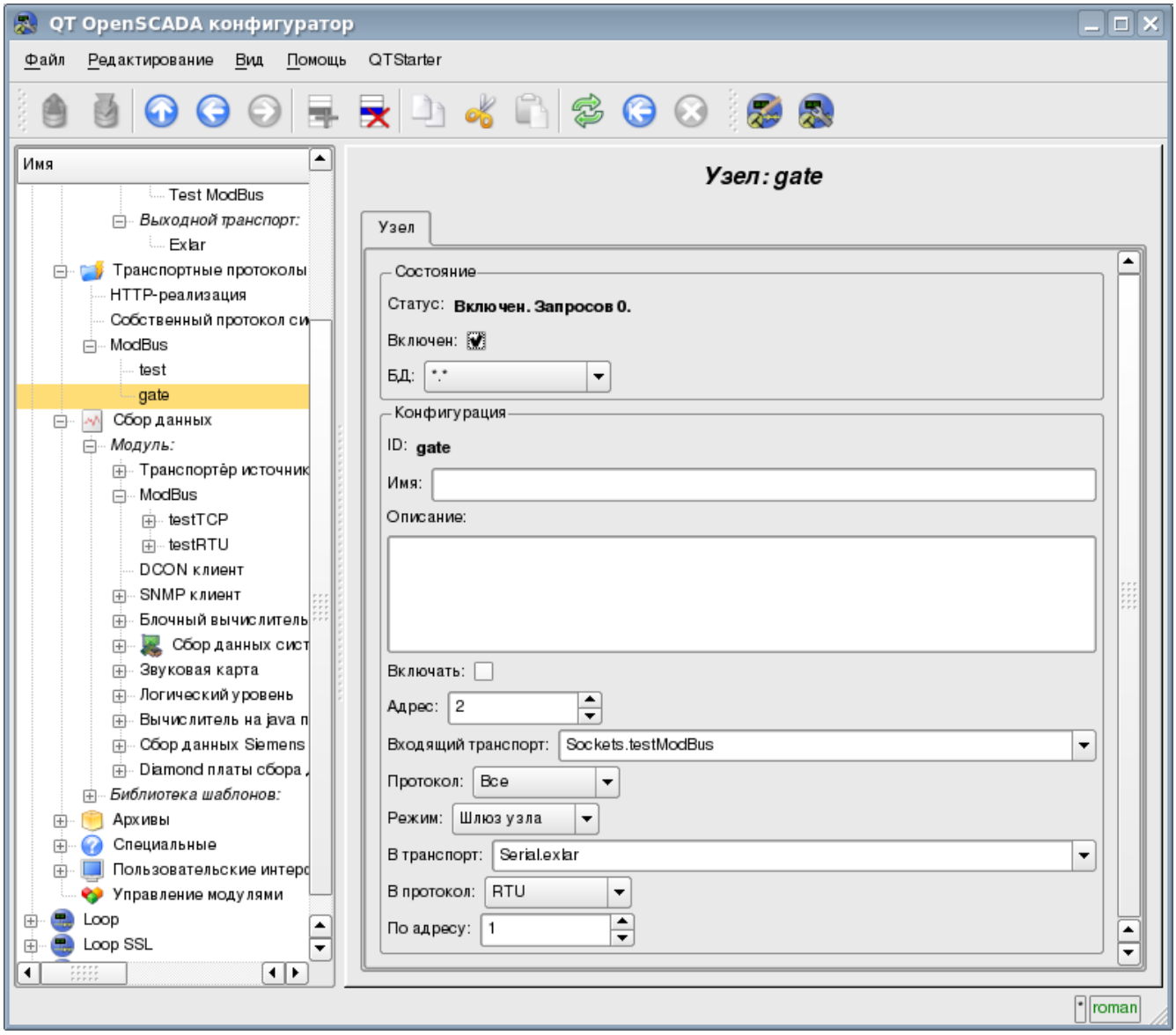


Рис.5. Вкладка «Узел» страницы конфигурации узла протокола в режиме «Шлюз узла».

Режим узла протокола «Шлюз сети»

Режим используется для проброса запросов сети целиком в другую сеть ModBus из сети ModBus для которой сконфигурирован данный узел протокола. Общая конфигурация узла протокола осуществляется во вкладке «Узел» (рис.6) параметрами:

- Состояние узла, а именно: статус, «Включен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание узла.
- Состояние, в которое переводить узел при загрузке: «Включен».
- Входящий транспорт сети, из которой пробрасываются запросы. Выбирается из перечня входных транспортов подсистемы «Транспорты» OpenSCADA.
- Вариант протокола ModBus, запросы в котором должен обрабатывать узел из списка: Все, RTU, ASCII, TCP/IP.
- Выбор режима, в данном случае этот режим «Шлюз сети».
- Транспорт сети, в которую перенаправлять запрос, из перечня исходящих транспортов подсистемы «Транспорты».
- Протокол в котором перенаправлять запрос.

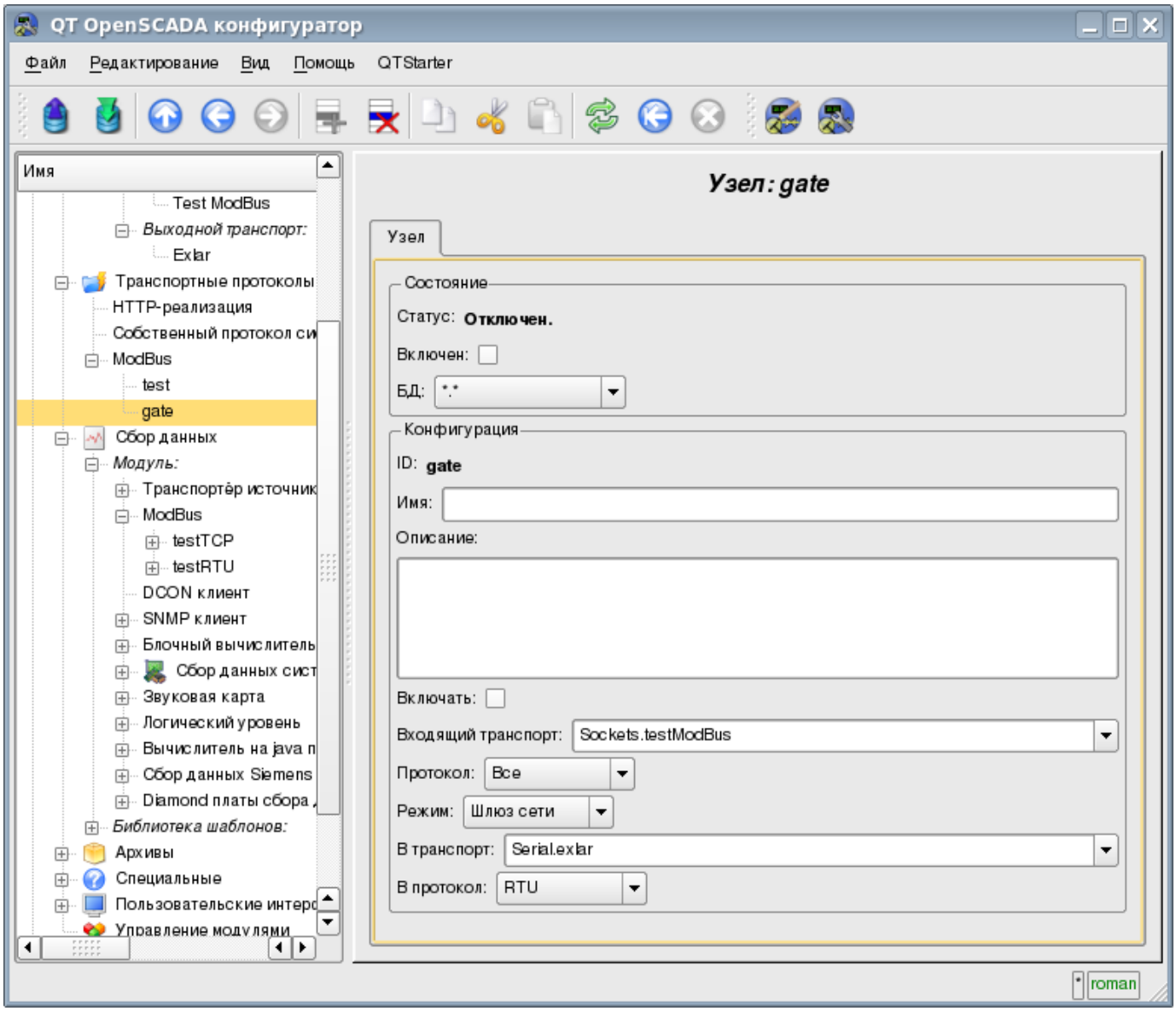


Рис.6. Вкладка «Узел» страницы конфигурации узла протокола в режиме «Шлюз сети».

2.3 Отчёт запросов ModBus

Для возможности контроля за корректностью осуществления запросов различными компонентами модулем предоставляется возможность включения отчёта запросов, проходящих через модуль протокола. Отчёт включается указанием не нулевого количества записей во вкладе «Отчёт» страницы модуля протокола (рис.7).

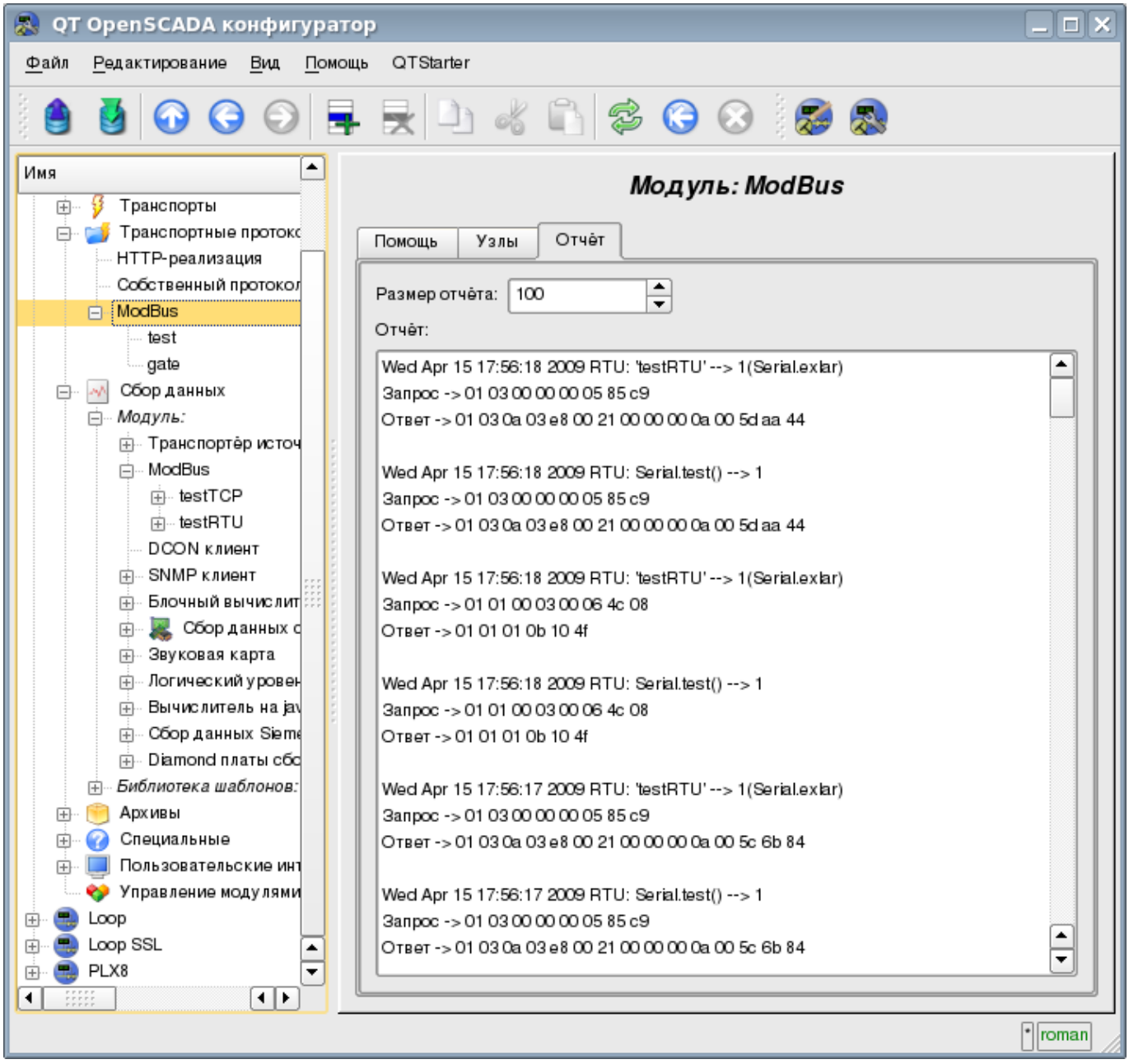


Рис.7. Вклада «Отчёт» страницы модуля протокола.

3. Модуль сбора данных

Модуль сбора данных предоставляет возможность опроса и записи регистров и битов устройств посредством режима протоколов TCP, RTU, ASCII и команд запроса 0x01 - 0x06.

3.1. Контроллер данных

Для добавления источника данных ModBus создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.8.

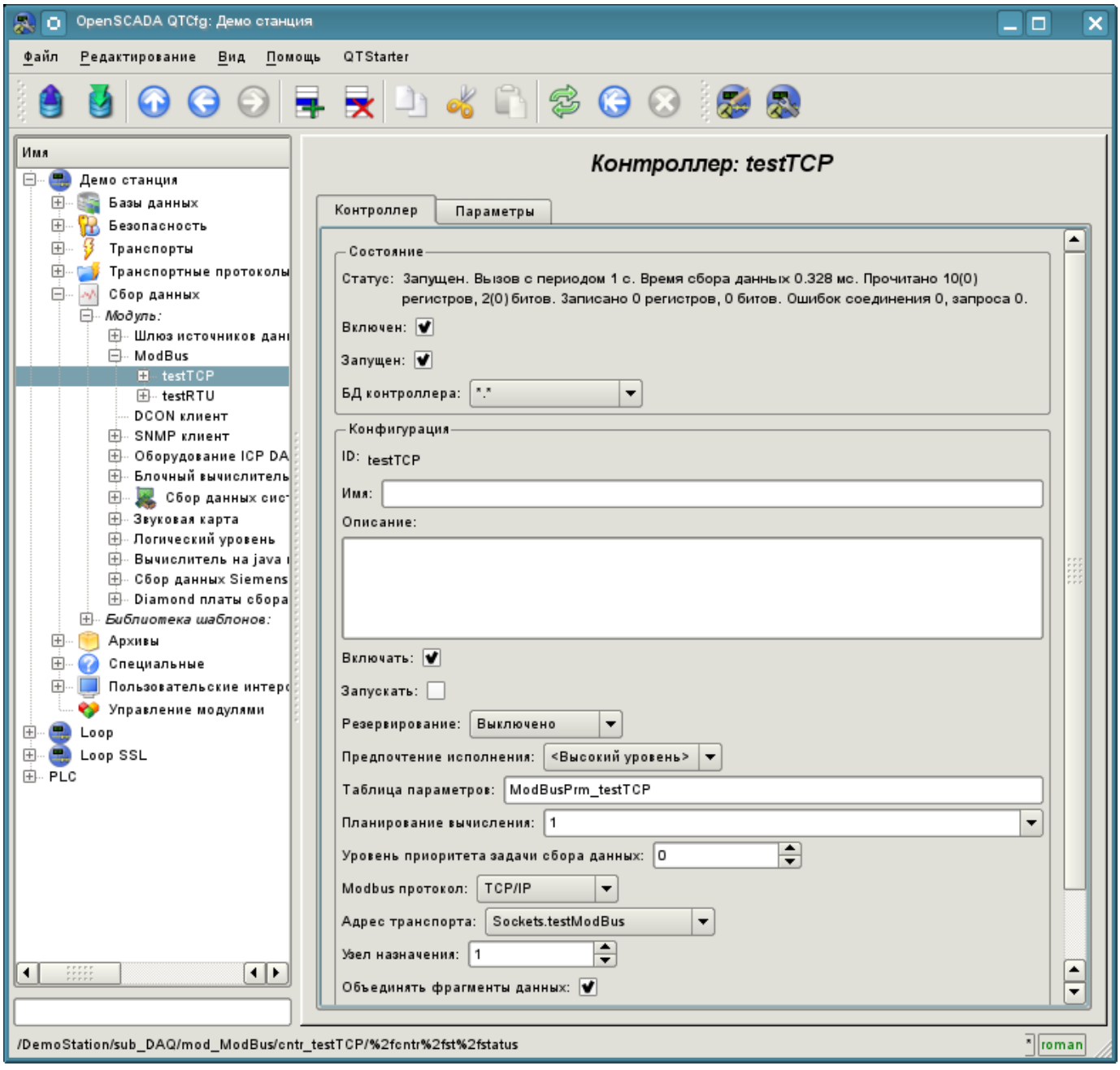


Рис.8. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.

- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи сбора данных.
- ModBus протокол, используемый для опроса физического устройства (TCP/IP, RTU или ASCII).
- Адрес исходящего транспорта из списка сконфигурированных исходящих транспортов в подсистеме «Транспорты» OpenSCADA.
- ModBus узел назначения. В случае с протоколами RTU и ASCII – это индивидуальный адрес физического устройства, а при TCP/IP – идентификатор единицы.
- Объединение фрагментов регистров. Стандартные функции 01–04 позволяют запросить сразу несколько смежных регистров или битов. Такая стратегия часто позволяет оптимизировать трафик. Однако нужные регистры не всегда расположены смежно друг к другу, данная опция позволяет собирать их в блоки до 100 регистров или 1600 битов. К установке данного параметра нужно подходить с осторожностью, поскольку не все устройства поддерживают доступ к регистрам между фрагментами.
- Время ожидания соединения в миллисекундах. Указывает промежуток времени, в течение которого ожидать ответа. В случае указания нулевого значения используется время ожидания по умолчанию в транспорте.
- Время восстановления соединения в секундах. Указывает промежуток времени по истечению которого осуществлять повторную попытку запроса к ранее недоступному устройству.
- Попыток запроса для протоколов RTU и ASCII. Указывает на количество попыток повторения запроса в случае получения неполного или повреждённого ответа.

3.2. Параметры

Модуль сбора данных предоставляет только один тип параметров - “Стандарт”. Дополнительным конфигурационным полем параметра данного модуля (рис.9) является перечень обрабатываемых атрибутов(регистров ModBus). Атрибут в этом перечне записывается следующим образом: **<dt>:<numb>:<wr>:<id>:<name>**.

Где:

dt - ModBus тип данных (R-регистр, C-бит, RI-регистр входа, CI-бит входа). R и RI могут быть расширены суффиксами: i2-Int16, i4-Int32, f-Float, b5-Bit5.

numb - номер регистра или бита ModBus устройства (десятичный, восьмеричный или шестнадцатеричный);

wr - режим чтения-записи (r-чтение, w-запись, rw-чтение и запись);

id - идентификатор атрибута OpenSCADA;

name - имя атрибута OpenSCADA.

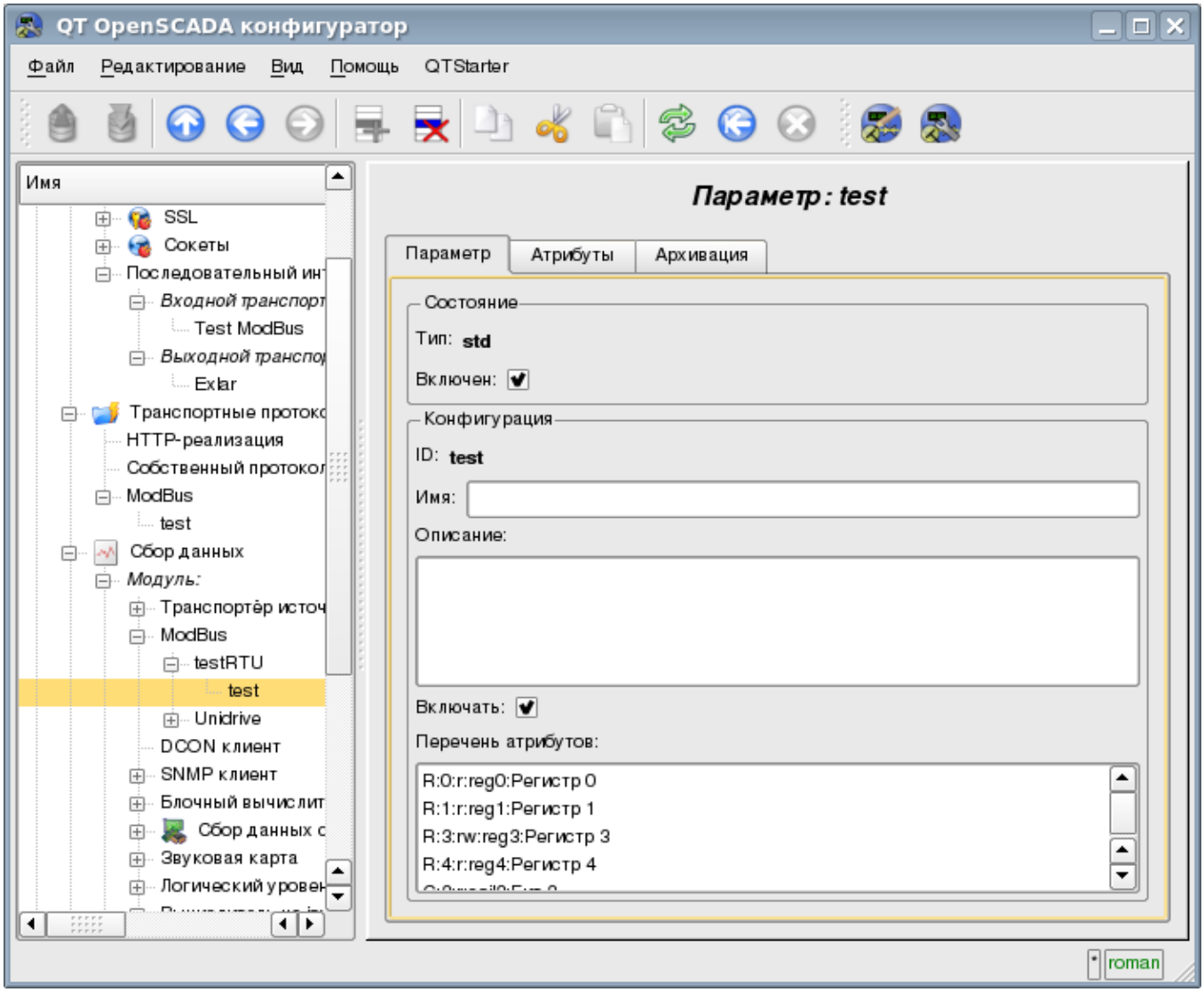


Рис.9. Вкладка конфигурации параметра.

В соответствии с указанным списком атрибутов выполняется опрос и создание атрибутов параметра (рис.10).

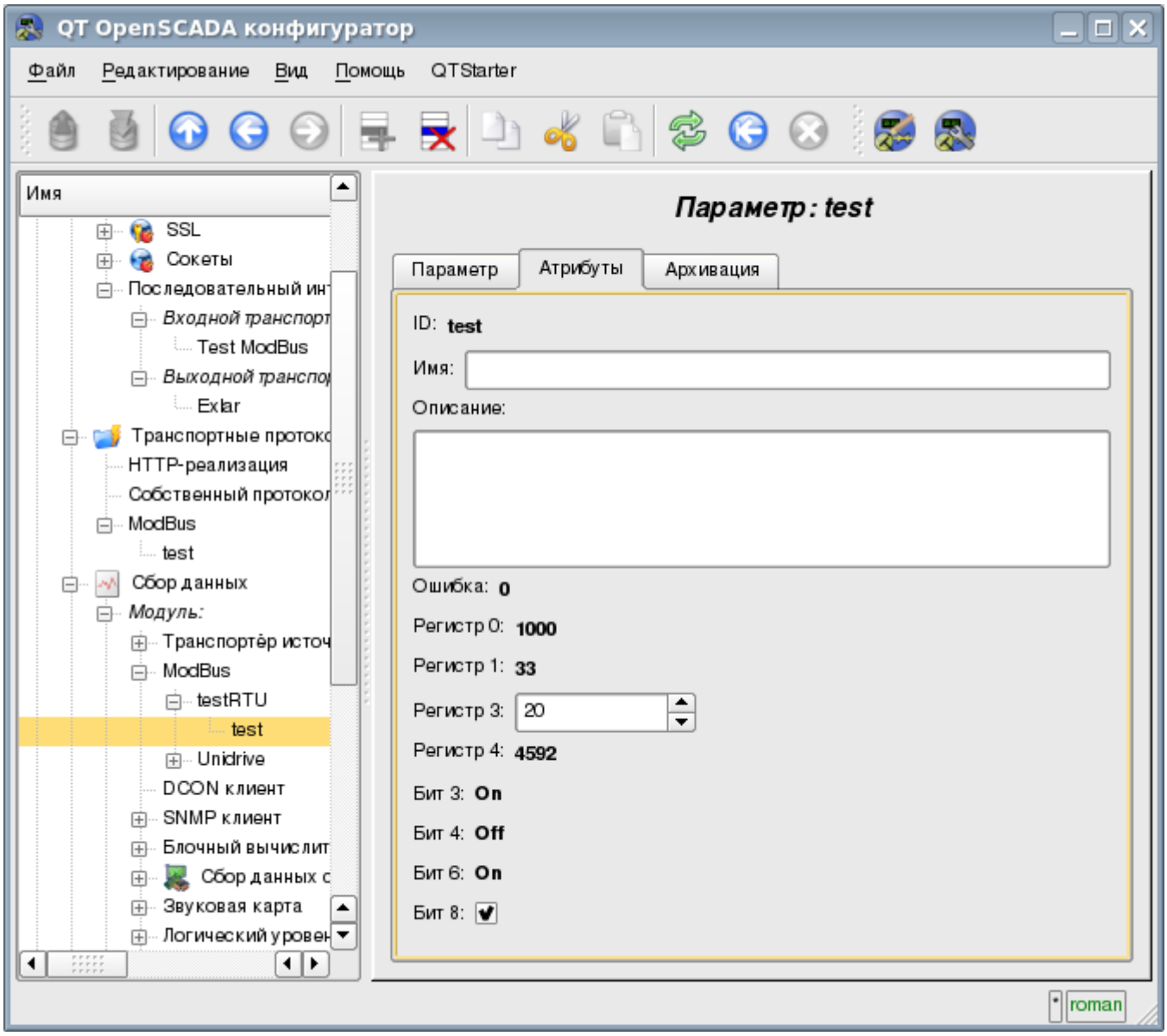


Рис.10. Вкладка атрибутов параметра.

Модуль подсистемы “Сбор данных” <DCON>

Модуль:	DCON
Имя:	DCON клиент
Тип:	DAQ
Источник:	daq_DCON.so
Версия:	0.3.3
Автор:	Алмаз Каримов
Описание:	Предоставляет реализацию клиента DCON-протокола. Поддерживает I-7000 DCON протокол.
Лицензия:	GPL

DCON – протокол семейств контроллеров ADAM (http://www.advantech.com/, http://ipc2u.ru/), ICP DAS (http://www.icpdas.com/, http://ipc2u.ru/), RealLab (http://www.RLDA.ru/) и подобных. Использует для передачи данных последовательные линии связи RS-485.

Данный модуль предоставляет возможность ввода-вывода информации с различных устройств по протоколу DCON. Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

1. Общее описание протокола DCON

Протокол DCON предполагает одно ведущее (запрашивающее) устройство в линии (master), которое может передавать команды одному или нескольким ведомым устройствам (slave), обращаясь к ним по уникальному в линии адресу. Синтаксис команд протокола позволяет адресовать 255 устройств на одной линии связи стандарта RS-485.

Инициатива проведения обмена всегда исходит от ведущего устройства. Ведомые устройства прослушивают линию связи. Мастер подаёт запрос (посылка, последовательность байт) в линию и переходит в состояние прослушивания линии связи. Ведомое устройство отвечает на запрос, пришедший в его адрес.

2. Модуль

Данный модуль предоставляет возможность прозрачного опроса и записи портов ввода-вывода устройств, совместимых с ICP DAS I-7000. На вкладках настроек модуля DCON вводятся необходимые настройки, а на вкладках атрибутов появляются соответствующие заданным параметрам переменные ввода-вывода.

2.1. Контроллер данных

Для добавления источника данных DCON создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

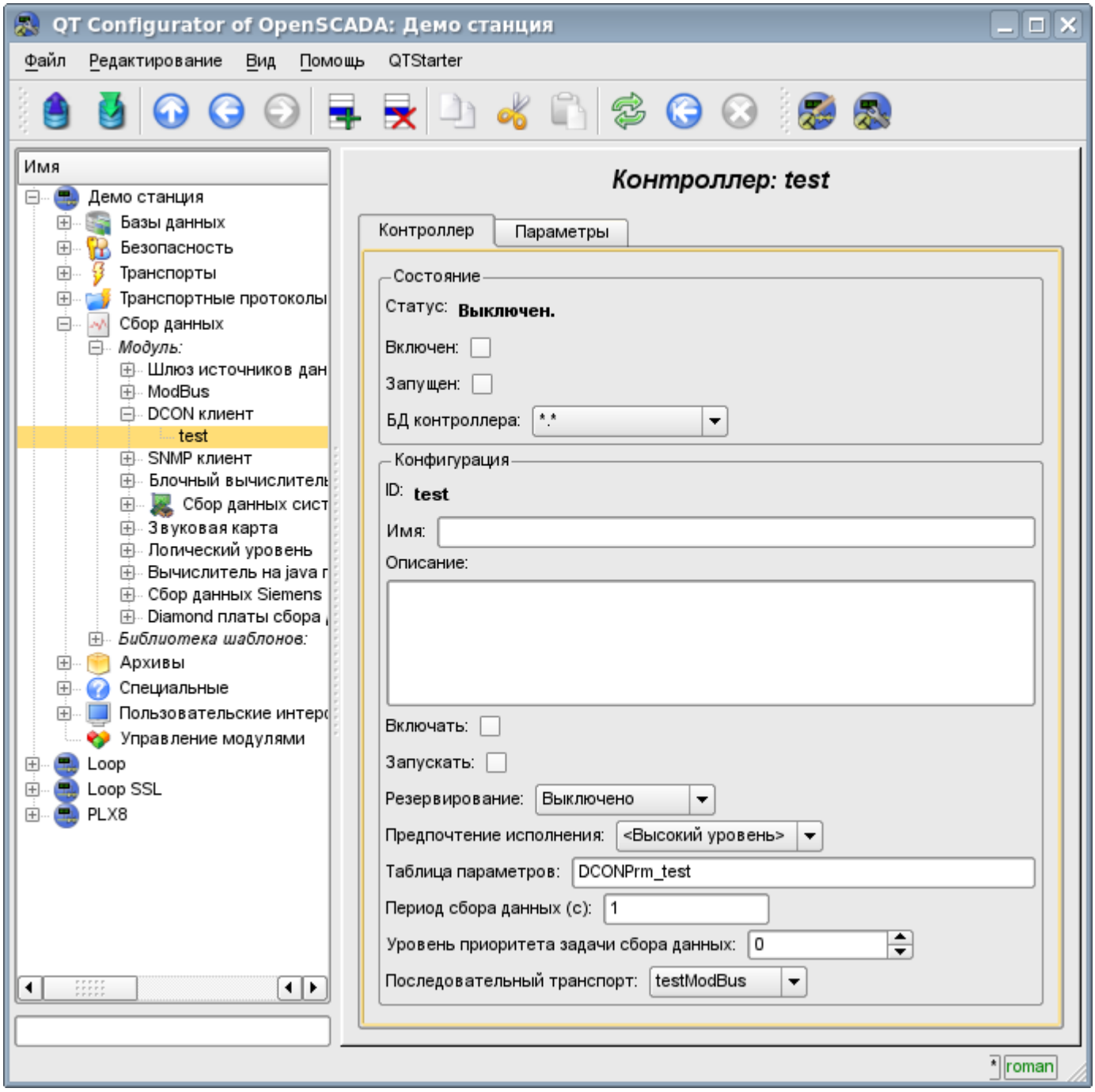


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Состояние, «Включен», Запущен» и имя БД содержащей конфигурацию.

- Идентификатор, имя и описание контроллера.
- Состояние в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи сбора данных.
- Имя исходящего транспорта последовательного интерфейса, сконфигурированного в модуле транспорта “Serial”.

2.2. Параметры

Модуль *DCON* предоставляет только один тип параметров – “Стандарт”. На вкладке параметров можно установить:

- Состояние параметра «Включен»: требует отключения-включения для вступления изменений на этой вкладке в силу.
- Идентификатор, имя и описание параметра.
- Состояние в которое переводить параметр при загрузке «Включен».
- Тип модуля ввода-вывода I-7000.
- Адрес устройства I-7000 в сети RS-485. В десятичном виде от 0 до 255.
- Флаг проверки контрольной суммы. Должен соответствовать заданному в модуле ввода-вывода I-7000.

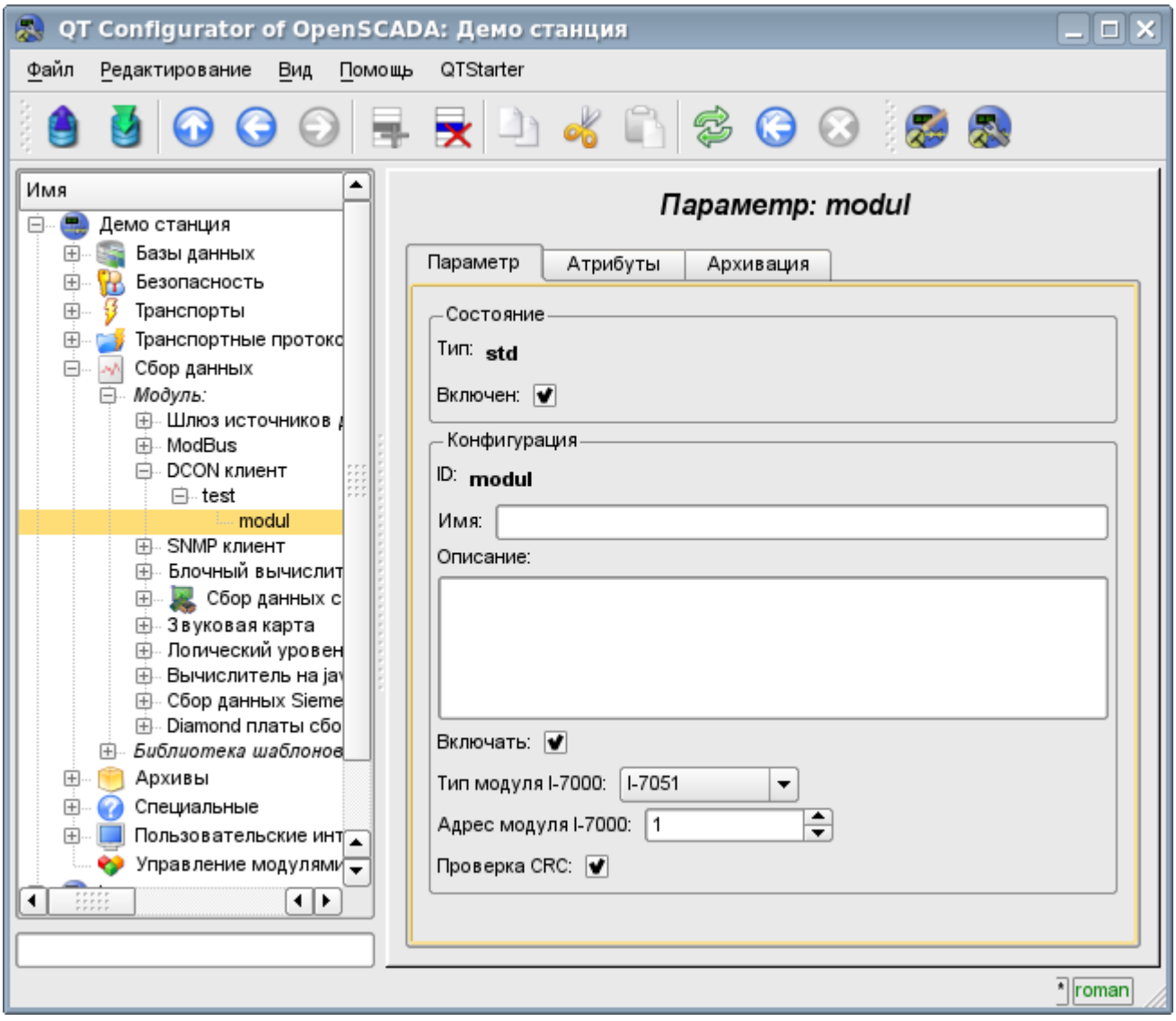


Рис.2. Вкладка конфигурации параметра.

В соответствии с настройками параметра выполняется опрос и создание атрибутов (рис.3).

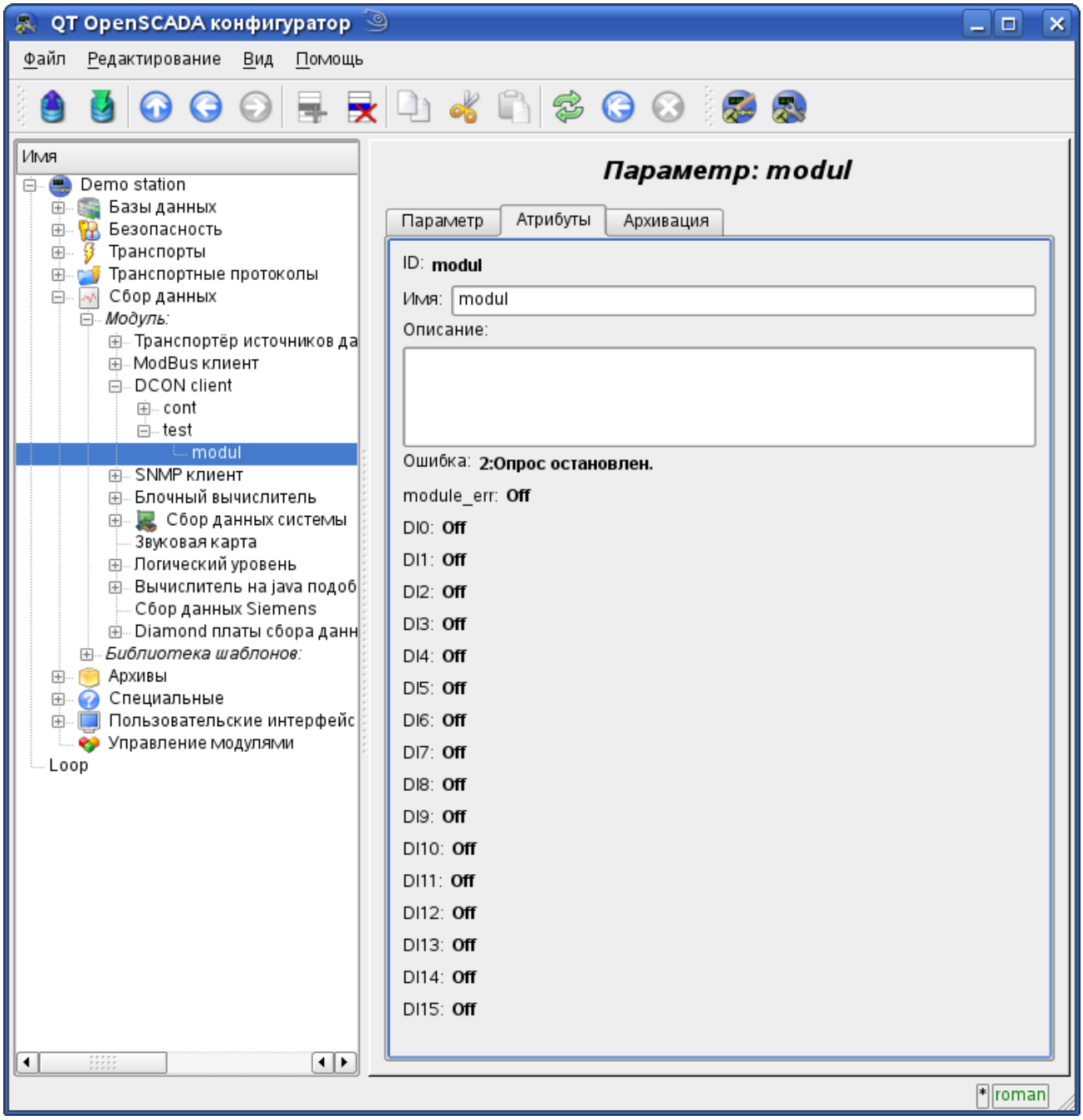


Рис.3. Вкладка атрибутов параметра.

3. Таблица совместимости модулей ввода-вывода различных производителей

№№ п/п	Модуль	ICP DAS	ADAM	RealLab
1	I-7051	I-7051, I-7053*	ADAM-4051*, ADAM-4053*	NL-16DI*, NL-16HV*
2	I-7045	I-7045, I-7043*		NL-16DO*
3	I-7063	I-7063		
4	I-7017	I-7017, I-7018*, I-7019*, I-7005*	ADAM-4017*, ADAM-4018*, ADAM-4019*	NL-8TI*, NL-8AI
5	I-7024	I-7024	ADAM-4024*	NL-4AO

* – не проверено.

Модуль подсистемы “Сбор данных” <ICP_DAS>

Модуль:	ICP_DAS
Имя:	Оборудование ICP_DAS
Тип:	DAQ
Источник:	daq_ICP_DAS.so
Версия:	0.7.2
Автор:	Савоченко Роман
Описание:	Предоставляет реализацию поддержки оборудования ICP DAS. Включена поддержка I-87000 и I-7000 DCON модулей и I-8000 быстрых модулей.
Лицензия:	GPL

Модуль предоставляет в систему OpenSCADA поддержку различного оборудования фирмы ICP DAS (http://www.icpdas.com/, http://ipc2u.ru/) посредством библиотеки API фирмы *libi8k.a*. Большинство оборудования фирмы ICP DAS работает по протоколу DCON, однако часть нового оборудования, например, серия I-8000 работает на параллельной шине, а другая часть устанавливаясь в слоты параллельной шины I-8000, доступные по последовательному интерфейсу и протоколу DCON, не адресуются прямо и требуют вызова специализированной команды выбора слота. Доступ к оборудованию, использующему прямые запросы по протоколу DCON, может быть осуществлён модулем *DAQ.DCON*. Поддержка остального оборудования не добавлялась в модуль *DAQ.DCON*, а реализовывалась в данном модуле по причине наличия библиотеки API фирмы ICP_DAS только для платформы x86_32, что вносит ограничения на доступ к оборудованию фирмы ICP DAS и другому оборудованию по протоколу DCON на других аппаратных платформах.

Причиной создания данного модуля стало ведение работ с контроллером фирмы ICP_DAS LP-8781 серии LinPAC с целью реализации среды исполнения PLC на основе системы OpenSCADA.

Библиотека API фирмы ICP_DAS (libi8k.a) размещена вместе с исходными текстами данного модуля и не требует отдельной инсталяции.

1. Контроллер данных

Для добавления источника данных ICP DAS создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

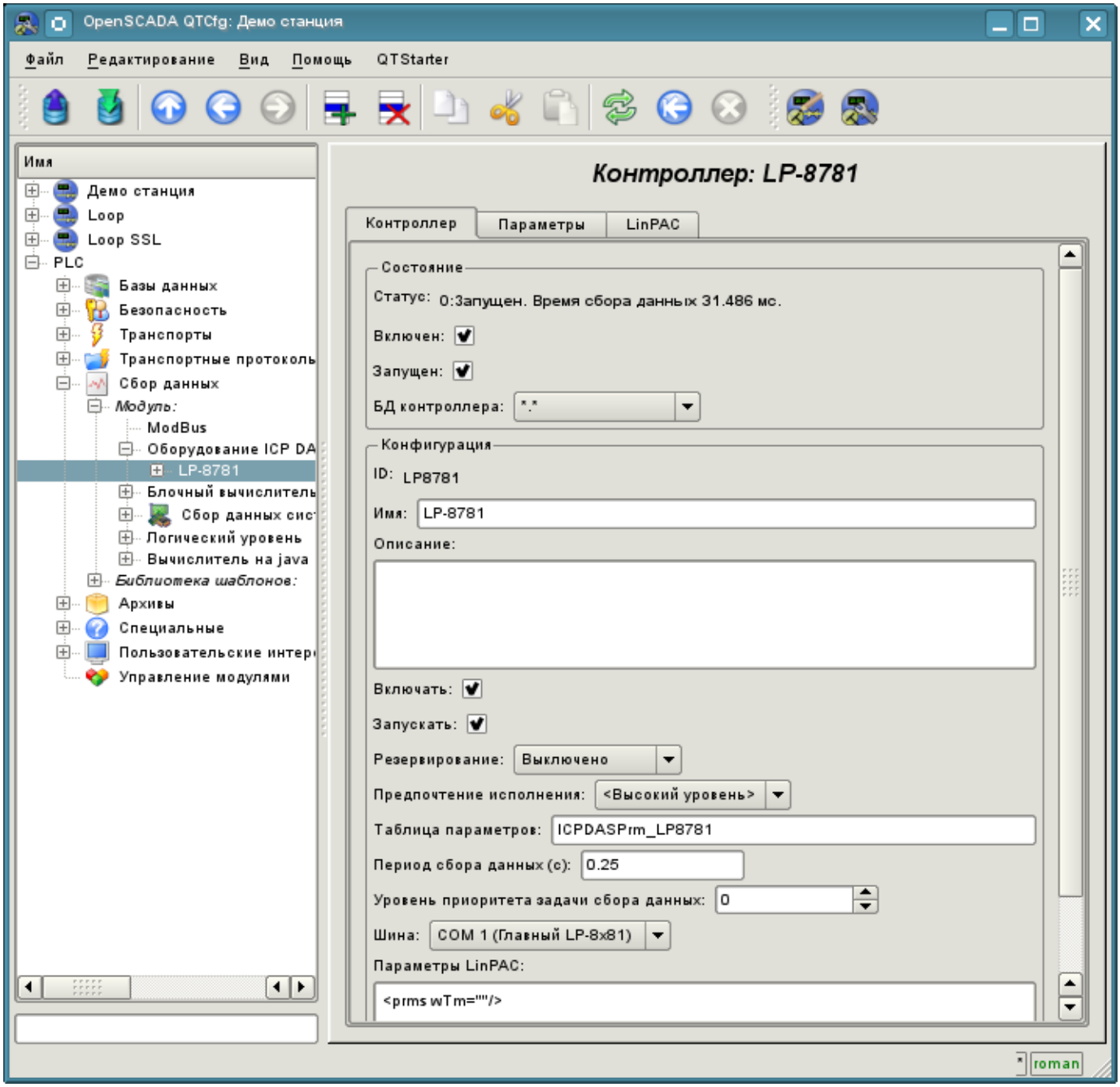


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: статус, состояния "Включен" и Запущен" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: "Включен" и "Запущен".
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы для хранения конфигурации параметров контроллера.
- Период и приоритет задачи сбора данных.
- Шина, на которой расположены модули. Если указан последовательный интерфейс (COMx), то доступ производится по протоколу DCON. В случае указания главной шины контроллера LP-8x81 доступ производится через API параллельной шины или смешанно.

- Параметры LinPAC. Обёрнутые в XML общие параметры контроллера серии LinPAC. Обычно это поле редактируется не в ручную, а во вкладке "LinPAC".
- Скорость передачи данных для последовательного интерфейса. Указывается для неглавной шины.
- Количество попыток последовательных запросов.

2. Параметры

Модуль предоставляет только один тип параметров - “Стандарт”. На вкладке параметров можно установить:

- Состояние параметра, а именно: тип и состояние "Включен".
- Идентификатор, имя и описание параметра.
- Состояние, в которое переводить параметр при загрузке: "Включен".
- Тип модуля ввода-вывода.
- Адрес модуля ввода-вывода, в случае работы не на главной шине - в десятичном виде от 0 до 255.
- Слот модуля в случае работы с устройствами серии I-8000.
- Дополнительные параметры модуля. Используется не всеми модулями и содержит текст в формате XML. Не предназначен для ручного редактирования, а формируется на вкладке "Конфигурация", которая обычно индивидуальна для каждого типа модулей.

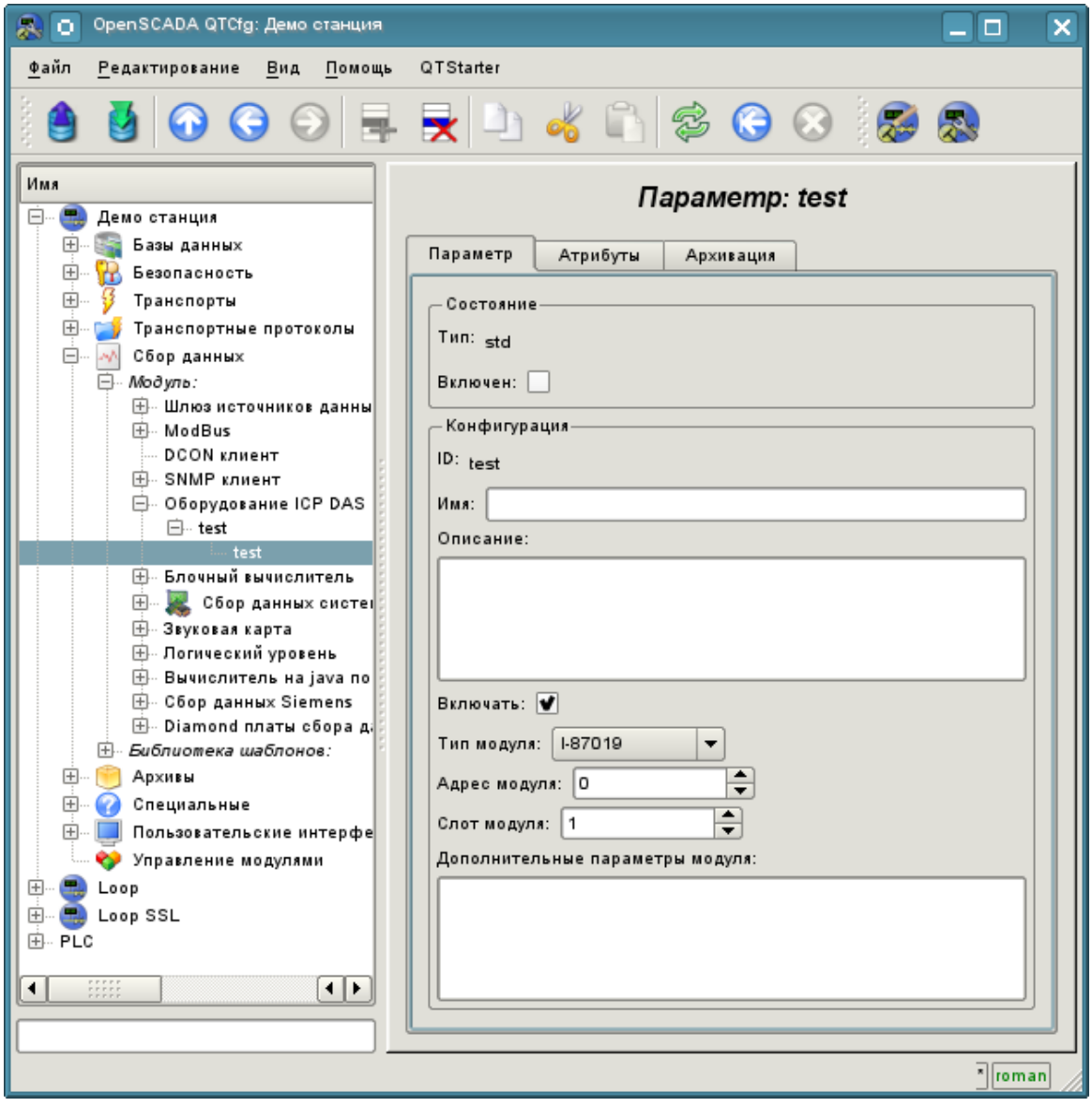


Рис.2. Вкладка конфигурации параметра.

В соответствии с настройками параметра выполняется опрос и создание атрибутов (рис.3).

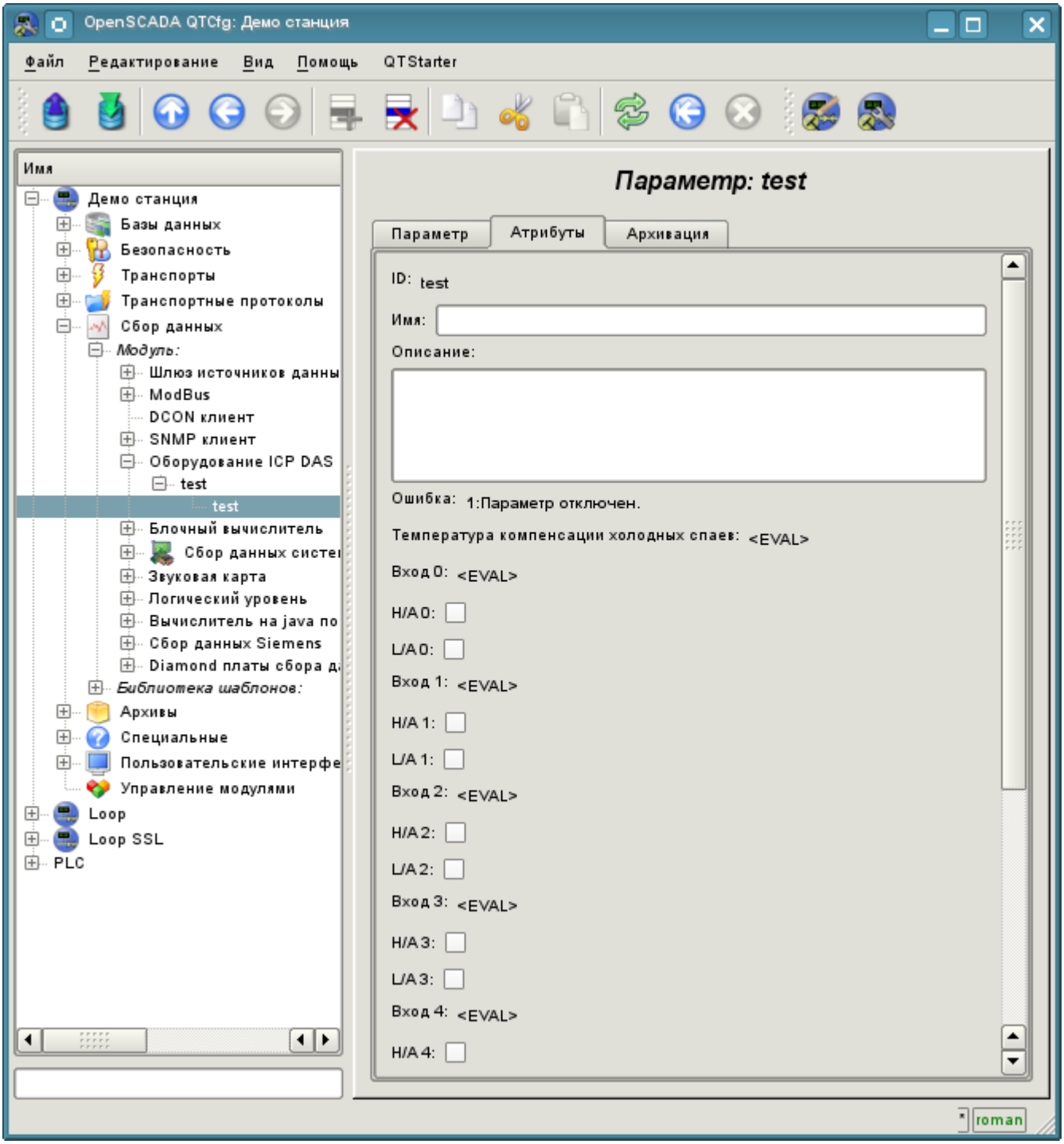


Рис.3. Вкладка атрибутов параметра.

2.1 Модуль I-8017

Быстрый модуль аналогового ввода, работающий на параллельной шине. Обеспечивает скорость доступа к данным по одному каналу в 130 КГц. Однако ввиду заложенного аппратного ограничения не позволяет достичь скорости более 33 КГц на канал при сканировании нескольких каналов. При этом ожидание данных осуществляется в глухом цикле, что приводит к большим потерям ресурсов процессора на высоких частотах сбора.

Модулем предоставляются восемь атрибутов аналоговых входов i{x} и по восемь признаков нарушения верхней ha{x} и нижней la{x} границ. Также модулем предоставляется вкладка "Конфигурация" с расширеной настройкой (рис.4):

- *Количество обрабатываемых параметров* -- указывает сколько входов обслуживать. Характерен для режима быстрого сбора данных и используется для ограничения количества обрабатываемый каналов соразмерно используемым ресурсам процессора.
- *Периодичность быстрого сбора данных (с)* -- указывает с какой периодичностью осуществлять быстрый сбор данных для количества каналов, указанных ранее. Режим быстрого сбора данных отключается указанием нулевого периода.
- Режимы усилений входов отдельно для каждого входа определяет усиление из ряда: +-1.25В, +-2.5В, +-5В, +-10В и +-20мА.

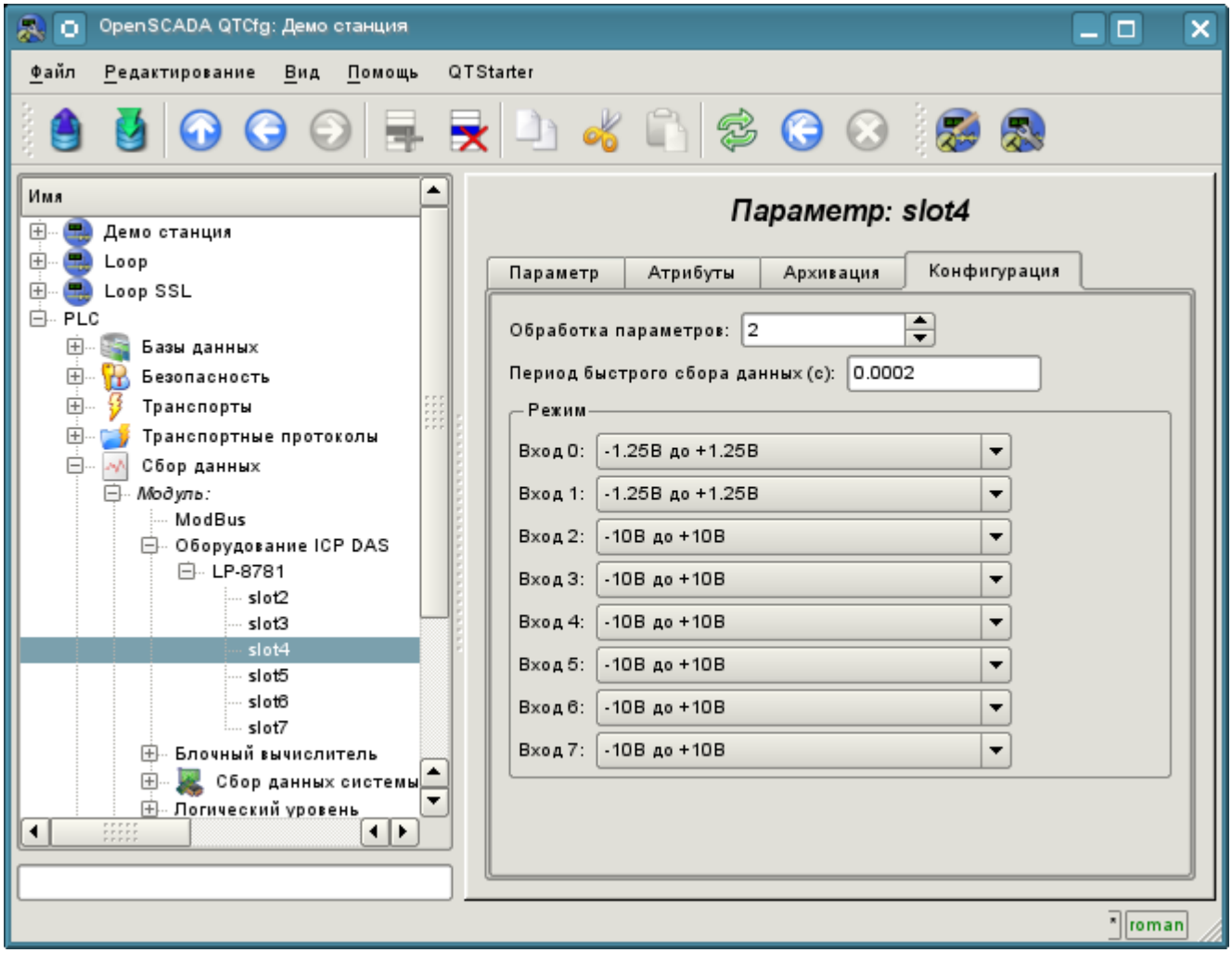


Рис.4. Вкладка "Конфигурация" модуля I-8017

2.2 Модуль I-8042

Быстрый модуль дискретных входов/выходов, работающий на параллельной шине. Предоставляет 16 атрибутов для входов i{x} и 16 для выходов o{x}.

2.3 Модуль I-87019

Модуль аналогового ввода на восемь каналов работающий на последовательной шине и доступный по протоколу DCON. Предоставляет восемь атрибутов аналоговых входов i{x} и по восемь признаков нарушения верхней ha{x} и нижней la{x} границ. Также модуль предоставляет показания температуры холодных спаев термопар.

Модулем предоставляется вкладка "Конфигурация" с расширеной настройкой режимов входов (рис.5) из ряда: +-15мВ, +-50мВ, +-100мВ, +-150мВ, +-500мВ, +-1В, +-2.5В, +-5В, +-10В, +-20мА, J тип, K тип, T тип, E тип, R тип, S тип, B тип, N тип, C тип, L тип, M тип, L тип (DIN43710C).

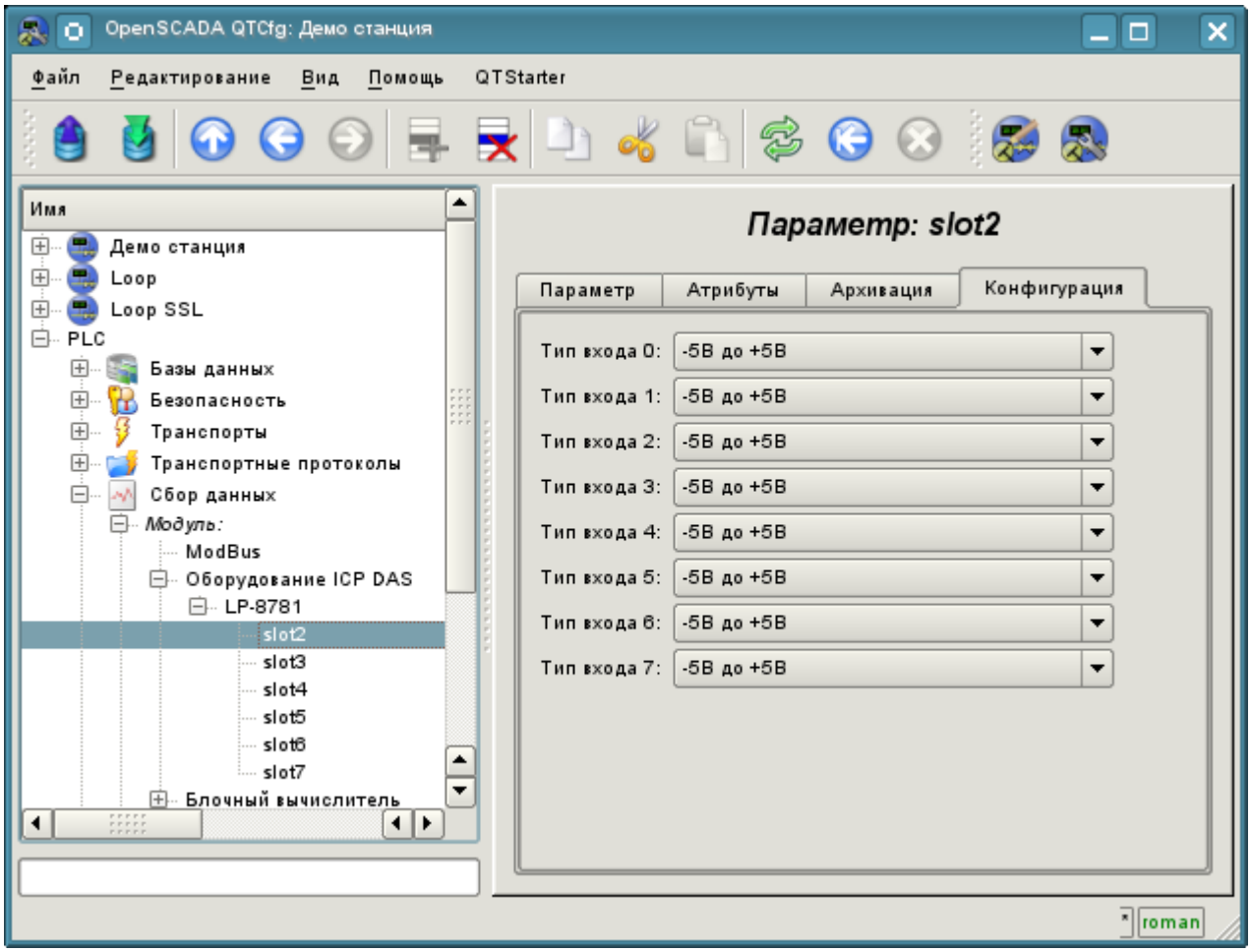


Рис.5. Вкладка "Конфигурация" модуля I-87019

2.4 Модуль I-87024

Модуль аналогового вывода на четыре канала, работающий на последовательной шине и доступный по протоколу DCON. Предоставляет четыре атрибута аналоговых выходов o{x}.

В дополнении содержит вкладку "Конфигурации" (рис.6) с настройкой хостового сторожевого таймера модуля и значений выходов при включении и перезапуске по сторожевому таймеру.

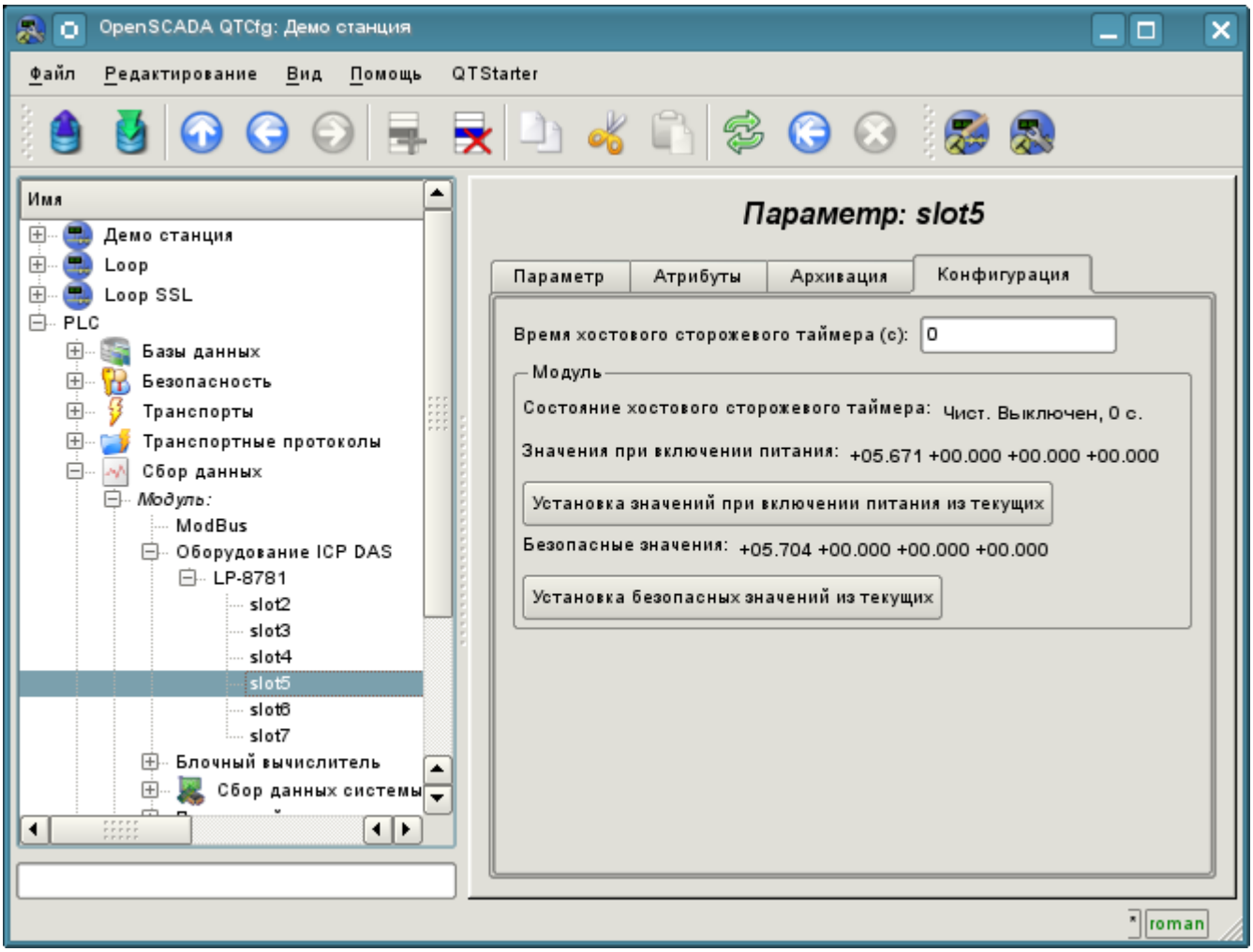


Рис.6. Вкладка "Конфигурация" модуля I-87024

2.5 Модуль I-87057

Модуль дискретных выходов на 16 каналов, работающий на последовательной шине и доступный по протоколу DCON. Предоставляет 16 атрибутов дискретных выходов o{x}.

В дополнении содержит вкладку "Конфигурации" с настройкой хостового сторожевого таймера модуля и значений выходов при включении и перезапуске по сторожевому таймеру.

3. Настройка контроллеров серии LP-8x81

Для конфигурации свойств контроллеров серии LP-8x81 предусмотрена соответствующая вкладка на странице модуля (рис.7), где можно получить информацию о серийном номере контроллера, версии SDK и значении DIP-переключателя, а так-же установить значение сторожевого таймера контроллера. Сторожевой таймер контроллера выключается установкой нулевого значения. Обновление значения сторожевого таймера осуществляется в задаче контроллера и с её периодом. Следовательно зависание задачи опроса приводит к перезапуска контроллера!

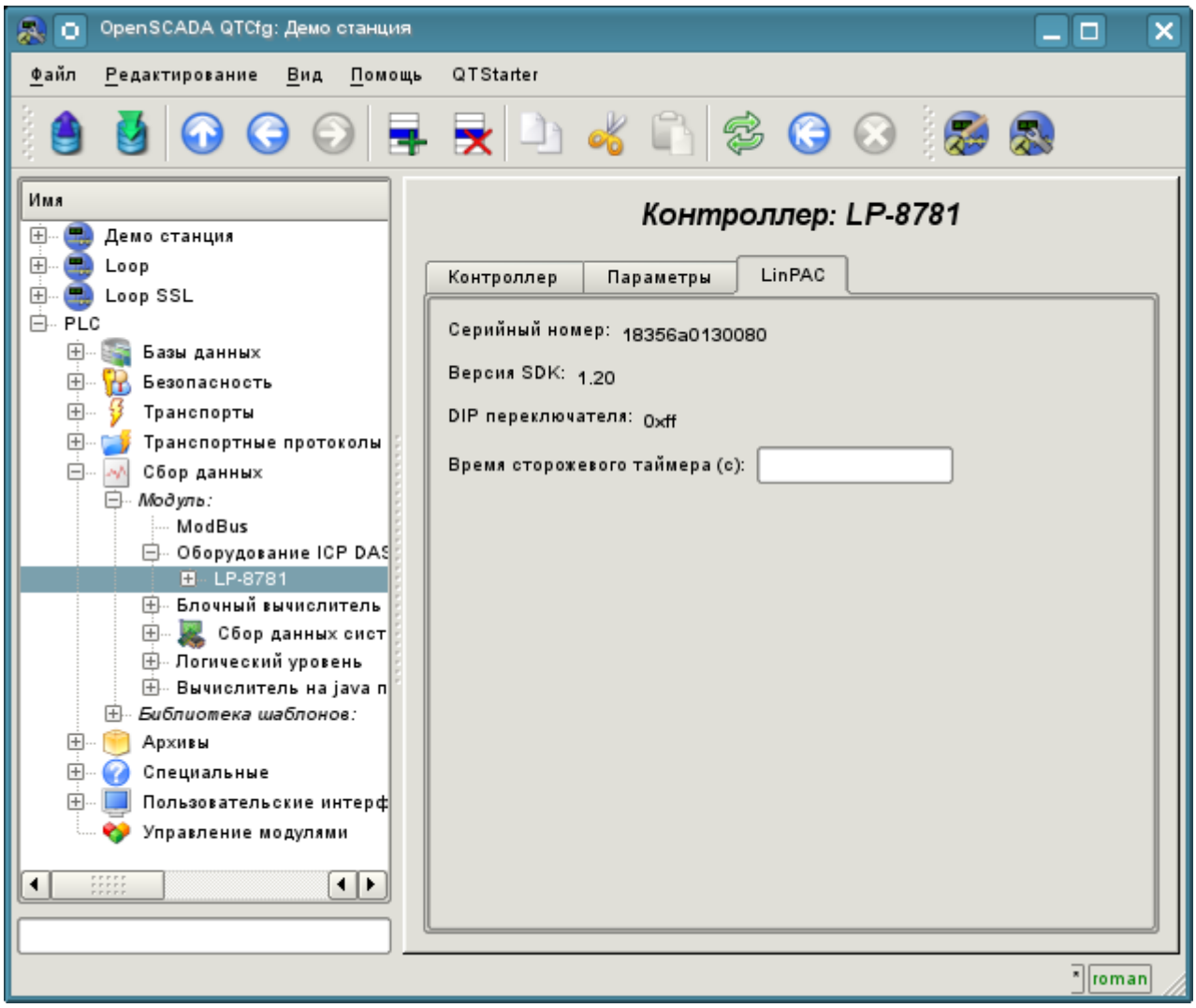


Рис.7. Вкладка конфигурации контроллеров серии LP-8x81

Ссылки

Специальные модули ядра Linux 2.6.29 для контроллеров LP-8x81: file:lp8x81_2629.tgz

Драйвер производителя (VIA) для сети контроллеров LP-8x81: file:rhinefet20070212111037.tgz
На стандартном Linux драйвере сети скорость падает на порядок после суток работы

Патч для сборки сетевого драйвера на Linux 2.6.29: file:build_2.6.29.patch

Модуль подсистемы “Сбор данных” <DAQGate>

Модуль:	DAQGate
Имя:	Шлюз источников данных
Тип:	DAQ
Источник:	daq_DAQGate.so
Версия:	0.9.1
Автор:	Роман Савоченко
Описание:	Позволяет выполнять шлюзование источников данных удалённых OpenSCADA станций в локальные.
Лицензия:	GPL

Основной функцией данного модуля является отражение данных подсистемы «Сбор данных» удалённых OpenSCADA станций на локальные. В своей работе модуль использует собственный протокол системы OpenSCADA (Self System) и сервисные функции подсистемы «Сбор данных».

Модулем реализуется следующие функции:

- Отражение структуры параметров подсистемы «Сбор данных» удалённой станции. Структура периодически, при работе, синхронизируется.
- Доступ к конфигурации параметров. Конфигурация параметров контроллеров удалённых станций прозрачно отражается, что позволяет менять её удалённо.
- Доступ к текущим значениям атрибутов параметров и возможность их модификации. Значения атрибутов параметров обновляются с периодичностью исполнения локального контроллера. Запросы на модификацию атрибутов транслируются на удалённую станцию.
- Отражение архивов значений отдельных атрибутов параметров. Реализовано отражение архивов двумя способами. Первый способ предусматривает создание локального архива для атрибута и его синхронизацию с удалённым, при этом поддерживается восстановление архива на время остановки станции. Второй способ предусматривает трансляцию запросов локального архива к архиву удалённой станции.
- Предоставление реализации механизма вертикального резервирования, а именно возможность отражения данных с нескольких станций одного уровня.
- Реализация функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

Использование доступных схем резервирования наглядно представлено на рис.1.

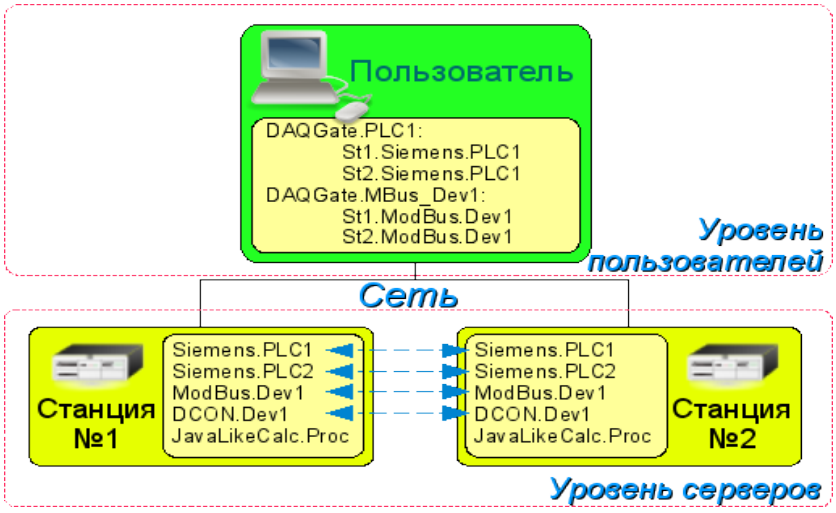


Рис.1. Горизонтальное и вертикальное резервирование.

1. Контроллер данных

Для добавления источника данных создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.2.

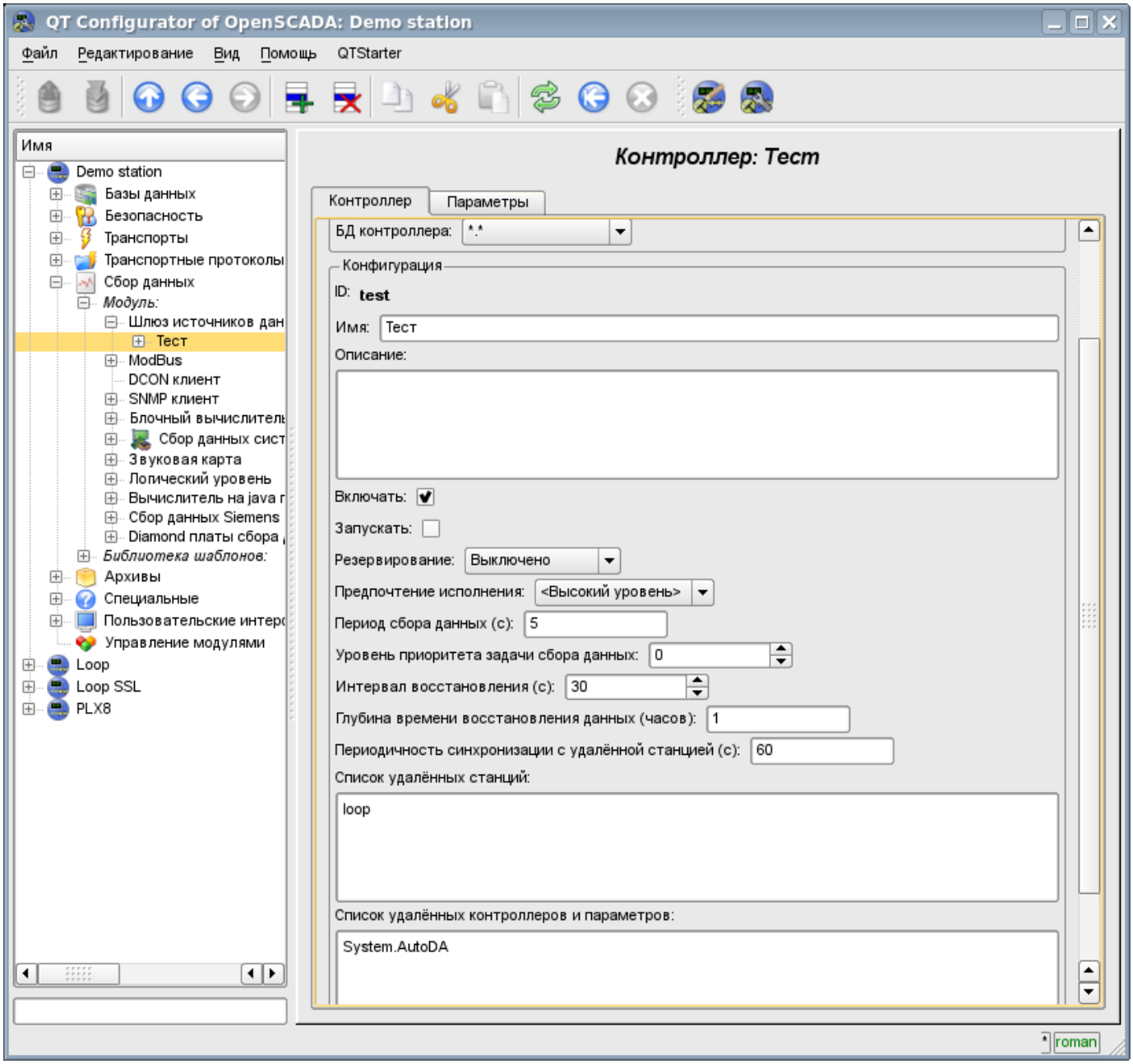


Рис.2. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Период, в секундах, и приоритет задачи сбора данных.
- Интервал времени повторения попытки восстановления связи с потерянной станцией, в секундах.
- Максимальная глубина данных архива для восстановления при запуске, в часах.
- Период синхронизации с удалённой станцией, в секундах.
- Список отражаемых удалённых станций. Несколько станций в списка включают механизм вертикального резервирования.

- Список отражаемых контроллеров и параметров. В списке можно указывать как только контроллеры, для отражения всех их параметров, так и отдельно взятые параметры.
- Команда перехода к конфигурации перечня удалённых станций.

2. Параметры

Модуль не предоставляет возможности создания параметров вручную, все параметры создаются автоматически с учётом списка отражаемых контроллеров и параметров. Пример отраженного параметра приведён на рис. 3.

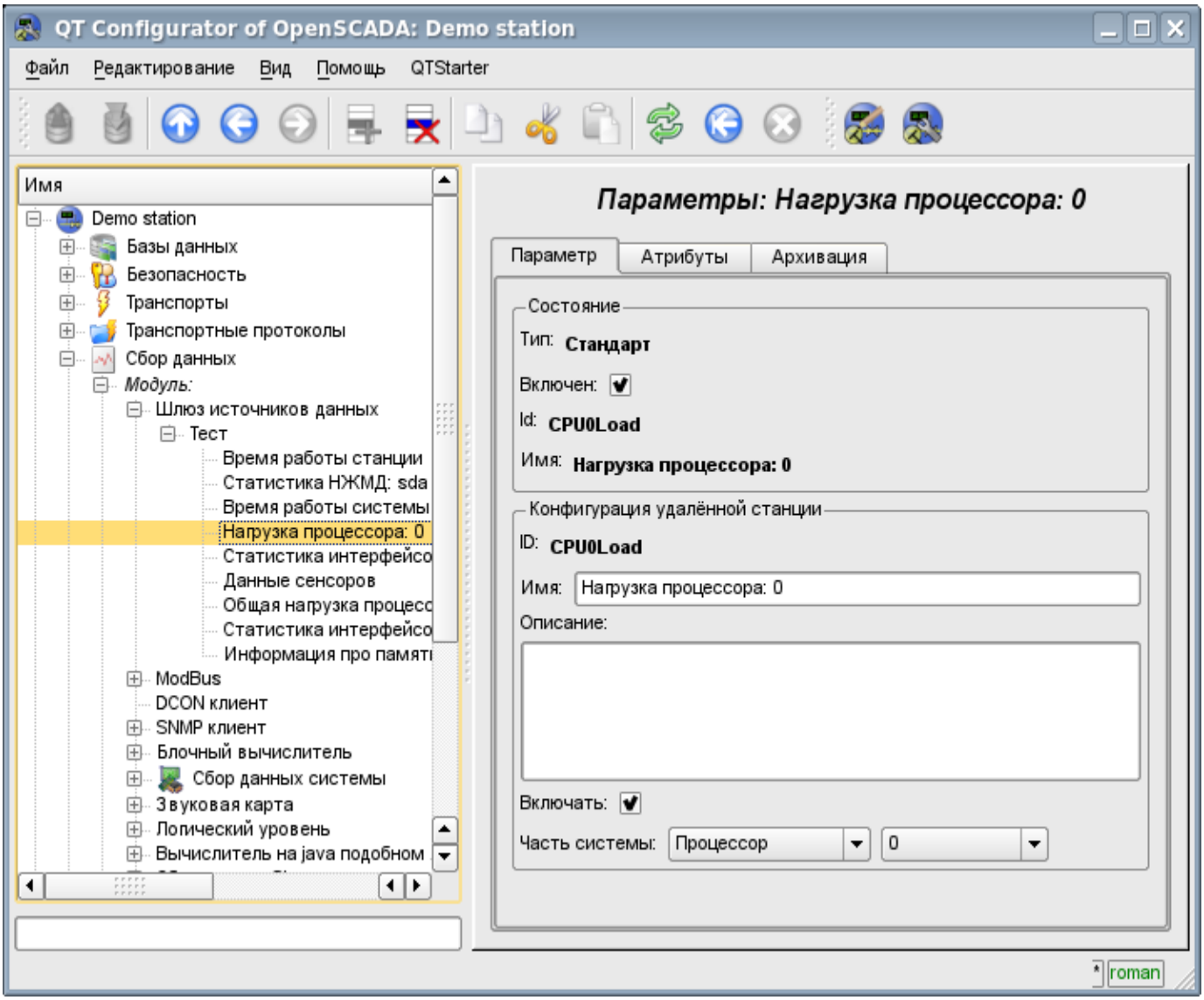


Рис.3. Вкладка конфигурации отражённого параметра.

Модуль подсистемы “Сбор данных” <SoundCard>

Модуль:	SoundCard
Имя:	Звуковая карта
Тип:	DAQ
Источник:	daq_SoundCard.so
Версия:	0.6.1
Автор:	Роман Савоченко
Описание:	Предоставляет доступ к звуковой карте.
Лицензия:	GPL

Данный модуль предназначен для предоставления данных со входов звуковых карт системы. Модуль основан на многоплатформенной библиотеке работы со звуком PortAudio (http://www.portaudio.com). Особенностью этой библиотеки является унифицированное API, которое позволяет легко адаптировать данный модуль для работы на разных платформах и даже с разными подсистемами звука на одной платформе.

Структура модуля заключается в отражении объекта «Контроллер» подсистемы «Сбор данных» на отдельное устройство ввода звука, доступное в системе. А объект «Параметр» подсистемы «Сбор данных» отражает отдельный канал доступный у устройства ввода звука на атрибут “val”. Наиболее функциональным является использование атрибута “val” совместно с архивом или хотя бы его буфером. В случае включения архивирования данные канала звукового ввода помещаются в буфер архива пакетами с частотой выборки данных устройства ввода, что позволяет выполнять дальнейшие операции над этими данными. Кроме этого, последнее значение пакета устанавливается как текущее значение атрибута. В случае отсутствия архива выполняется только операция помещения последнего значения пакета как текущее значение атрибута.

Также модулем реализуются функции горизонтального резервирования, а именно совместной работы с удалённой станцией этого-же уровня.

1. Контроллер данных

Для добавления устройства ввода звука создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.1.

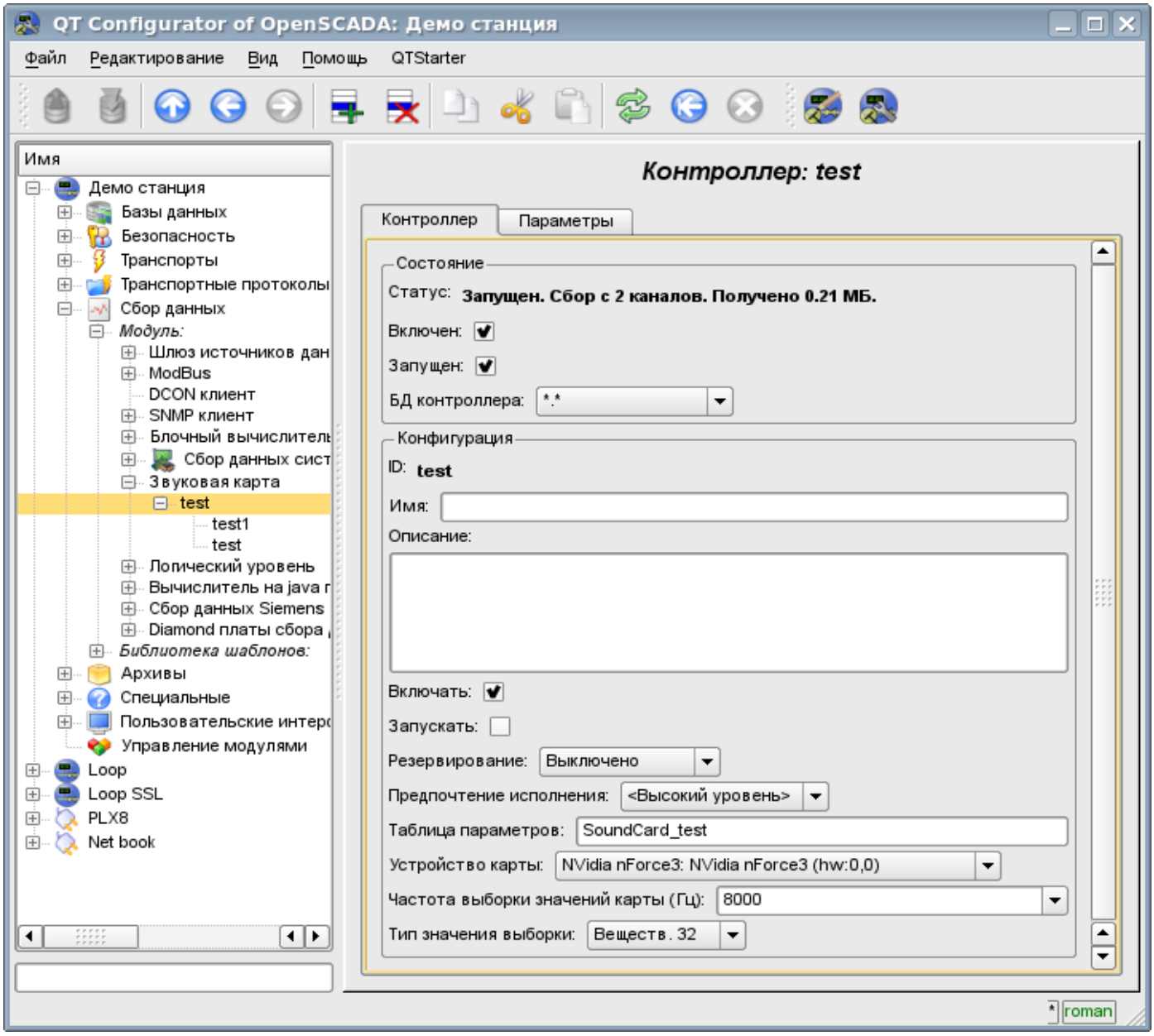


Рис.1. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, «Включен», Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: «Включен» и «Запущен».
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.
- Имя таблицы БД, в которой хранить параметры этого контроллера.
- Устройство карты из списка доступных.
- Частота выборки значений карты в герцах.
- Тип значения выборки из списка: Вещественный 32, Целый 32 и Целый 16.

2. Параметры

Для добавления канала устройства ввода звука создаётся и конфигурируется параметр контроллера в системе OpenSCADA. Пример вкладки конфигурации параметра данного типа изображен на рис.2.

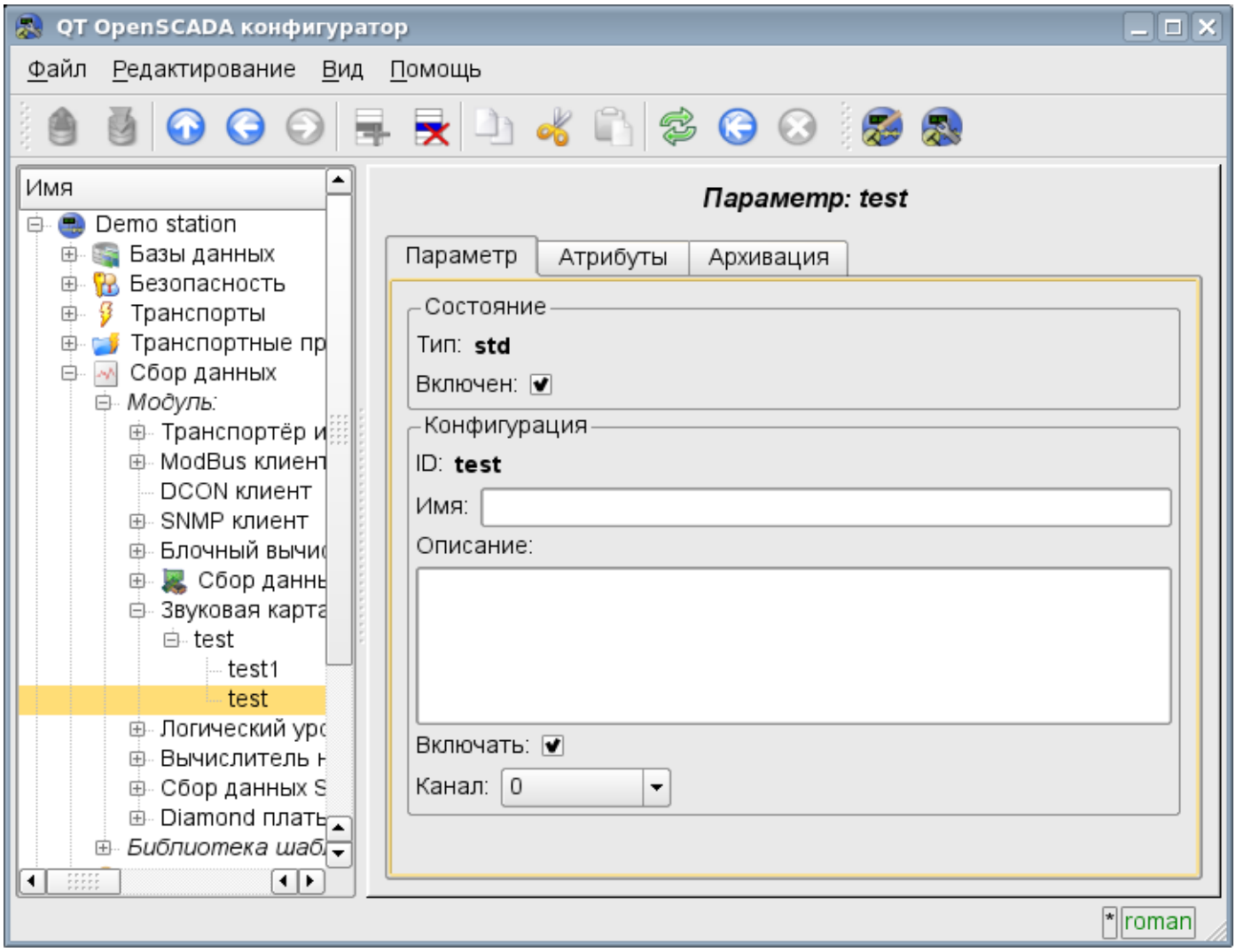


Рис.2. Вкладка конфигурации параметра.

С помощью этой вкладки можно установить:

- Тип параметра и указать состояние параметра «Включен».
- Идентификатор, имя и описание параметра.
- Состояние, в которое переводить параметр при загрузке: «Включен».
- Канал устройства ввода звука из списка доступных каналов.

Вкладка атрибутов параметра имеет вид, представленный на рис.3, а вкладка значений архива атрибута “val” представлена на рис.4.

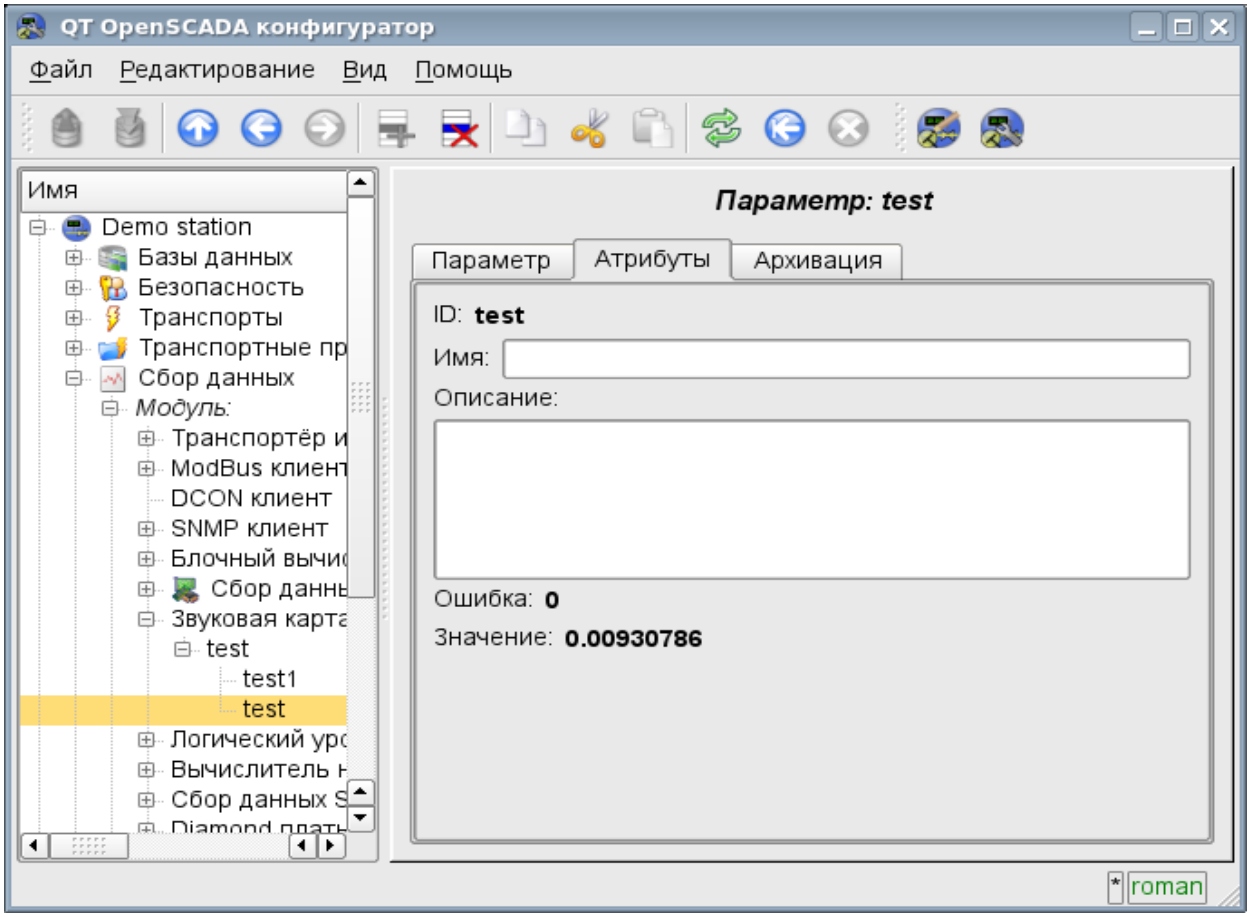


Рис.3. Вкладка атрибутов параметра.

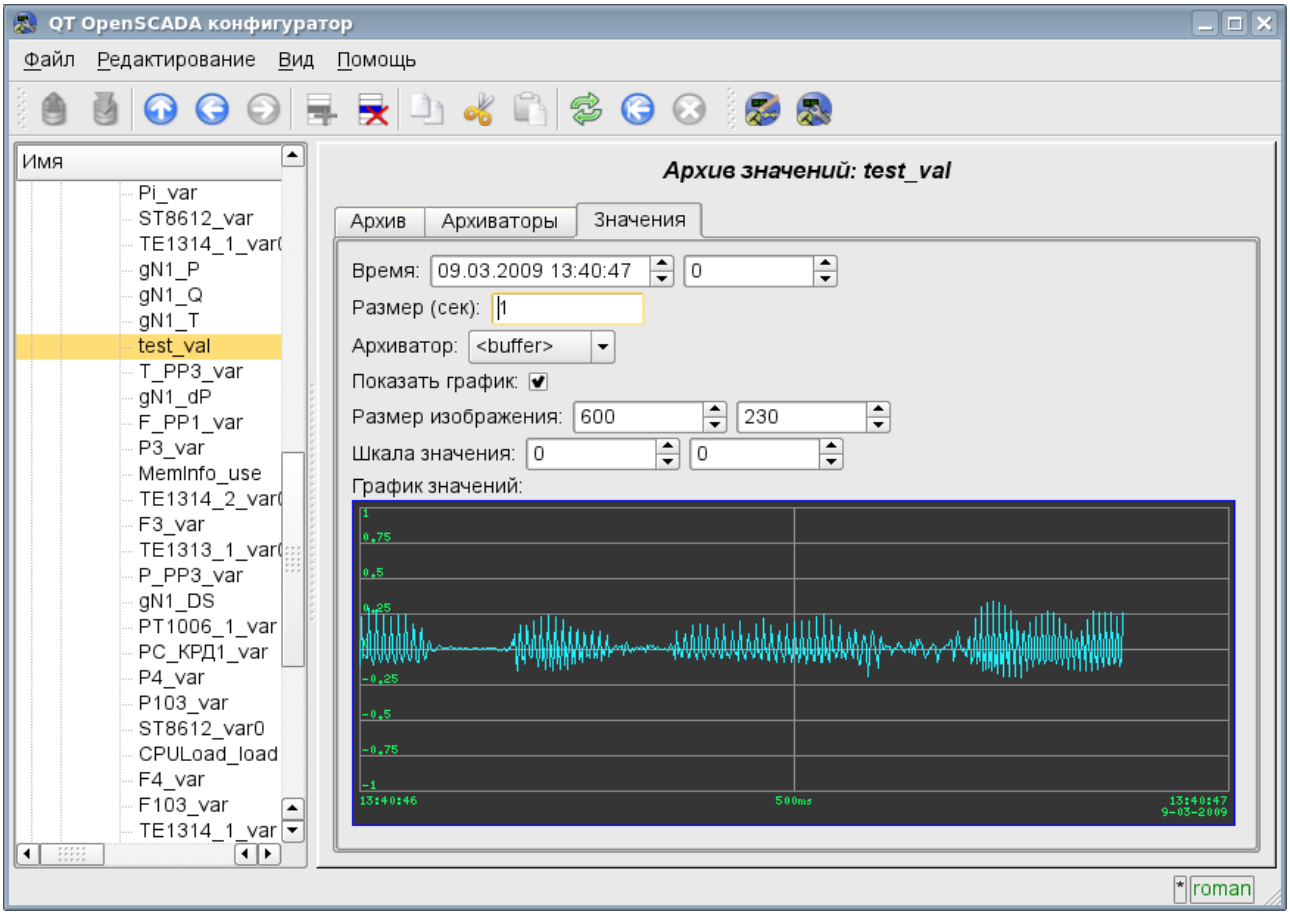


Рис.4. Вкладка значений архива атрибута “val”.

Модули <OPC_UA> подсистемы “Сбор данных” и подсистемы "Транспортные протоколы"

Параметр	Модуль 1	Модуль 2
ID:	OPC_UA	OPC_UA
Имя:	OPC_UA	OPC_UA
Тип:	DAQ	Протокол
Источник:	daq_daq_OPC_UA.so	
Версия:	0.6.0	0.6.0
Автор:	Роман Савоченко	
Описание:	Предоставляет реализацию клиентского сервиса OPC UA.	Предоставляет реализацию OPC UA протокола.
Лицензия:	GPL	

OPC (OLE for Process Control) - это семейство протоколов и технологий, предоставляющих единый интерфейс для управления объектами автоматизации и технологическими процессами. Создание и поддержку спецификаций OPC координирует международная некоммерческая организация OPC Foundation, созданная в 1994 году ведущими производителями средств промышленной автоматизации.

В виду того, что значительное влияние в организации OPC Foundation имеет корпорация Microsoft, протоколы OPC до последнего времени были одноплатформеными и закрытыми, ввиду привязки к закрытым технологиям MS Windows. Однако, с недавних пор организацией OPC Foundation были созданы такие многоплатформенные интерфейсы, как OPC XML DA и OPC UA. Наибольший интерес из них представляет интерфейс OPC UA, как унифицирующий все интерфейсы ранних версий в рамках открытых и многоплатформенных технологий.

Данный модуль реализует поддержку интерфейса и протокола OPC UA как в виде клиентского сервиса, так и в виде сервера OPC UA. Клиентский сервис OPC UA реализуется одноимённым модулем подсистемы "Сбор данных", а сервер реализуется модулем подсистемы "Протоколы".

В текущей версии данных модулей реализуются бинарная часть протокола и базовые сервисы в небезопасном режиме и безопасных режимах политик "Base128Rsa15" и "Base256". В последствии планируется расширение модуля для работы через HTTP/SOAP и реализации остальных сервисов OPC UA.

Хотя протокол OPC UA и является многоплатформенным, его спецификация и SDK не являются свободнодоступными, а предоставляются только членам организации OPC Foundation. По этой причине реализация данных модулей столкнулась со значительными препятствиями и проблемами.

Во-первых, протокол OPC UA сложен и реализация его вообще без спецификации крайне трудоёмка. По этой причине работы над данными модулями долгое время не начиналась, и только благодаря спонсорской помощи одной из организаций-члена OPC Foundation проект OpenSCADA получил документацию спецификации. При этом SDK и исходные тексты ANSIC-API протокола OPC-UA получены не были по причине несовместимости их лицензии с GPL и, как следствие, потенциальной угрозы нарушения лицензии при работе с исходными текстами, что могло привести к последующим юридическим проблемам при свободном распространении данных модулей.

Во-вторых, даже наличие спецификации не позволяет решить ряд технических вопросов без примера реализации и возможности проверки на рабочем прототипе клиента и сервера OPC UA. Например, именно технические особенности реализации алгоритмов симметричного шифрования и

получения ключей для них не позволили реализовать поддержку политик безопасности сразу.

Для отладки функционирования модулей было использовано демонстрационное ПО фирмы Unified Automation в составе OPC UA клиента - UA Expert и сервера - OPC UA Demo Server, из пакета SDK.

1. Протокол OPC UA

OPC UA - это платформо-независимый стандарт, с помощью которого системы и устройства различного типа могут взаимодействовать путём отправки сообщений между клиентом и сервером через различные типы сетей. Протокол поддерживает безопасное взаимодействие путём валидации клиентов и серверов, а также противодействия атакам. OPC UA определяет понятие Сервисы, которые сервера могут предоставлять, а также сервисы, которые сервер поддерживает для клиента. Информация передаётся в виде типов данных, определённых OPC UA и производителем, кроме того сервера определяют объектную модель, для которой клиенты могут осуществлять динамический обзор.

OPC UA предоставляет совмещение интегрированного адресного пространства и сервисной модели. Это позволяет серверу интегрировать данные, нарушения (Alarms) и события (Events), историю в этом адресном пространстве, а также предоставлять доступ к ним посредством интегрированных сервисов. Сервисы также предоставляют интегрированную модель безопасности.

OPC UA позволяет серверам предоставлять для клиентов определения типов, для доступа к объектам из адресного пространства. OPC UA допускает предоставление данных в различных форматах, включая бинарные структуры и XML-документы. Через адресное пространство клиенты могут запросить у сервера метаданные, которые описывают формат данных.

OPC UA добавляет поддержку множественной связности между узлами вместо простого ограничения иерархичностью. Такая гибкость в комбинации с определением типов позволяет применять OPC UA для решения задач в широкой проблемной области.

OPC_UA спроектирован для обеспечения надёжной выдачи данных. Основная особенность всех OPC серверов - способность выдавать данные и события.

OPC_UA спроектирован для поддержки широкого диапазона серверов, от простых ПЛК до промышленных серверов. Эти сервера характеризуются широким спектром размеров, производительности, платформ исполнения и функциональной ёмкости. Следовательно, OPC UA определяет исчерпывающее множество возможностей, и сервер может имплементировать подмножества этих возможностей. Для обеспечения совместимости OPC UA определяет подмножества, именуемые Профилями, которые сервера могут указывать для согласования. Клиенты могут в последствии выполнять обзор профилей сервера и пробрасывать взаимодействие с сервером, основанном на профилях.

OPC UA спецификация спроектирована как ядро в слое, изолированном от подлежащих компьютерных технологий и сетевых транспортов. Это позволяет OPC UA при необходимости расширяться на будущие технологии без отторжения основы дизайна. На данный момент спецификацией определены два способа кодирования данных: XML/text и UA Binary. В дополнение, определено два типа транспортного слоя: TCP и HTTP/SOAP.

OPC UA спроектирован как решение для миграции с OPC клиентов и серверов, которые основаны на Microsoft COM технологиях. OPC COM сервера (DA, HDA и A&E) могут быть легко отражены в OPC UA. Производители могут самостоятельно осуществлять такую миграцию или же рекомендовать пользователям использовать обёртки и конверторы между этими протоколами. OPC UA унифицирует предыдущие модели в едином адресном пространстве с единым множеством сервисов.

2. Модуль реализации протокола

Модуль протокола содержит код реализации протокольной части OPC UA как для клиентского, так и для серверного сервисов. Для построения OPC UA сервера достаточно создать входящий транспорт, обычно это TCP-транспорт модуля Sockets, и выбрать в нём модуль данного протокола, а также сконфигурировать хотя бы один конечный узел модуля протокола, о чём ниже.

2.1. Обслуживание запросов по протоколу OPC UA

Входящие запросы к модулю-протоколу обрабатываются модулем в соответствии со сконфигурированными конечными узлами OPC UA (EndPoints) (рис.1).

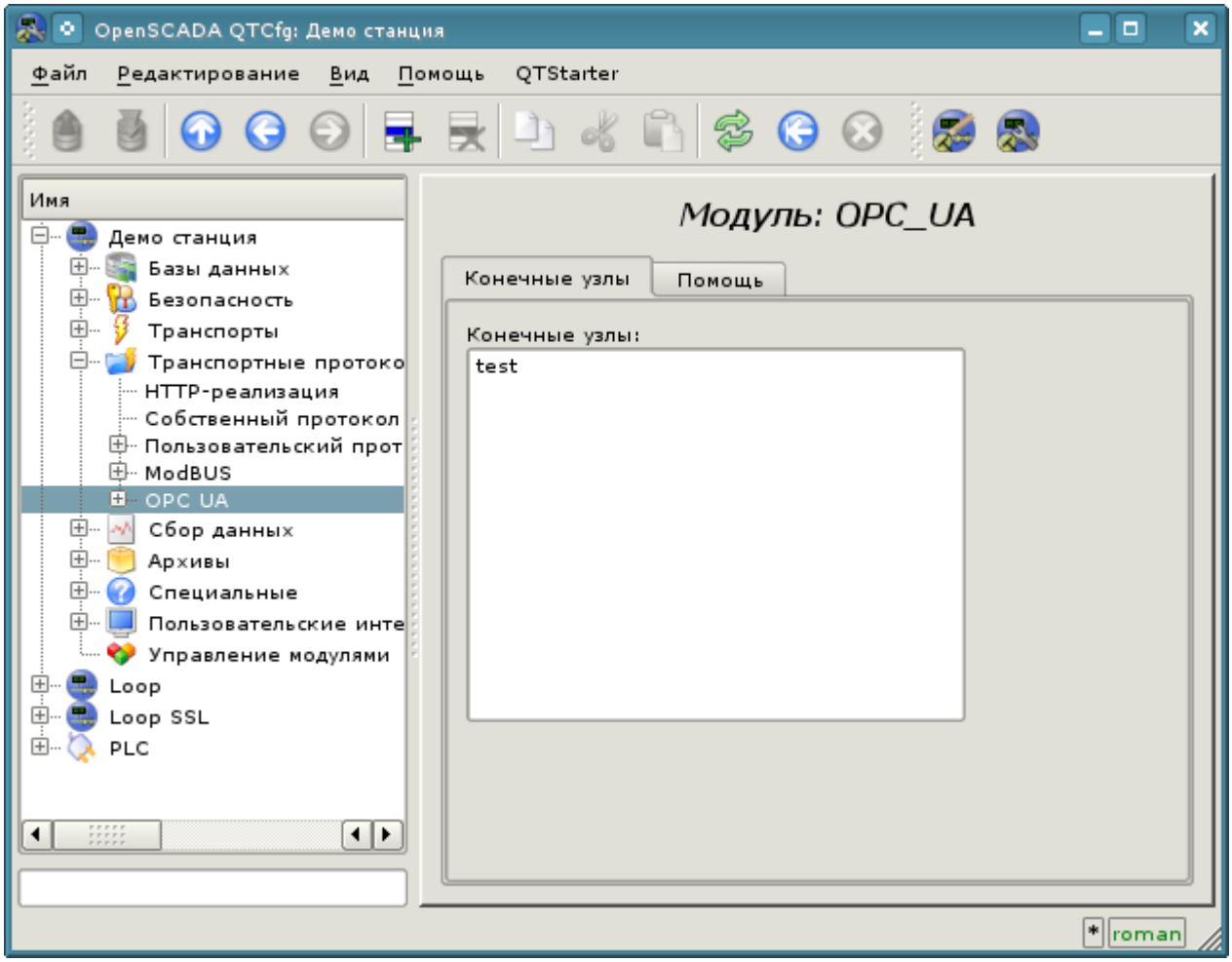


Рис.1. Конечные узлы протокола.

Конечный узел протокола OPC UA - это фактически объект сервера OPC UA. Конечные узлы в OPC UA могут быть как локальными, так и удалёнными. Локальный конечный узел предназначен для предоставления ресурсов станции OpenSCADA по протоколу OPC UA, в тоже время удалённые конечные узлы служат для выполнения как сервиса обзора доступных OPC-UA узлов, так и для шлюзования запросов к удалённым станциям. В данной версии модуля поддерживается только конфигурация локальных конечных узлов.

Общая конфигурация конечного узла осуществляется на главной вкладке страницы конечного узла (рис.2) параметрами:

- Состояние узла, а именно: статус, "Включен" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание узла.
- Состояние, в которое переводить узел при загрузке: "Включен".
- Тип кодирования протокола. На данный момент это только "Бинарный".
- URL конечной точки.

- Сертификат сервера в формате PEM.
- Приватный ключ в формате PEM.
- Политики безопасности сервера.

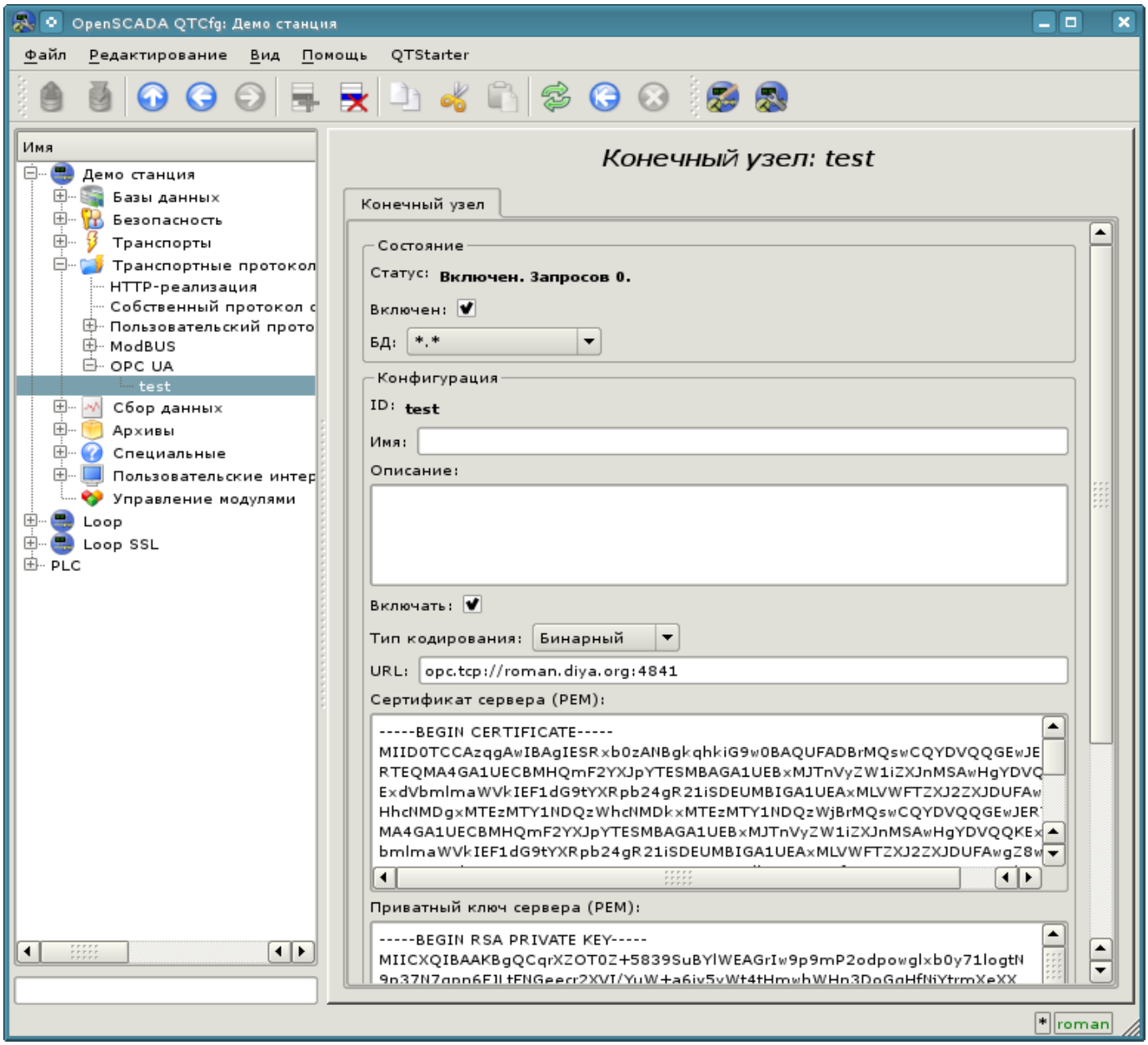


Рис.2. Главная вкладка страницы конечного узла.

3. Модуль сбора данных

Модуль сбора данных предоставляет возможность опроса и записи атрибутов значения(13) узлов типа "Переменная".

3.1. Контроллер данных

Для добавления источника данных OPC UA создаётся и конфигурируется контроллер в системе OpenSCADA. Пример вкладки конфигурации контроллера данного типа изображен на рис.3.

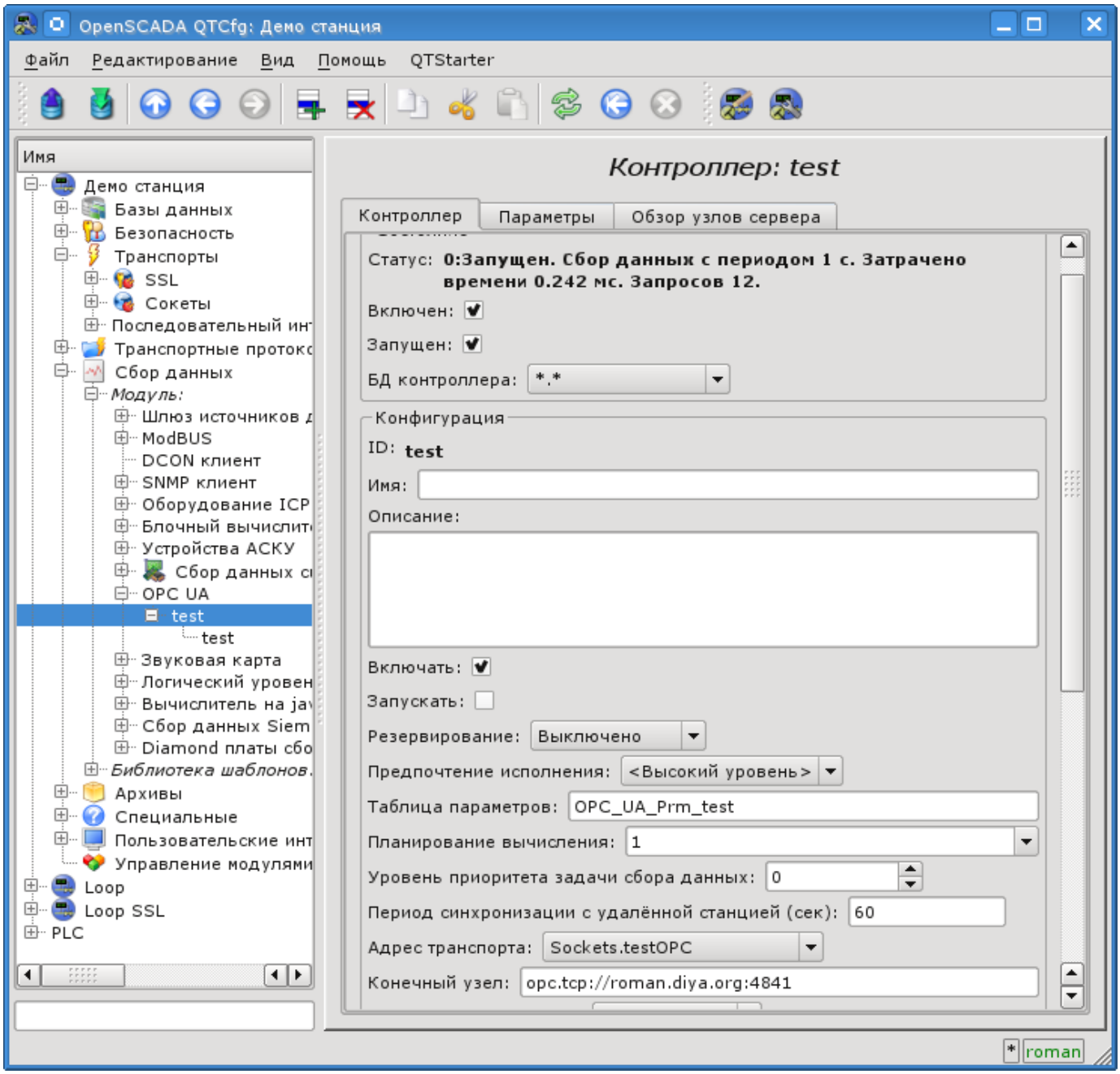


Рис.3. Вкладка конфигурации контроллера.

С помощью этой вкладки можно установить:

- Состояние контроллера, а именно: Статус, "Включен", Запущен" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание контроллера.
- Состояние, в которое переводить контроллер при загрузке: "Включен" и "Запущен".
- Режим горизонтального резервирования и предпочтение исполнения данного контроллера.

- Имя таблицы для хранения конфигурации параметров контроллера.
- Параметры планировщика исполнения задачи сбора и её приоритетность.
- Период синхронизации конфигурации атрибутов параметров с удалённой станцией, а также время повтора попыток восстановления подключения.
- Адрес исходящего транспорта из списка сконфигурированных исходящих транспортов в подсистеме "Транспорты" OpenSCADA.
- URL конечного узла удалённой станции.
- Политика безопасности и режим безопасности сообщения.
- Сертификат клиента и приватный ключ в формате PEM.
- Ограничение числа атрибутов в параметре, для режима импорта всех атрибутов принадлежащих объекту.

С целью облегчения идентификации узлов на удалённой станции, а также выбора их для вставки в параметре контроллера, в объекте контроллера предусмотрена вкладка навигации по узлам удалённой станции, где можно пройти по дереву объектов и ознакомится с их атрибутами (рис.4).

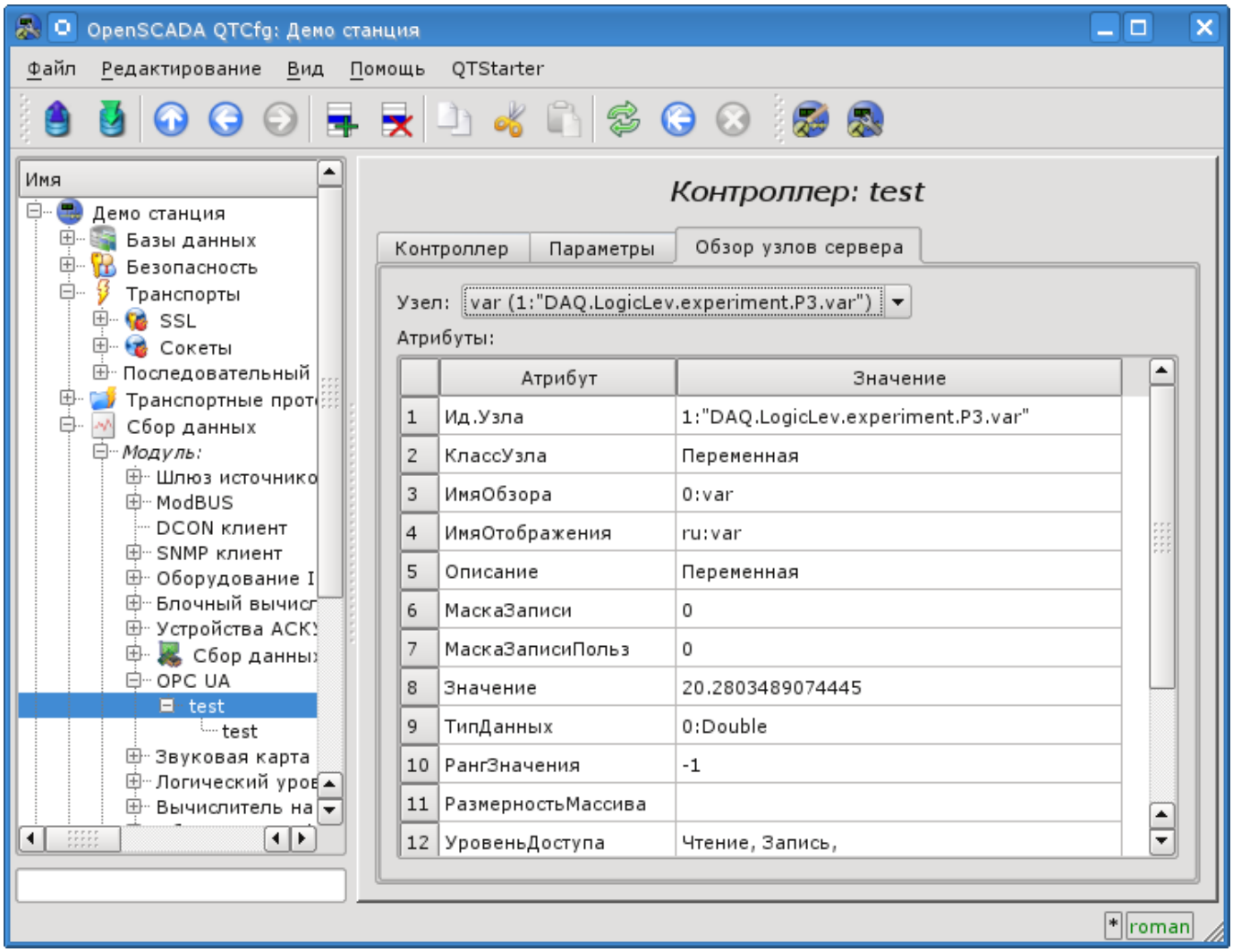


Рис.4. Вкладка "Обзор узлов сервера" страницы контроллера.

3.2. Параметры

Модуль сбора данных предоставляет только один тип параметров - “Стандарт”. Дополнительным конфигурационным полем параметра данного модуля (рис.5) является перечень узлов OPC UA. Атрибут в этом перечне записывается следующим образом: [ns:id].

Где:

ns - область имён, числом; нулевое значение может быть опущено;
id - идентификатор узла, числом, строкой, строкой байт или GUID.

Пример:

84 - корневая директория;
3:"BasicDevices2" - узел базовых устройств в области имён 3 и в виде строки;
4:"61626364" - узел в области имён 4 и в виде строки байт;
4:{40d95ab0-50d6-46d3-bffd-f55639b853d4} - узел в области имён 4 и в виде GUID.

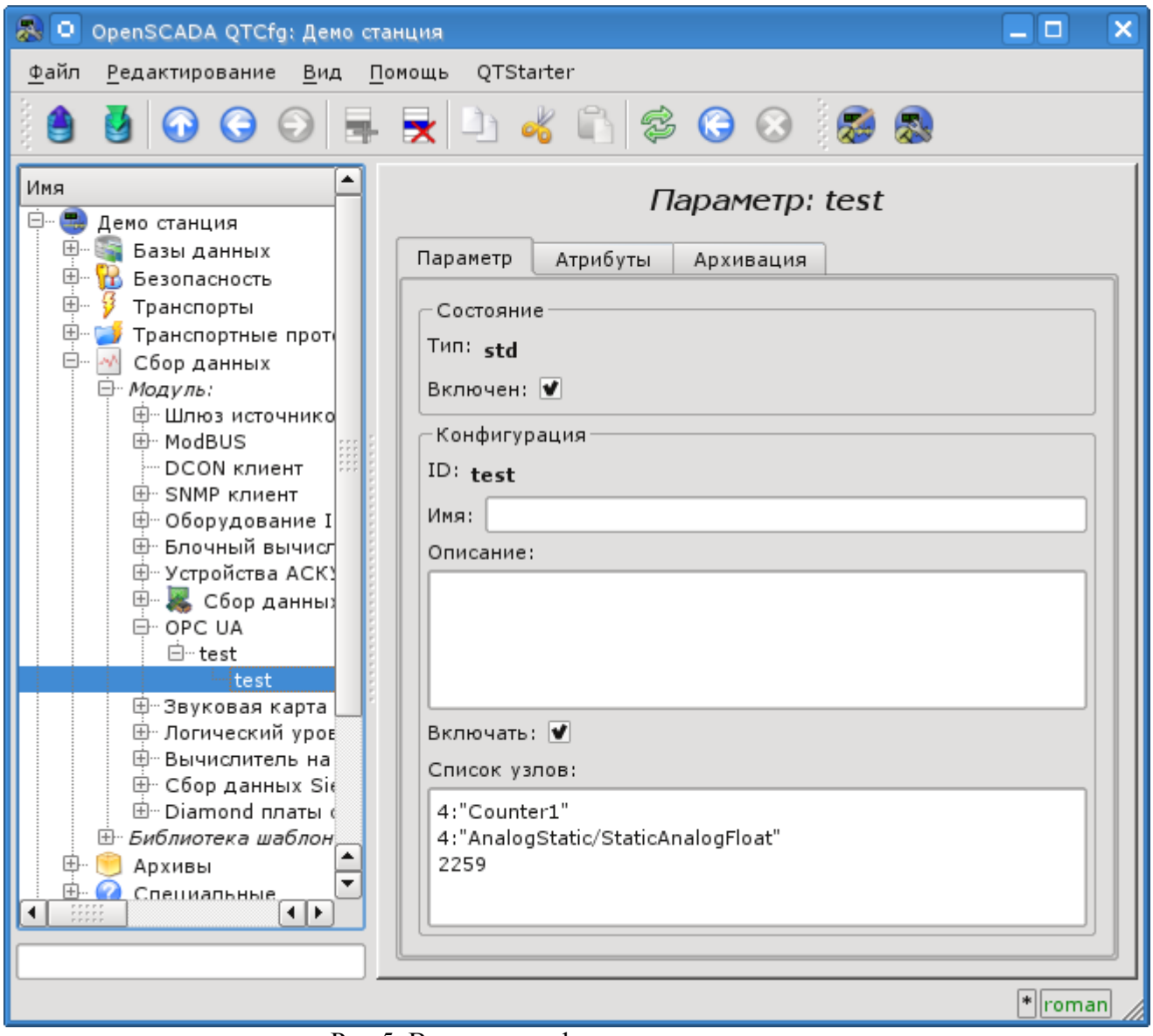


Рис.5. Вкладка конфигурации параметра.

В соответствии с указанным списком узлов выполняется опрос и создание атрибутов параметра (рис.6).

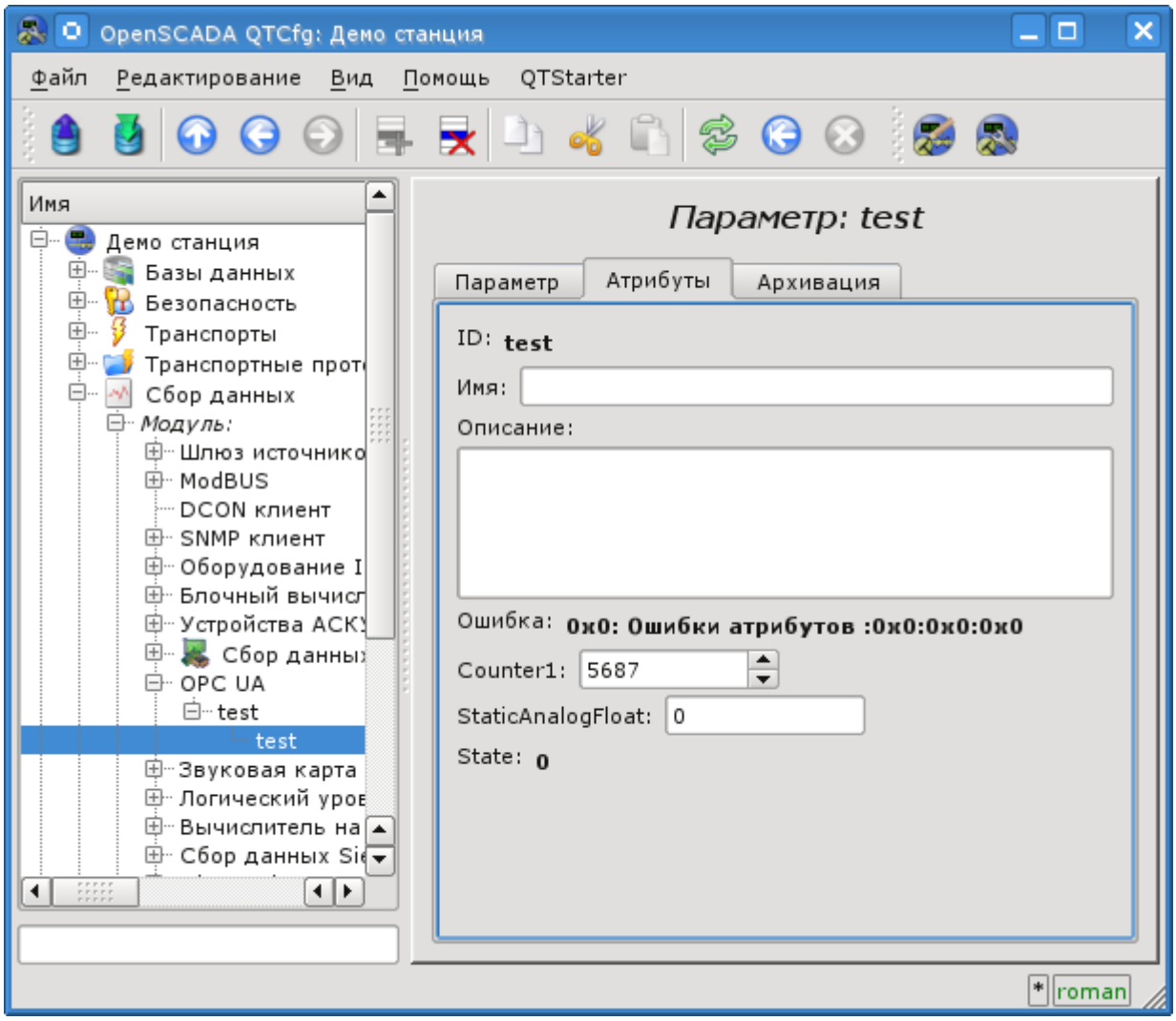


Рис.6. Вкладка атрибутов параметра.

4. Замечания

В процессе реализации модулей поддержки OPC UA был обнаружен ряд несоответствий официального SDK со спецификацией OPC UA:

- OPC UA Part 6 на странице 27 содержит изображение процесса рукопожатия для установления безопасного канала. Пакет создания сессии, исходя из этого процесса, подписывается клиентским симметричным ключём, а кодируется серверным. На самом деле и подпись и шифрование осуществляется серверным ключём.
- OPC UA Part 4 на странице 141 содержит описание структуры данных подписи, где первыми идут данные подписи, а затем строка алгоритма. На самом деле реализован обратный порядок.

Модуль подсистемы “Транспорты” <Sockets>

Модуль:	Sockets
Имя:	Сокеты
Тип:	Транспорт
Источник:	tr_Sockets.so
Версия:	1.4.5
Автор:	Роман Савоченко
Описание:	Предоставляет транспорт, основанный на сокетах. Поддерживаются интернет и unix сокеты. Интернет сокет использует TCP и UDP протоколы.
Лицензия:	GPL

Модуль транспорта Sockets предоставляет в систему поддержку транспортов, основанных на сокетах. Поддерживаются входящие и исходящие транспорты, основанные на интернет сокетах: TCP, UDP и UNIX сокете. Добавить новые входящие и исходящие сокеты можно посредством конфигурации транспортной подсистемы в любом конфигураторе системы OpenSCADA.

1. Входящие транспорты

Сконфигурированный и запущенный входящий транспорт открывает серверный сокет для ожидания соединения клиентов. В случае с UNIX сокетом создаётся файл UNIX сокета. Сокеты TCP и UNIX являются многопоточными, т.е. при подключении клиента к сокетам данных типов создаётся клиентский сокет и новый поток, в котором производиться обслуживание клиента. Серверный сокет в этот момент переходит к ожиданию запросов от нового клиента. Таким образом достигается параллельное обслуживание клиентов.

Каждый входящий сокет обязательно связывается с одним из доступных транспортных протоколов, которому передаются входящие сообщения. В связке с транспортным протоколом поддерживается механизм объединения кусков разрозненных при передаче запросов.

Диалог конфигурации входящего сокета изображён на рис.1.

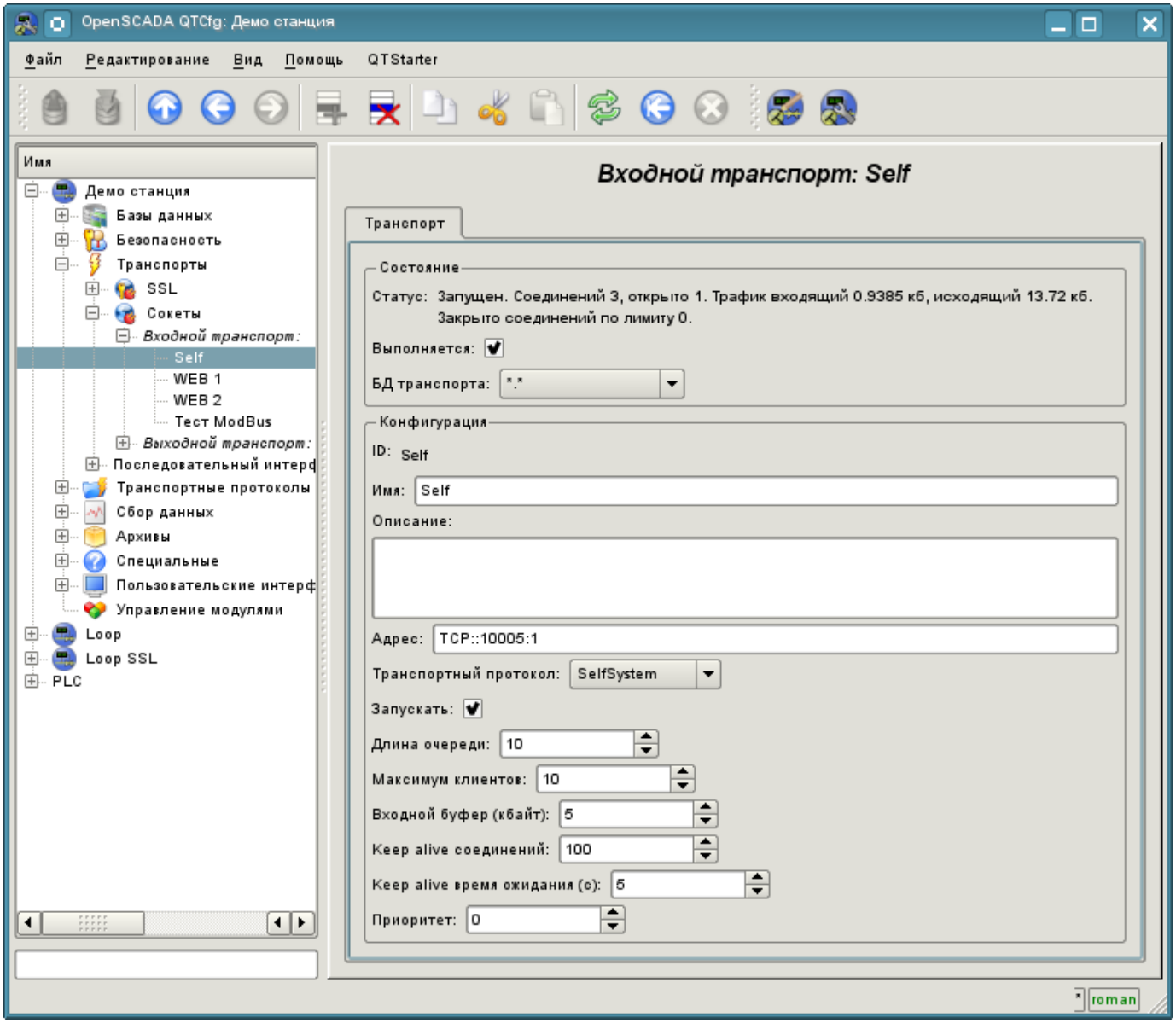


Рис.1. Диалог конфигурации входящего сокета.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: «Статус», «Запущен» и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес транспорта. Формат адреса описан в таблице ниже.
- Выбор транспортного протокола.
- Состояние, в которое переводить контроллер при загрузке: «Запущен».
- Длина очереди сокетов, максимальное количество обслуживаемых клиентов и размер входного буфера.
- Ограничения режима "Keep-alive" по количеству запросов и времени ожидания.
- Приоритет задач транспорта.

Особенности формирования адресов входящих сокетов приведены в таблице ниже:

Тип сокета	Адрес
TCP	*TCP:[адрес]:[порт]:[режим]* где: • адрес – Адрес, на котором открывается сокет. Должен быть одним из адресов хоста. Если ничего не указано, то сокет будет доступен на всех интерфейсах хоста. Допускаются как символьное, так и IP представление адреса. • порт – Сетевой порт, на котором открывается сокет. Возможно указание

	символьного имени порта (в соответствии с /etc/services). • режим – режим работы входящего сокета (0 – разрывать соединение после сеанса приём-ответ; 1 – не разрывать). Пример: *<TCP::10001:1>* – TCP-сокет доступен на всех интерфейсах, открыт на порту 10001 и соединения не разрывает.
UDP	*UDP:[адрес]:[порт]* где: • адрес – тоже что в TCP; • порт – тоже что в TCP. Пример: *<UDP:localhost:10001>* – UDP-сокет доступен только на интерфейсе “localhost” и открыт на порту 10001.
UNIX	*UNIX:[имя]:[режим]* где: • имя – имя файла UNIX сокета; • режим – тоже что в TCP. Пример: *<UNIX:/tmp/oscada:1>* – UNIX-сокет доступен через файл /tmp/oscada и соединения не разрывает.

2. Исходящие транспорты

Сконфигурированный и запущенный исходящий транспорт открывает соединение с указанным сервером. При разрыве соединения исходящий транспорт отключается. Для возобновления соединения транспорт нужно снова запустить.

Главная вкладка страницы конфигурации исходящего сокета изображёна на рис.2.

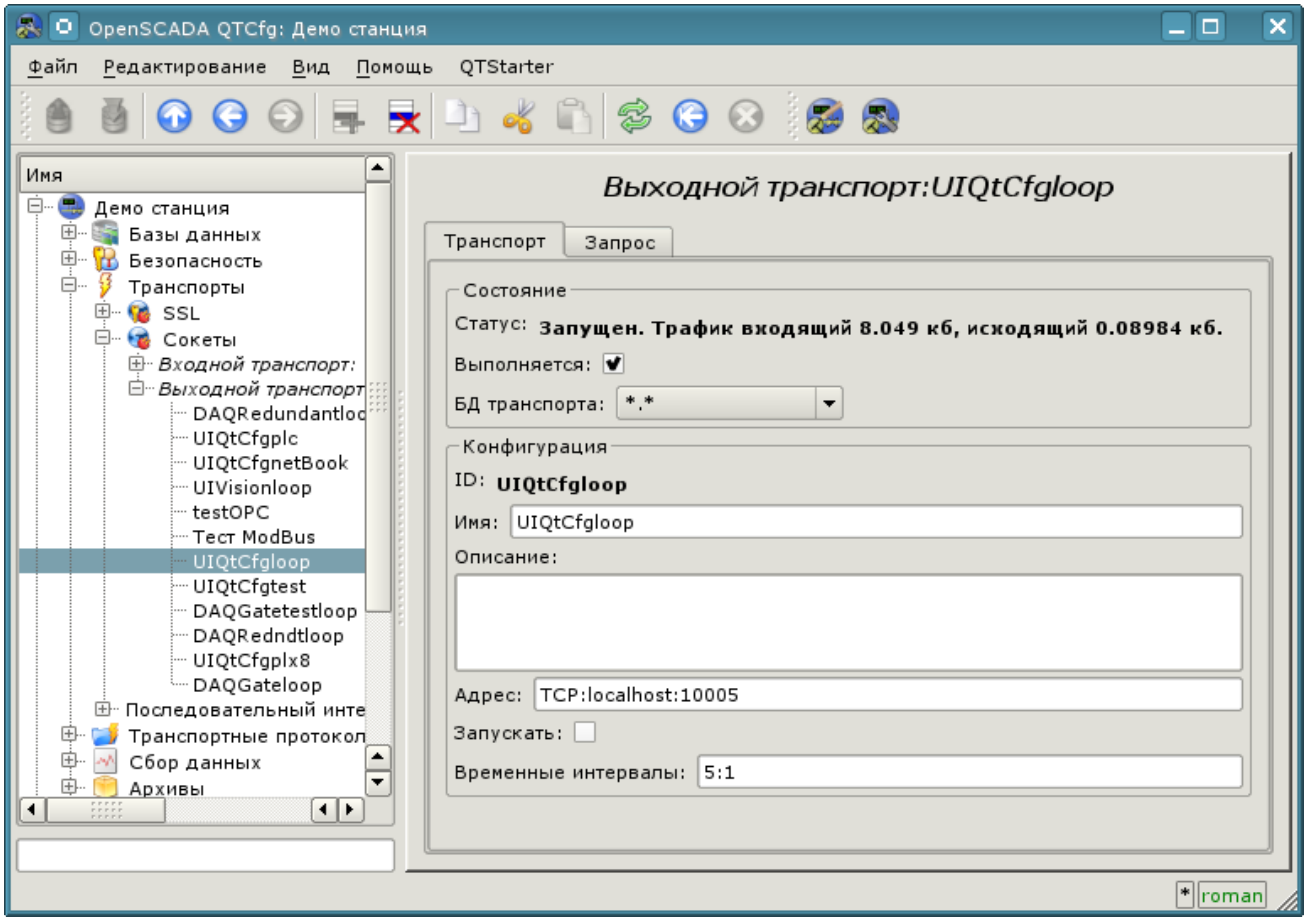


Рис.2. Главная вкладка страницы конфигурации исходящего сокета.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: "Статус", "Запущен" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес транспорта. Формат адреса описан в таблице ниже.

- Состояние, в которое переводить контроллер при загрузке: "Запущен".
- Таймаут по умолчанию для ожидания соединения и ответа, отдельно.

Адреса исходящих сокетов различного типа формируются следующим образом:

Тип сокета	**Адрес**
TCP/UDP	*TCP:[адрес]:[порт] UDP:[адрес]:[порт]* где: • адрес – Адрес, с которым выполняется соединение. Допускаются как символьное так и IP представление адреса. • порт – Сетевой порт, с которым выполняется соединение. Возможно указание символьного имени порта (в соответствии с /etc/services). Пример: <*TCP:127.0.0.1:7634*> – соединится с портом 7634 на хосте 127.0.0.1.
UNIX	*UNIX:[имя]* где: • имя – имя файла UNIX сокета. Пример: <*UNIX:/tmp/oscada*> – соединится с UNIX-сокетом через файл /tmp/oscada.

Модуль подсистемы “Транспорты” <SSL>

Модуль:	SSL
Имя:	SSL
Тип:	Транспорт
Источник:	tr_SSL.so
Версия:	0.9.5
Автор:	Роман Савоченко
Описание:	Предоставляет транспорт основанный на слое безопасных сокетов. Используется OpenSSL и поддерживаютя SSLv2, SSLv3 and TLSv1.
Лицензия:	GPL

Модуль транспорта SSL предоставляет в систему поддержку транспортов основанных на слое безопасных сокетов (SSL). В основе модуля лежит библиотека OpenSSL. Поддерживаются входящие и исходящие транспорты протоколов SSLv2, SSLv3 и TLSv1.

Добавить новые входящие и исходящие транспорты можно посредством конфигурации транспортной подсистемы в любом конфигураторе системы OpenSCADA.

1. Входящие транспорты

Сконфигурированный и запущенный входящий транспорт открывает серверный SSL-сокет для ожидания соединения клиентов. SSL-сокет являются много-поточными, т.е. при подключении клиента создаётся клиентское SSL-соединение и новый поток в котором производиться обслуживание клиента. Серверный SSL-сокет, в этот момент, переходит к ожиданию запросов от нового клиента. Таким образом достигается параллельное обслуживание клиентов.

Каждый входящий транспорт обязательно связывается с одним из доступных транспортных протоколов, которому передаются входящие сообщения. В связке с транспортным протоколом поддерживается механизм объединения кусков раздробленных, при передаче, запросов.

Диалог конфигурации входящего SSL-транспорта изображён на рис.1.

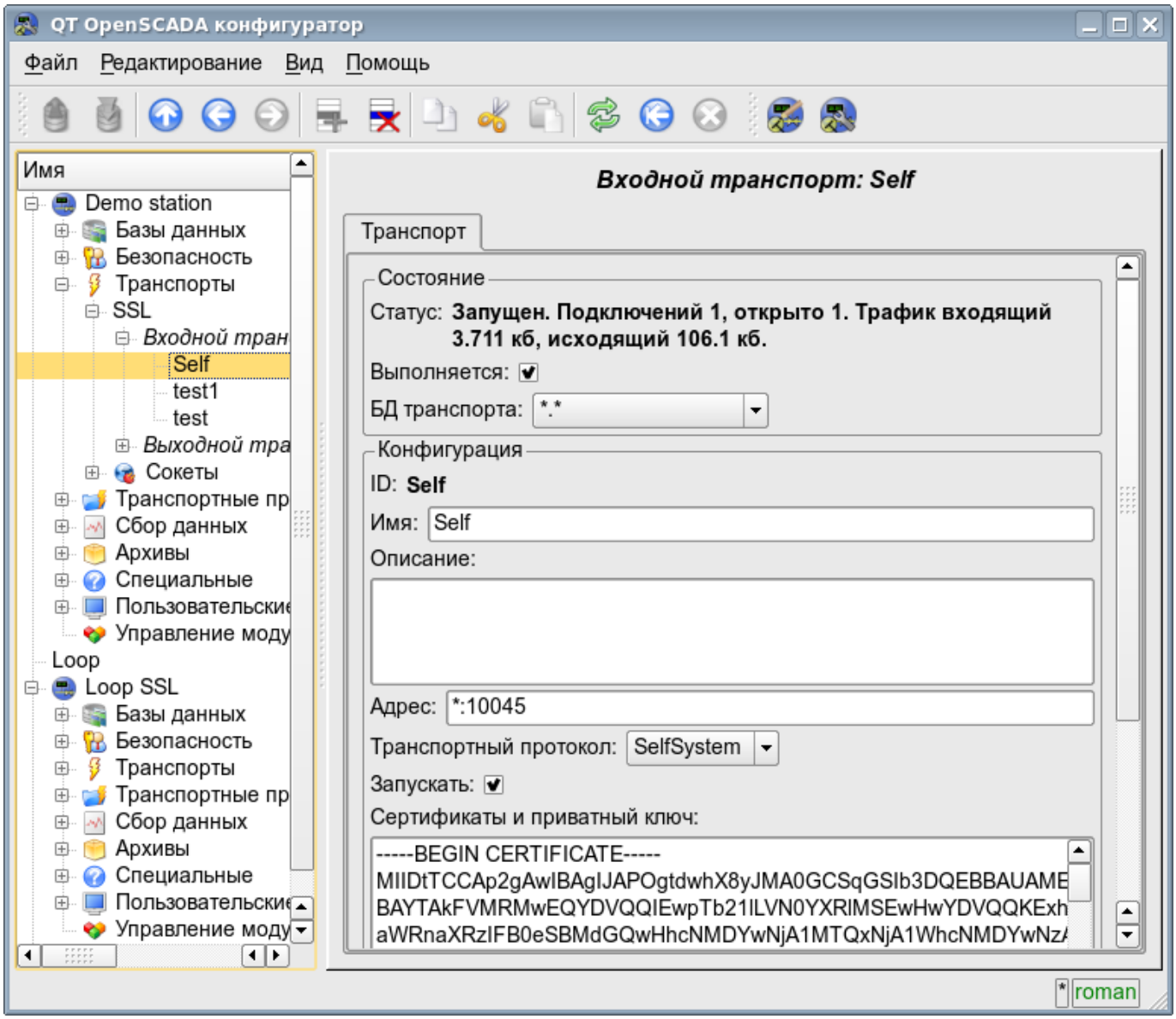


Рис.1. Диалог конфигурации входящего SSL-транспорта.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: «Статус», «Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес транспорта в формате "[адрес]:[порт]:[режим]" где:
  - адрес – Адрес, на котором открывается SSL. Должен быть одним из адресов хоста. Если указано "*" то SSL будет доступен на всех интерфейсах хоста. Допускаются как символьное, так и IP представление адреса.

- порт – Сетевой порт, на котором открывается SSL. Возможно указание символьного имени порта (в соответствии с /etc/services).
- режим – SSL-режим и версия (SSLv2, SSLv3, SSLv23, TLSv1). По умолчанию и при ошибке используется SSLv23

- Выбор транспортного протокола.
- Состояние, в которое переводить транспорт при загрузке: «Запущен».
- Сертификаты, приватный ключ SSL и пароль приватного ключа SSL.
- Максимальное количество обслуживаемых клиентов и размер входного буфера.
- Ограничения режима "Keep-alive" по количеству запросов и времени ожидания.
- Приоритет задач транспорта.

2. Исходящие транспорты

Сконфигурированный и запущенный исходящий транспорт открывает SSL соединение с указанным сервером. При разрыве соединения, исходящий транспорт отключается. Для возобновления соединения транспорт нужно опять запустить.

Главная вкладка страницы конфигурации исходящего SSL-транспорта изображёна на рис.2.

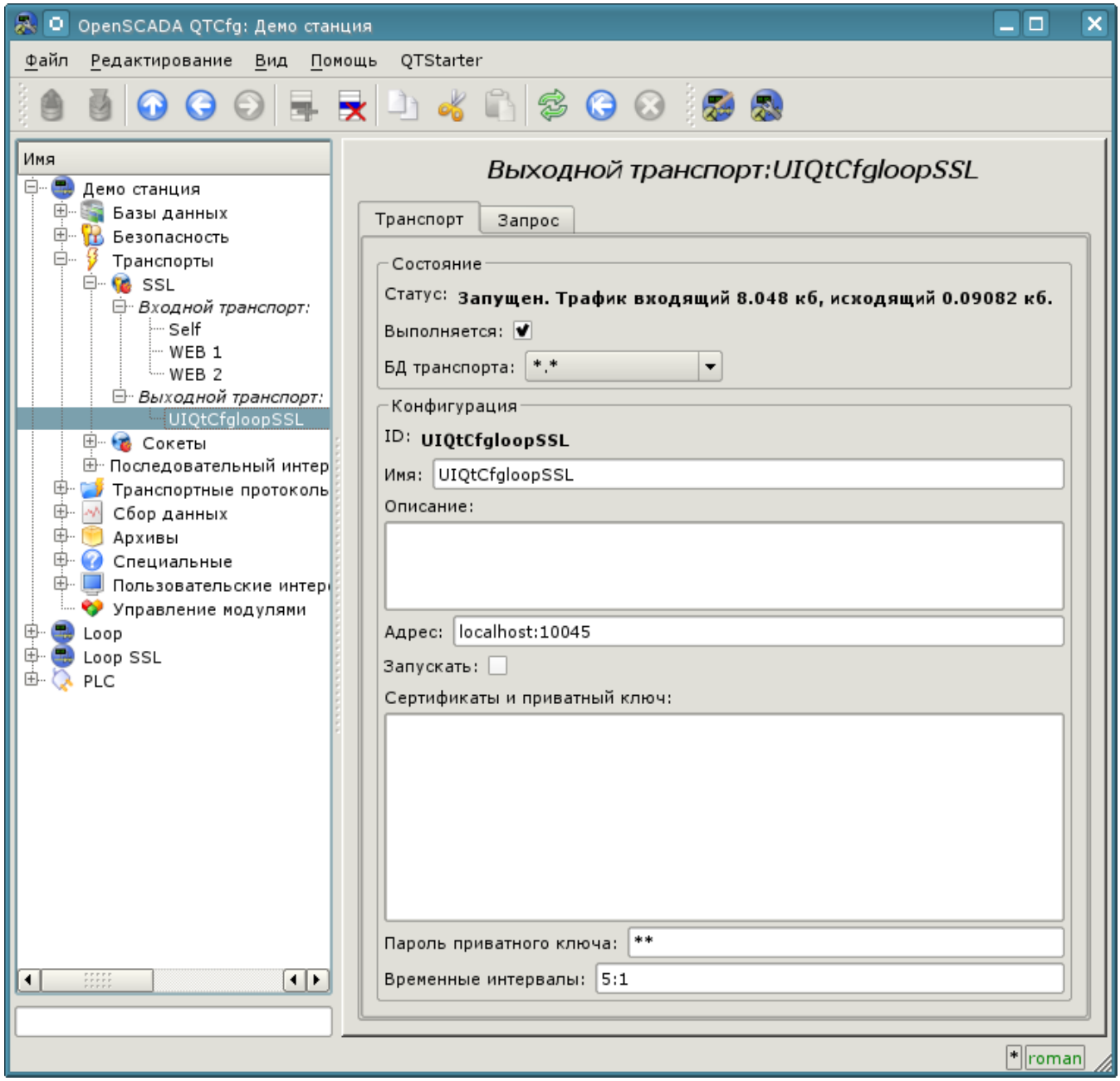


Рис.2. Главная вкладка страницы конфигурации исходящего SSL-транспорта.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: «Статус», «Запущен» и имя БД содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес транспорта в формате "[адрес]:[порт]:[режим]" где:
  - адрес – Адрес, с которым выполняется соединение. Допускаются как символьное так и IP представление адреса.
  - порт – Сетевой порт, с которым выполняется соединение. Возможно указание символьного имени порта (в соответствии с /etc/services).
  - режим – SSL-режим и версия (SSLv2, SSLv3, SSLv23, TLSv1). По умолчанию и при ошибке используется SSLv23
- Состояние, в которое переводить транспорт при загрузке: «Запущен».
- Сертификаты, приватный ключ SSL и пароль приватного ключа SSL.
- Таймаут по умолчанию для ожидания соединения и ответа, отдельно.

3. Сертификаты и ключи

Для полноценной работы модуля необходимы сертификаты и приватные ключи. В случае с входящим SSL-транспортом (сервером) они обязательны. В случае с исходящим SSL-транспортом они могут и не устанавливаться хотя их использование желательно.

Простейшей конфигурацией сертификата является самоподписной сертификат и приватный ключ. Ниже описана процедура их создания с помощью утилиты openssl:

```
# Генерация секретного ключа
$ openssl genrsa -out ./key.pem -des3 -rand /var/log/messages 2048
# Генерация самоподписанного сертификата
$ openssl req -x509 -new -key ./key.pem -out ./selfcert.pem -days 365
```

Далее содержимое файлов selfcert.pem и key.pem копируется в текстовое поле сертификата и ключа. Пароль приватного ключа устанавливается в соответствующем поле.

Модуль подсистемы “Транспорты” <Serial>

Модуль:	Serial
Имя:	Последовательный интерфейс
Тип:	Транспорт
Источник:	tr_Serial.so
Версия:	0.7.1
Автор:	Роман Савоченко
Описание:	Предоставляет последовательный интерфейс. Используется для обмена данными через последовательные интерфейсы типа RS232, RS485, GSM и другое.
Лицензия:	GPL

Модуль транспорта Serial предоставляет в систему поддержку транспортов, основанных на последовательных интерфейсах типа RS232, RS485, GSM и другие. Поддерживаются входящие и исходящие транспорты. Добавить новые входящие и исходящие интерфейсы можно посредством конфигурации транспортной подсистемы в любом конфигураторе системы OpenSCADA.

В режиме модема модулем поддерживается смешанный режим работы. Смешанный режим подразумевает наличие входящего транспорта, который ожидает внешних подключений, а также исходящего транспорта на том-же устройстве. Т.е. входящий транспорт будет игнорировать все запросы при наличии установленного исходящим транспортом соединения, в то-же время исходящий транспорт не будет осуществлять попыток установить соединение при наличии подключения к входящему транспорту или соединения другого исходящего транспорта например, с другим номером телефона.

1. Входящие транспорты

Сконфигурированный и запущенный входящий транспорт открывает порт последовательного интерфейса для ожидания запросов клиентов. Каждый входящий интерфейс обязательно связывается с одним из доступных транспортных протоколов, к которому передаются входящие сообщения.

Диалог конфигурации входящего последовательного интерфейса изображён на рис.1.

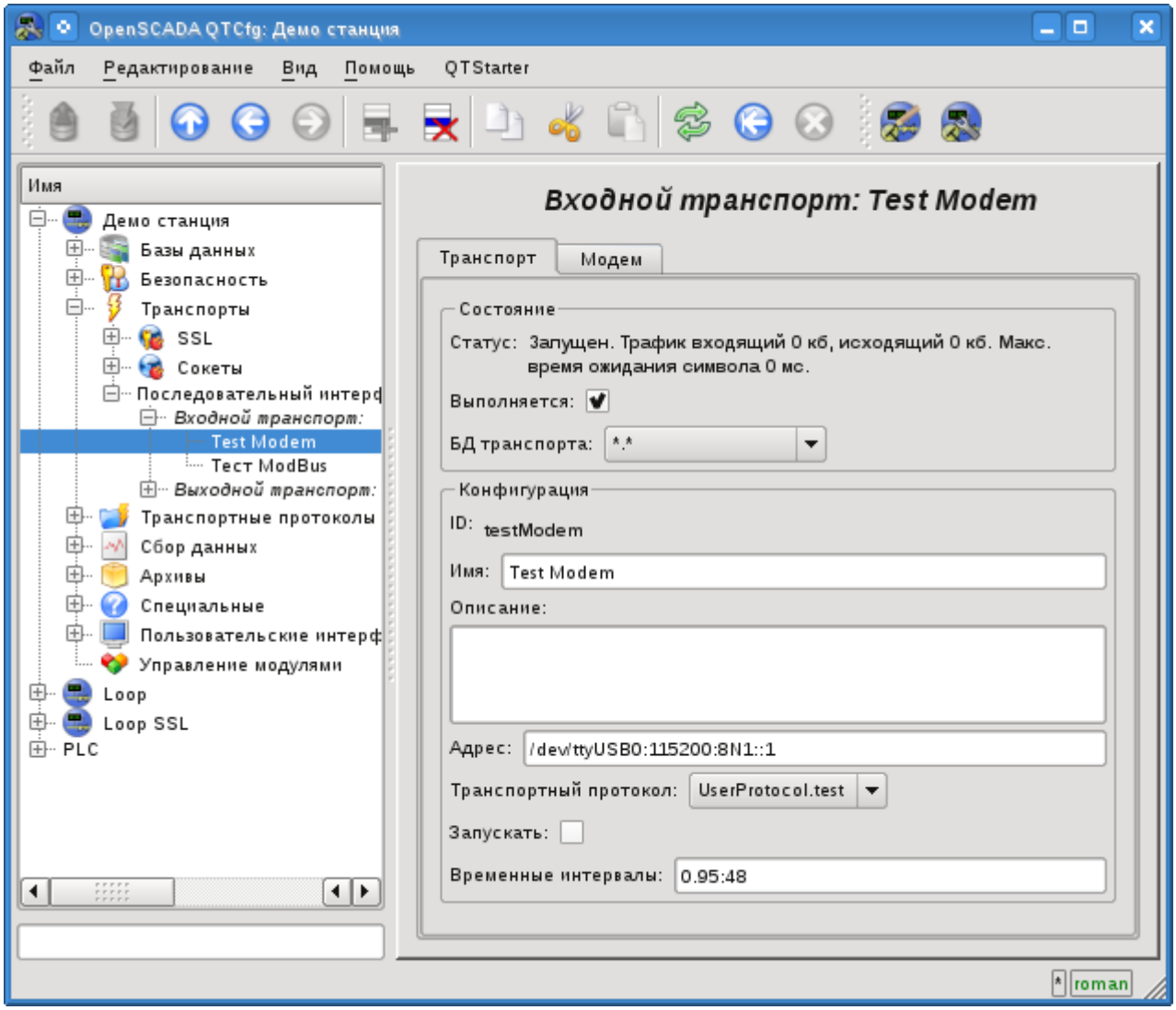


Рис.1. Диалог конфигурации входящего последовательного интерфейса.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: "Статус", "Выполняется" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес интерфейса в формате строки: *"[dev]:[spd]:[format]:[fc]:[mdm]"*. Где:
  - *dev* - адрес последовательного устройства (/dev/ttyS0);
  - *spd* - скорость последовательного устройства из ряда: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 500000, 576000 или 921600;
  - *format* - формат асинхронных данных "<размер><чётность><стоп>" (8N1, 7E1, 5O2, ...);
  - *fc* - управление потоком: 'h' - аппаратное (CRTSCTS), 's' - программное (IXON|IXOFF);
  - *mdm* - режим модема, ожидание 'RING'.
- Выбор транспортного протокола.
- Состояние, в которое переводить транспорт при загрузке: "Запущен".

- Временные интервалы интерфейса в формате строки: *"[symbol]:[frm]"*. Где:
  - *symbol* - время символа в миллисекундах. Используется для контроля факта окончания фрейма;
  - *frm* - максимальное время фрейма в миллисекундах. Используется для ограничение максимального размера пакета запроса (фрейма).

Транспорт поддерживает возможность работы в режиме модема. Данный режим включается пятым параметром адреса и подразумевает ожидания звонка от удалённого модема (запрос "RING"), ответа на звонок (команда "ATA") и последующей передачи запросов от удалённой станции протоколу транспорта. Отключение сеанса связи осуществляется инициатором соединения и приводит к переподключению модема приёмника на ожидание новых звонков.

Для настройки модема входящего транспорта предусмотрена специальная вкладка "Модем" (рис.2).

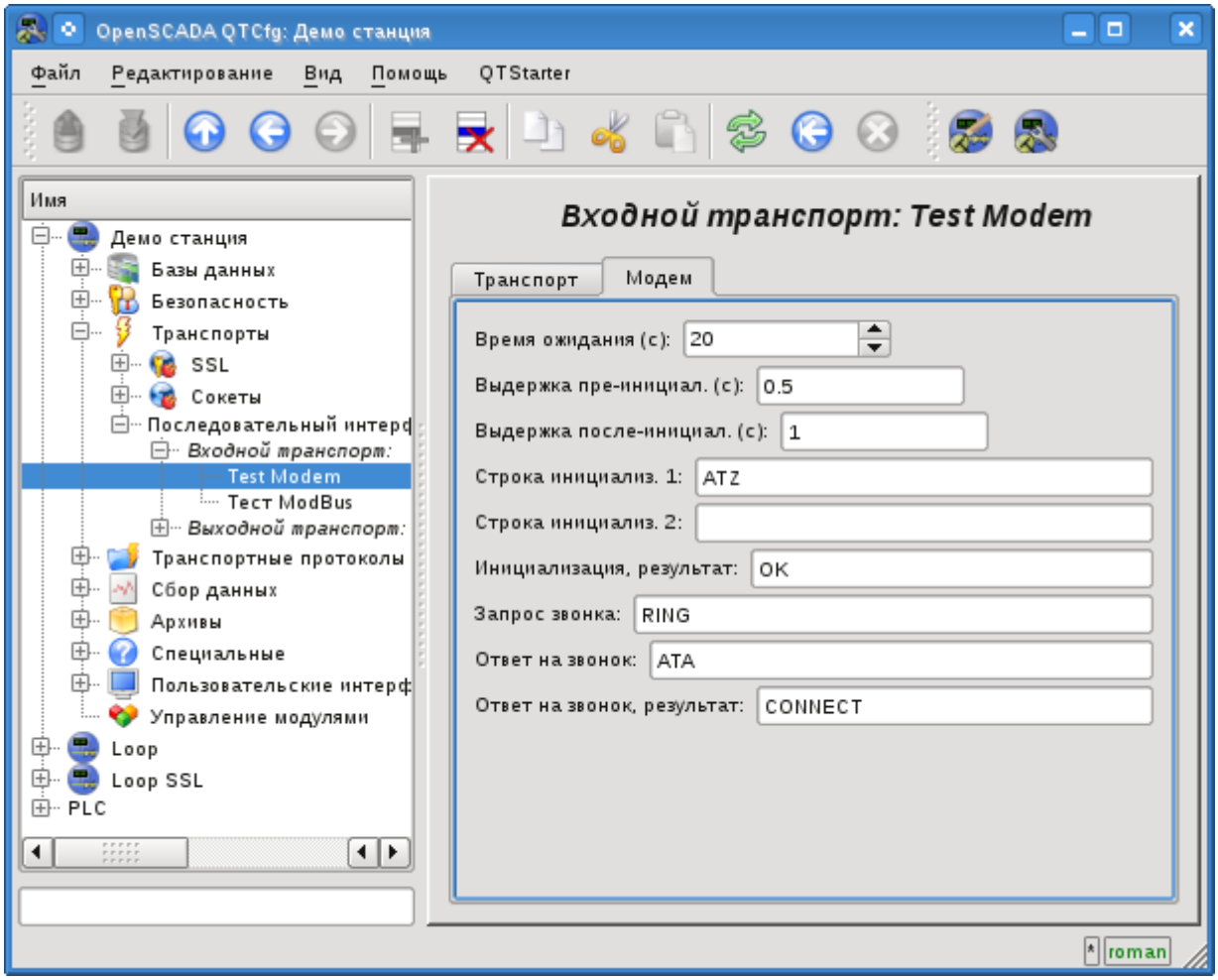


Рис.2. Вкладка "Модем" конфигурации модема входящего последовательного интерфейса.

С помощью этого диалога можно установить следующие свойства работы с модемом:

- Время ожидания, таймаут, модема на запросы, в секундах.
- Выдержка времени перед инициализацией модема, в секундах.
- Выдержка времени после инициализацией модема, в секундах.
- Первая строка инициализации, обычно содержит команду сброса настроек модема "ATZ".
- Вторая строка инициализации.
- Строка результата инициализации модема, обычно "OK", которой отвечает модем на инициализацию и которую нужно ожидать.
- Запрос звонка, обычно "RING", который шлёт модем в случае поступления исходящего вызова.
- Ответ на звонок, обычно "ATA", который отправляется модему для ответа на звонок.
- Строка результата на ответ на звонок, обычно "CONNECT", которой отвечает модем на команду ответа и которую нужно ожидать.

2. Исходящие транспорты

Сконфигурированный и запущенный исходящий транспорт открывает порт последовательного интерфейса для отправки запросов через него.

Главная вкладка страницы конфигурации исходящего последовательного интерфейса изображёна на рис.3.

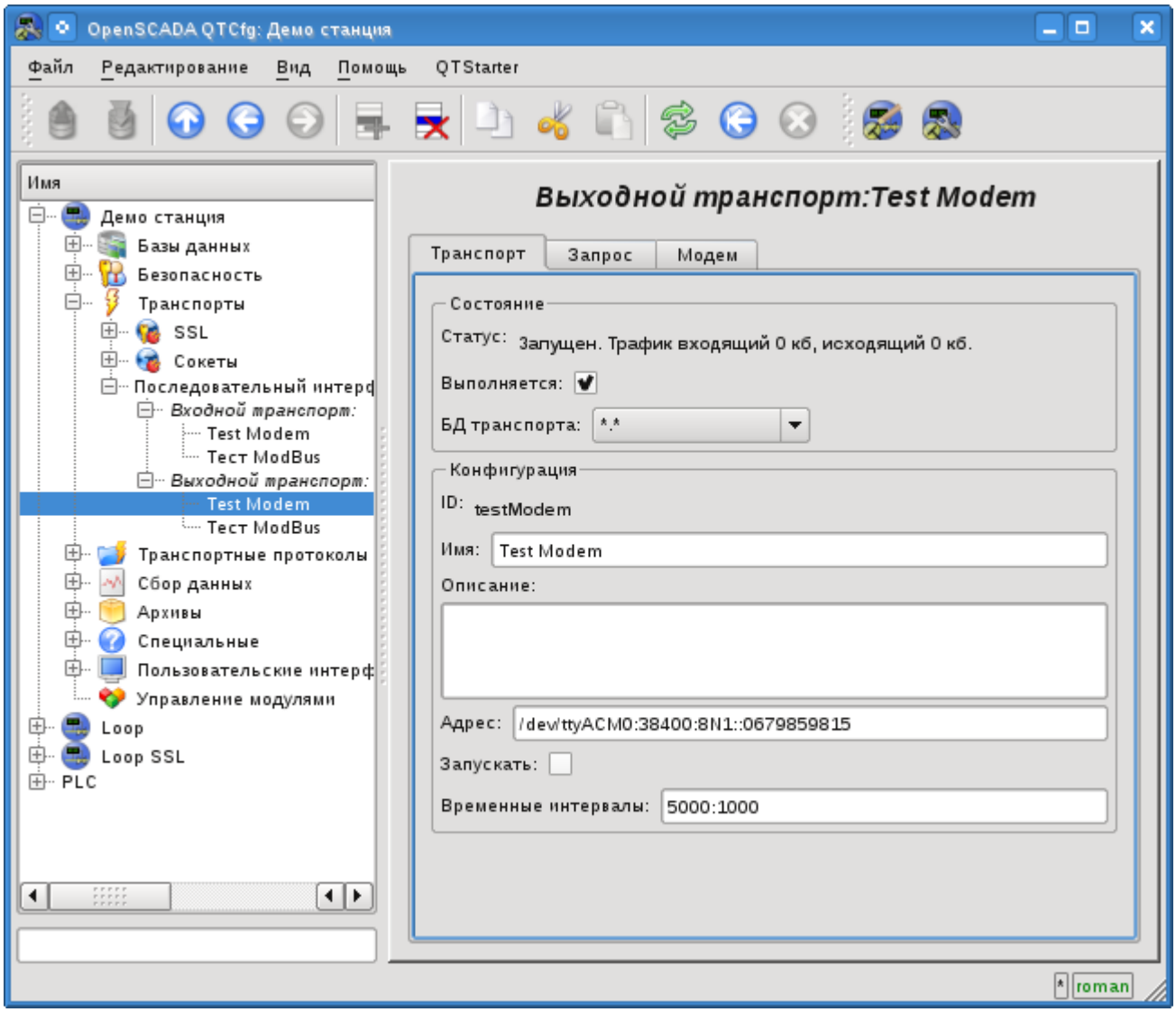


Рис.3. Главная вкладка страницы конфигурации исходящего последовательного интерфейса.

С помощью этого диалога можно установить:

- Состояние транспорта, а именно: "Статус", "Запущен" и имя БД, содержащей конфигурацию.
- Идентификатор, имя и описание транспорта.
- Адрес интерфейса в формате строки: *"[dev]:[spd]:[format]:[fc]:[modTel]"*. Где:
  - *dev* - адрес последовательного устройства (/dev/ttyS0);
  - *spd* - скорость последовательного устройства из ряда: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 500000, 576000 или 921600;
  - *format* - формат асинхронных данных "<размер><чётность><стоп>" (8N1, 7E1, 5O2, ...);
  - *fc* - управление потоком: 'h' - аппаратное (CRTSCTS), 's' - программное (IXON|IXOFF);
  - *modTel* - телефон модема, присутствие этого поля переключает транспорт на работу в режиме модема.
- Состояние, в которое переводить транспорт при загрузке: "Запущен".
- Временные интервалы интерфейса в формате строки: *"[conn]:[symbol]"*. Где:

- *conn* - время ожидания соединения т.е. ответа от удалённого устройства.
- *symbol* - время символа в миллисекундах. Используется для контроля факта окончания фрейма.

Транспорт поддерживает возможность работы в режиме модема. Данный режим включается наличием пятого параметра адреса и подразумевает осуществление звонка по телефону, указанному пятым параметром, в момент запуска транспорта. После установки связи с удалённым модемом все запросы передачи направляются станции за удалённым модемом. Отключение сеанса связи, с остановкой транспорта, осуществляется по таймауту активности.

Для настройки модема исходящего транспорта предусмотрена специальная вкладка "Модем" (рис.4).

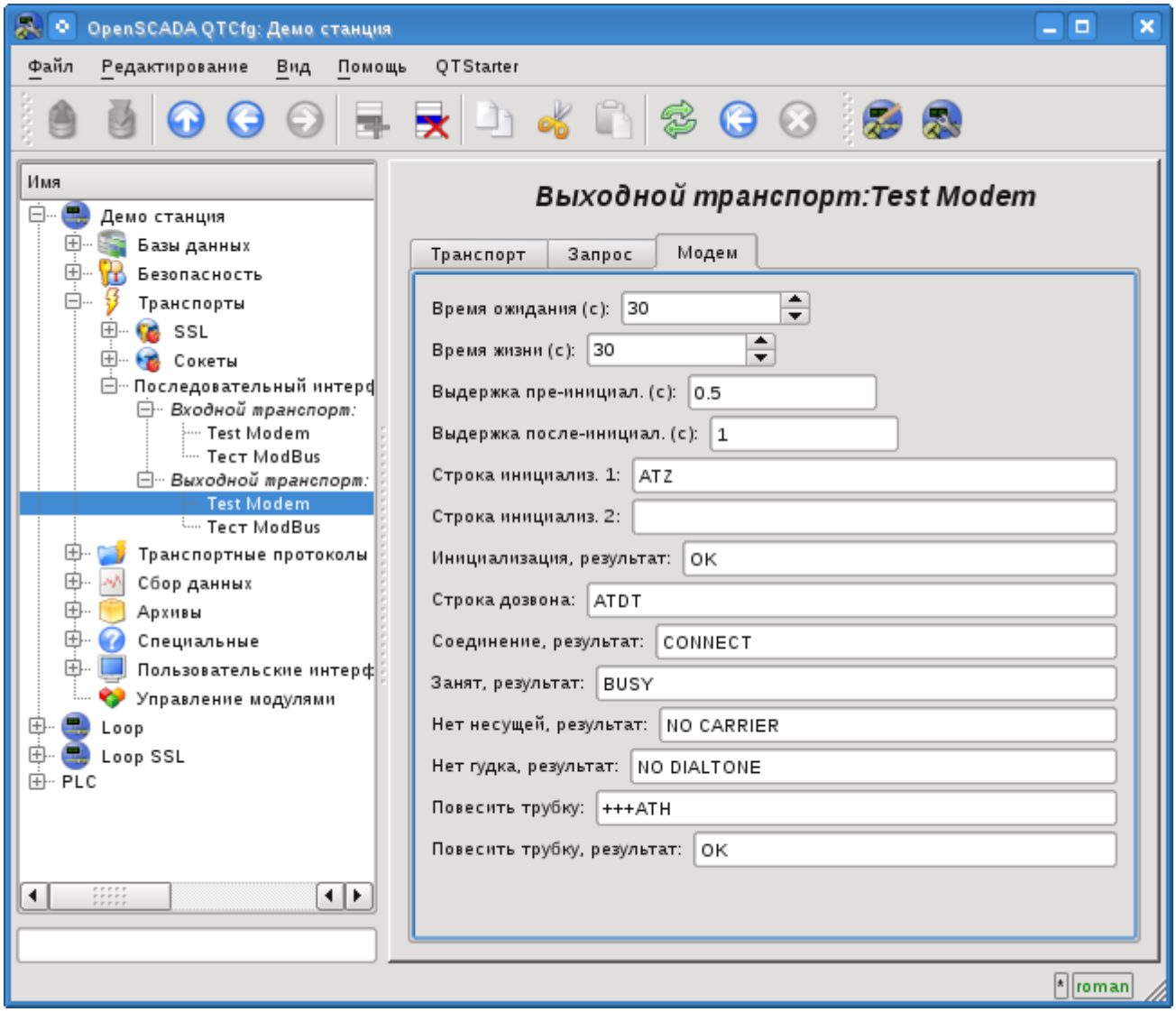


Рис.4. Вкладка "Модем" конфигурации модема исходящего последовательного интерфейса.

С помощью этого диалога можно установить следующие свойства работы с модемом:

- Время ожидания, таймаут, модема на запросы, в секундах.
- Время жизни соединения, в секундах. Если в течении этого времени будет отсутствовать передача данных через транспорт то соединение будет разорвано.
- Выдержка времени перед инициализацией модема, в секундах.
- Выдержка времени после инициализацией модема, в секундах.
- Первая строка инициализации, обычно содержит команду сброса настроек модема "ATZ".
- Вторая строка инициализации.
- Строка результата инициализации модема, обычно "OK", которой отвечает модем на инициализацию и которую нужно ожидать.
- Строка дозвона к удалённому модему, обычно "ATDT". При дозвоне номер телефона

добавляется к данному префиксу.

- Строка результата удачного соединения, обычно "CONNECT".
- Строка результата занятости линии, обычно "BUSY".
- Строка результата отсутствия несущей в линии, обычно "NO CARRIER".
- Строка результата отсутствия гудка линии, обычно "NO DIALTONE".
- Команда повесить трубку, обычно "+++ATH". Данная команда вызывается всегда, когда нужно разорвать соединение.
- Строка результата команды повесить трубку, обычно "OK", которой отвечает модем на команду и которую нужно ожидать.

Модуль подсистемы “Протоколы” <HTTP>

Модуль:	HTTP
Имя:	HTTP
Тип:	Протокол
Источник:	prot_HTTP.so
Версия:	1.5.0
Автор:	Роман Савоченко
Описание:	Предоставляет поддержку протокола HTTP для WWW-основанных пользовательских интерфейсов.
Лицензия:	GPL

Модуль транспортного протокола HTTP предназначен для реализации поддержки сетевого протокола HTTP(Hypertext Transfer Protocol) в системе OpenSCADA.

Протокол HTTP используется для передачи содержимого WWW. Так, через HTTP передаются следующие типы документов: html, xhtml, png, java и многие другие. Добавление поддержки HTTP в систему OpenSCADA в комплексе с транспортом Sockets позволяет реализовывать различные пользовательские функции на основе WWW интерфейса. Модуль HTTP реализует два основных метода протокола HTTP: GET и POST. Модуль HTTP обеспечивает контроль целостности HTTP-запросов и, совместно с транспортом Sockets, позволяет “собирать” целостные запросы из их фрагментов, а также обеспечивать сохранение соединения живым (Keep-Alive).

Для гибкого подключения пользовательских интерфейсов к данному модулю используется модульный механизм в рамках самого модуля "HTTP". В роли модулей используются модули подсистемы “Пользовательские интерфейсы” с дополнительным информационным полем “SubType”, имеющим значение “WWW”.

В запросах к Web ресурсам принято использовать URL(Universal Resource Locator), следовательно URL передаётся как основной параметр через HTTP. Первый элемент запрашиваемого URL используется для идентификации модуля UI. Например URL: http://localhost:10002/WebCfg означает обращение к модулю "WebCfg" на хосте http://localhost:10002. В случае ошибочного указания идентификатора модуля или при обращении вообще без идентификатора "HTTP" модуль генерирует диалог информации о входе и с предоставлением выбора одного из доступных пользовательских интерфейсов. Пример диалога показан на рисунке 1.

Рис.1. Диалог выбора модуля WWW-интерфейса.

1. Аутентификация

Модулем поддерживается аутентификация в системе OpenSCADA при предоставлении доступа к модулям WEB-интерфейсов (рис.2). Диалог формируется на языке XHTML 1.0 Transitional!

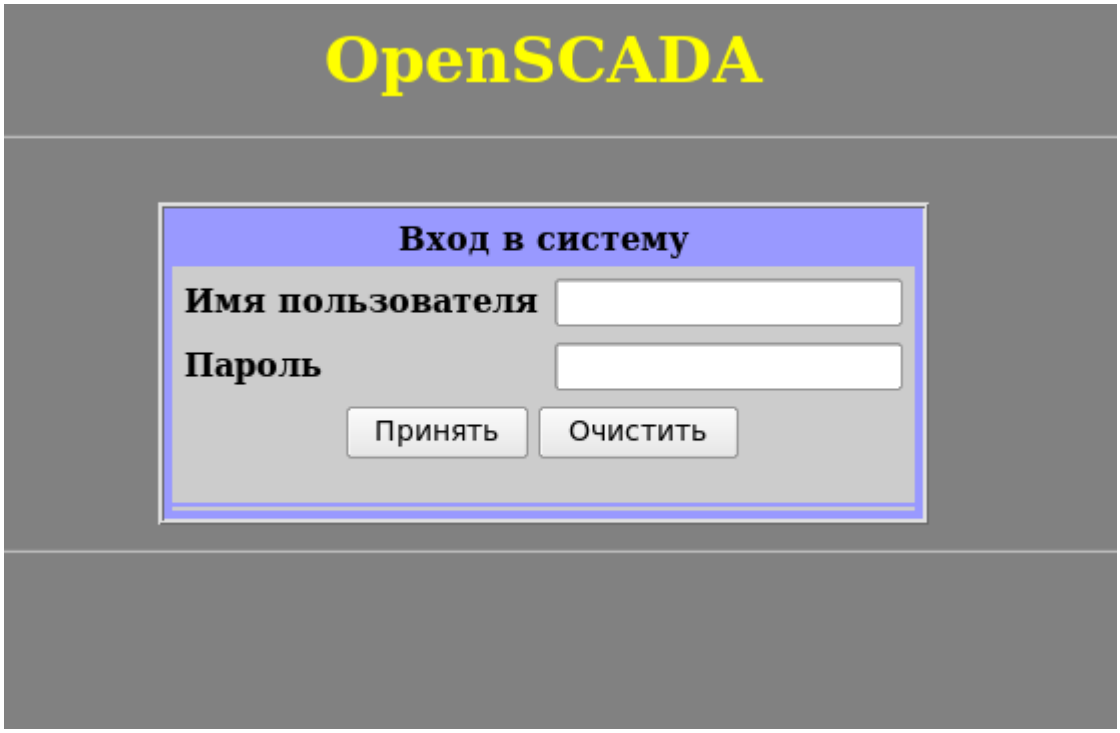


Рис.2. Диалог авторизации в системе OpenSCADA.

Для облегчения работы с Web-интерфейсами модуль предусматривает возможность автоматического входа от имени указанного пользователя. Конфигурация автоматического входа осуществляется на странице настройки модуля (рис.3).

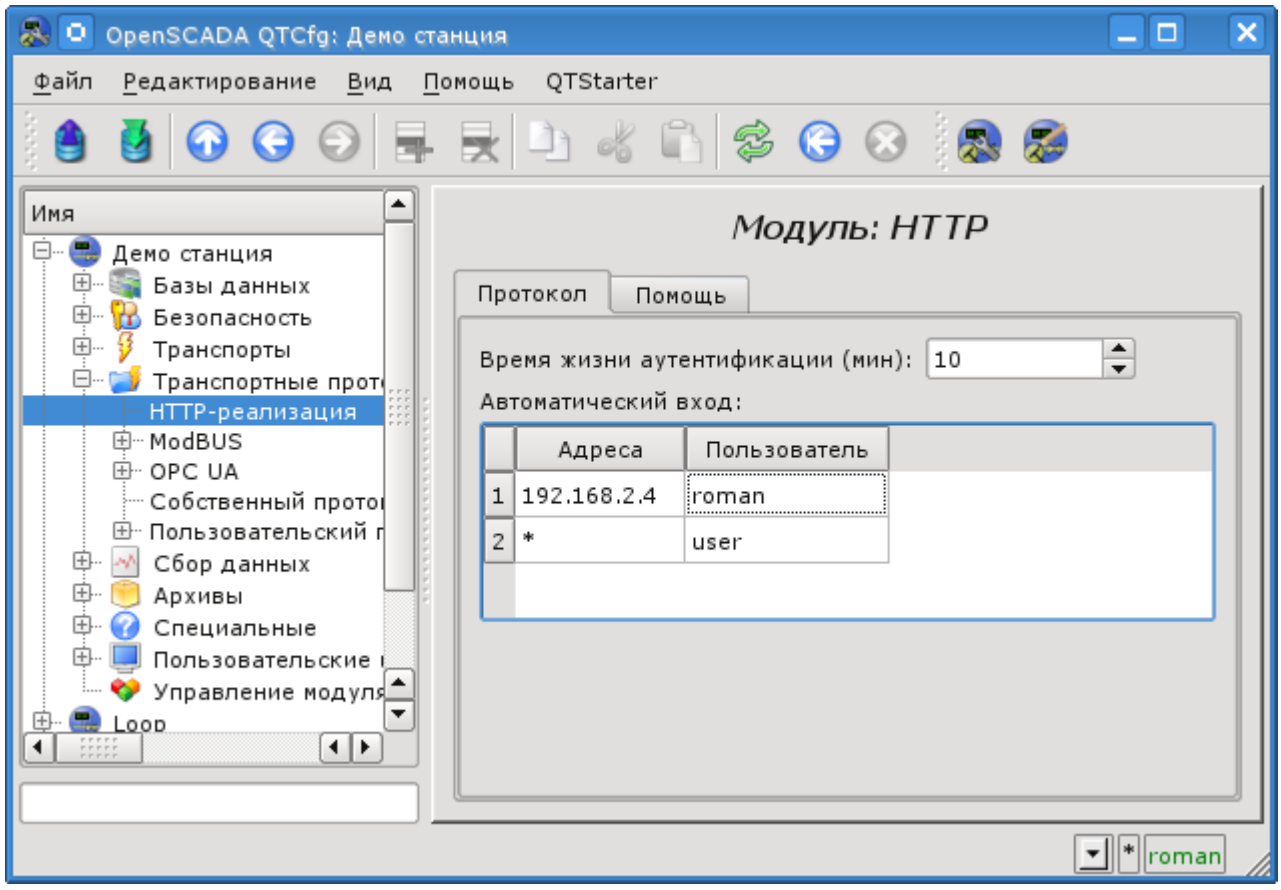


Рис.3. Страница настройки модуля.

На странице настройки модуля можно указать время жизни аутентификации и настроить автоматический вход. Автоматический вход осуществляется по совпадению адреса, указанного в колонке "Адреса", от имени пользователя, указанного в колонке "Пользователь".

2. Модули пользовательского WEB-интерфейса

Модули пользовательского интерфейса (UI), предназначенные для совместной работы с модулем HTTP, должны устанавливать информационное поле “SubType” значением “WWW” и поле "Auth" значением "1", если модуль требует аутентификации при входе. Для связи модуля HTTP и модулей UI используется расширенный механизм коммуникации. Этот механизм подразумевает экспорт интерфейсных функций. В данном случае UI модули должны экспортировать функции:

- *void HttpGet(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);* - Метод GET с параметрами:
  url - адрес запроса; *page* - страница с ответом; *sender* - адрес отправителя; *vars* - переменные запроса; *user* - пользователь системы.
- *void HttpPost(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);* - метод POST с параметрами:
  url – адрес запроса; *page* – страница с ответом и содержимым тела запроса POST; *sender* – адрес отправителя; *vars* – переменные запроса; *user* - пользователь системы.

Далее в случае поступления HTTP запроса "GET" будет вызываться функция "HttpGet", а в случае запроса "POST" будет вызываться функция "HttpPost" в соответствующем модуле UI.

3. API функции исходящих запросов

Функция исходящих запросов оперирует обменом содержимым HTTP-запросов, завёрнутыми в XML-пакеты. Структура запроса имеет вид:

```
<req Host="host" URI="uri">
      <prm id="pId">pVal</prm>
      <cnt name="cName" filename="cFileName">
            <prm id="cpId">cpVal</prm>
            cVal
      </cnt>
</req>
```

Где:

- *req* - метод запроса, поддерживаются методы "GET" и "POST".
- *host* - адрес узла http-сервера в формате *[HostAddr]:[HostIp]*. Если данное поле опущено то используется адрес узла, указанный в поле адреса транспорта.
- *uri* - адрес ресурса, обычно файл или директория, на http-сервере.
- *pId*, *pVal* - идентификатор и значение дополнительного http-параметра. Http-параметров может быть указано множество, отдельными *prm* тегами.
- *cName*, *cFileName*, *cVal* - имя, имя-файла и значение элемента содержимого POST-запроса. Элементов содержимого может быть указано множество, отдельными *cnt* тегами.
- *cpId*, *cpVal* - идентификатор и значение дополнительного параметра содержимого. Параметров содержимого может быть указано множество, отдельными *prm* тегами.

Результатом запроса является ответ со структурой:

```
<req Host="host" URI="uri" err="err" Protocol="prt" RezCod="rCod" RezStr="rStr">
      <prm id="pId">pVal</prm>
      respVal
</req>
```

Где:

- *req* - метод запроса.
- *host* - адрес узла http-сервера.
- *uri* - адрес ресурса.
- *err* - ошибка, возникшая во время запроса. В случае успешного запроса это поле пустое.
- *RezCod*, *RezStr* - результат запроса в виде кода и текста.
- *pId*, *pVal* - идентификатор и значение дополнительного http-параметра, ответа. Http-параметров может быть множество, определённый отдельными *prm* тегами.
- *respVal* - содержимое ответа.

В качестве примера использования данной функции в пользовательских процедурах приведём формирование GET и POST запросов на языке JavaLikeCalc.JavaScript:

```
//GET запрос
req = SYS.XMLNode("GET");
req.setAttr("URI","/");
SYS.Transport.Sockets.out_testHTTP.messIO(req,"HTTP");
test = req.text();

//POST запрос
req = SYS.XMLNode("POST");
req.setAttr("URI","/WebUser/FlowTec.txt");
cntNode = req.childAdd("cnt").setAttr("name","pole0").setAttr("filename","Object2-
k001-100309-17.txt");
cntNode.childAdd("prm").setAttr("id","Content-Type").setText("text/plain");
cntText = "Object2-k001\r\n";
cntText += "\r\n";
cntText += "v002\r\n";
cntText += " n1\r\n";
```

```
cntText += "  09.03.10 16   Polnyj    7155.25    216.0  32.000  17.5\r\n";
cntText += "v005\r\n";
cntText += " n1\r\n";
cntText += "  09.03.10 16   Polnyj     188.81    350.0   4.000  40.0\r\n";
cntText += "\r\n";
cntNode.setText(cntText);
SYS.Transport.Sockets.out_testHTTP.messIO(req,"HTTP");
```

Модуль подсистемы “Протоколы” <SelfSystem>

Модуль:	SelfSystem
Имя:	Собстенный протокол системы OpenSCADA
Тип:	Протокол
Источник:	prot_SelfSystem.so
Версия:	0.9.3
Автор:	Роман Савоченко
Описание:	Собственый протокол системы OpenSCADA, поддерживает основные функции.
Лицензия:	GPL

Модуль транспортного протокола SelfSystem предназначен для отражения интерфейса управления системы OpenSCADA в сеть с целью предоставления возможности внешним системам взаимодействовать с системой OpenSCADA, а также для взаимодействия станций, построенных на основе OpenSCADA между собой.

Первым опытом использования функций данного модуля стала поддержка возможности удалённого конфигурирования одной OpenSCADA станции из другой через сеть посредством модуля конфигурирования QTCfg.

1. Синтаксис протокола

Протокол построен по механизму запрос-ответ. Запросы и их структура сведены в таблице 1.

Таблица 1 Структура запроса.

Запросы
REQ: «SES_OPEN <user> <password>\n» **REZ OK:** «REZ 0 <ses_id>\n» **REZ ERR:** «REZ 1 Auth error. User or password error.\n» Запрос на открытие сессии от имени пользователя *<user>* с паролем *<password>*. В случае удачи будет получен идентификатор сессии, иначе – код и сообщение об ошибке.
REQ: «SES_CLOSE <ses_id>\n» **REZ:** «REZ 0\n» Закрытие сессии. Результат всегда удачен.
REQ 1: «REQ <ses_id> <req_size> \n <control interface command>" **REQ 2:** «REQDIR <user> <password> <req_size> \n <control interface command>" **REZ OK:** «REZ 0 <rez_size> \n <control interface command result>" **REZ ERR:** «REZ 1 Auth error. Session no valid.\n» **REZ ERR:** “REZ 2 <control interface err>" Основные запросы: сеансовый и прямой. Реализуются путём отправки стандартной команды интерфейса управления OpenSCADA в поле *<control interface command>*. В результате будет получен ответ интерфейса управления *<control interface command result>* или одна из ошибок.
REQ: “ERR REQUEST” **REZ ERR:** «REZ 3 Command format error.\n» Любой некорректный запрос.

Протоколом поддерживается возможность упаковки трафика. Пакуются только данные интерфейса управления <control interface command> и <control interface command result>. Факт прихода упакованного запроса или ответа определяется отрицательным значением размера запроса <req_size> или ответа <rez_size>.

Для управления параметрами упаковки модулем предоставляется форма конфигурации (рис.1).

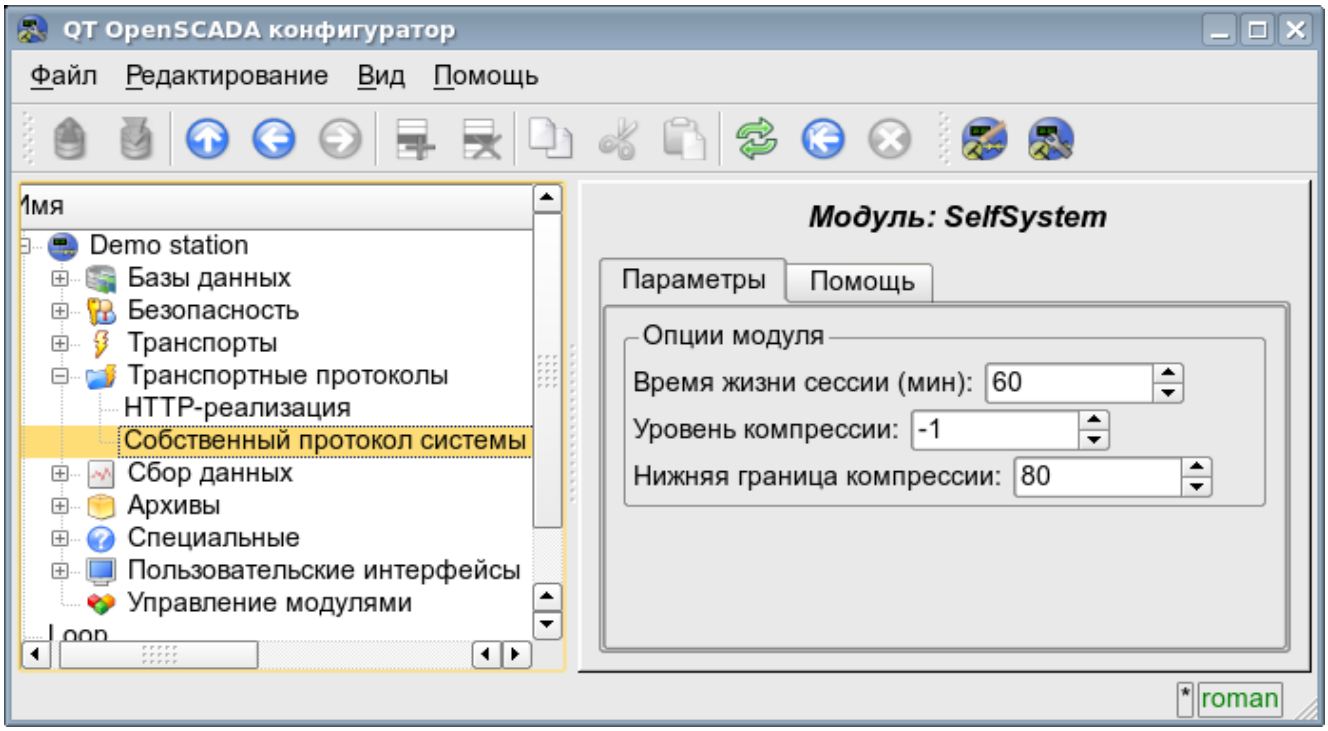


Рис.1. Форма конфигурации параметров упаковки.

На данной форме можно указать:

- время жизни сессии аутентификации;
- уровень компрессии протокола в диапазоне от 0 до 9 (0-отключение компрессии, -1-оптимальный по производительности и качеству уровень компрессии);
- нижний порог использования компрессии; выключает компрессию мелких запросов.

2. Внутренняя структура исходящего протокола

Внутренняя структура формируется деревом XML запросов языка интерфейса управления OpenSCADA с резервированием дополнительных служебных атрибутов протокола в корневом теге:

- *rqDir* — признак отправки сообщения, минуя процедуру открытия сеанса (0-открывать сеанс, 1-отсылать сразу);
- *rqUser* — пользователь;
- *rqPass* — пароль.

Результатом запроса также является дерево XML языка интерфейса управления OpenSCADA.

Модуль подсистемы “Протоколы” <UserProtocol>

Модуль:	UserProtocol
Имя:	Пользовательский протокол
Тип:	Протокол
Источник:	prot_UserProtocol.so
Версия:	0.6.0
Автор:	Роман Савоченко
Описание:	Позволяет создавать собственные пользовательские протоколы на любом OpenSCADA языке.
Лицензия:	GPL

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).

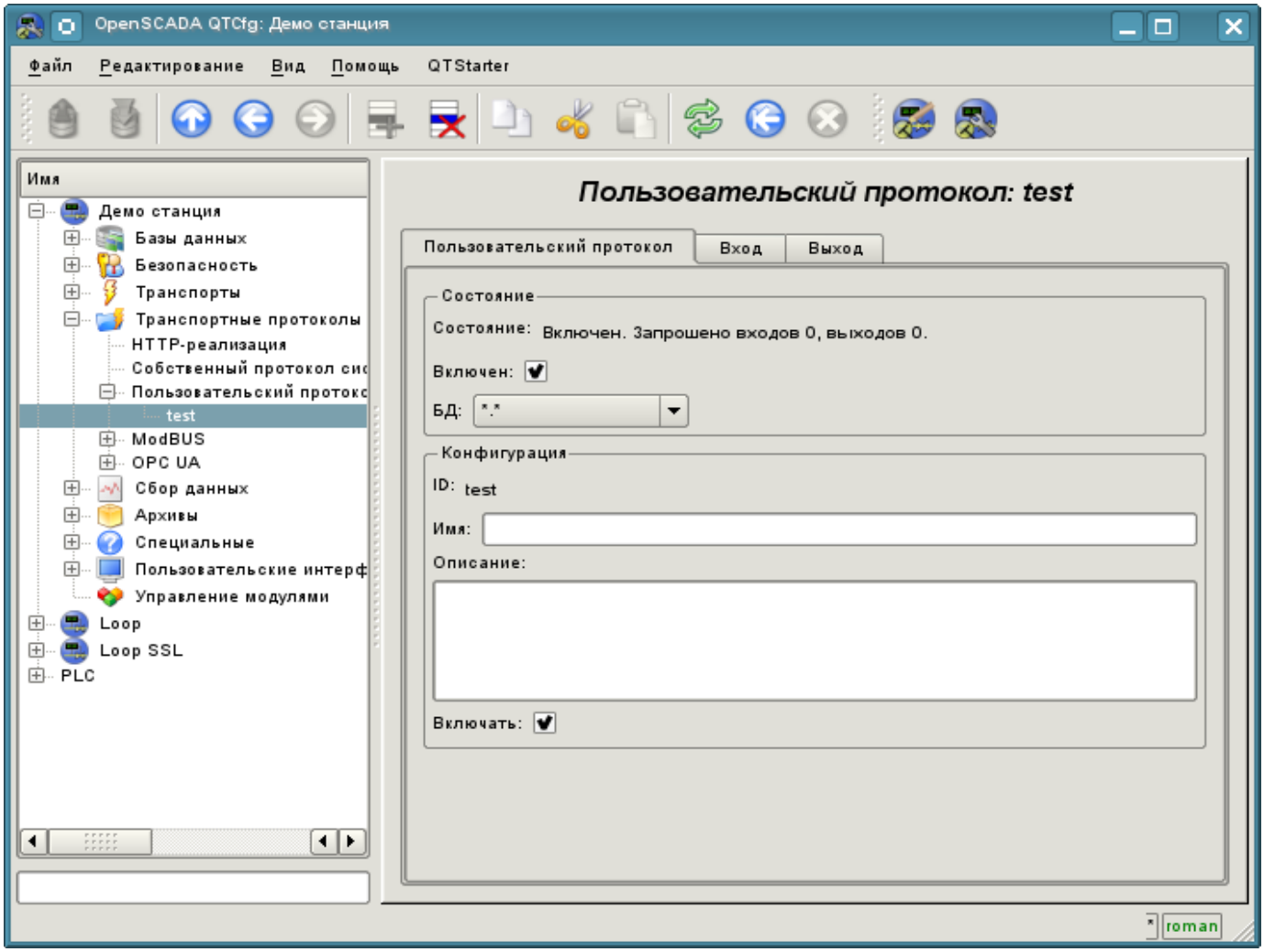


Рис.1. Основная вкладка объекта "Пользовательский протокол".

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).

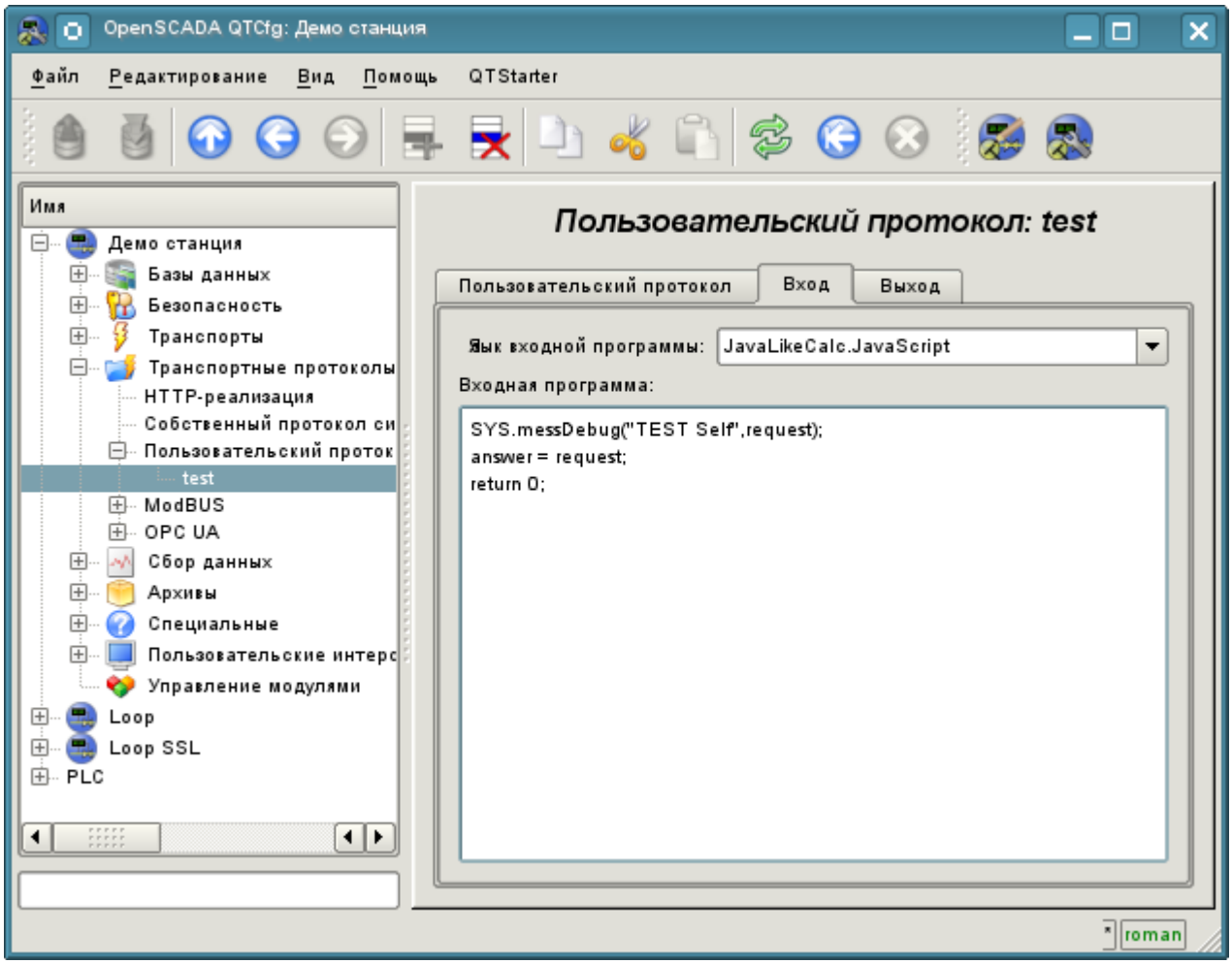


Рис.2. Вкладка процедуры обработки входящих запросов.

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.

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").

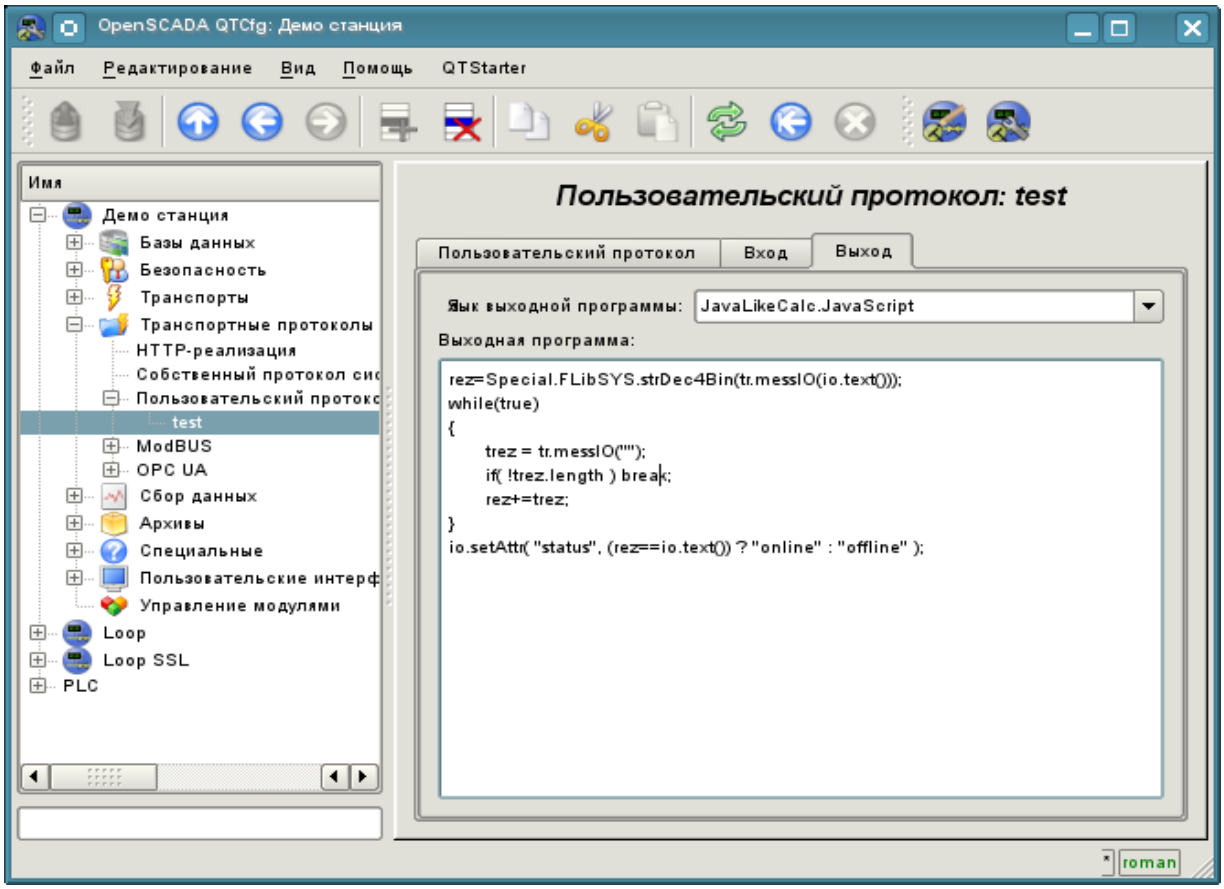


Рис.3. Вкладка процедуры обслуживания исходящих запросов.

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.

Модуль подсистемы “Специальные” <FLibComplex1>

Модуль:	FLibComplex1
Имя:	Библиотека функций совместимости со SCADA Complex1.
Тип:	Специальные
Источник:	spec_FLibComplex1.so
Версия:	1.0.6
Автор:	Роман Савоченко
Описание:	Предоставляет библиотеку функций совместимости со SCADA Complex1 фирмы НИП “DIYA”.
Лицензия:	GPL

Специальный модуль FLibComplex1 предоставляет в систему OpenSCADA статическую библиотеку функций совместимости со SCADA Complex1 фирмы НИП “DIYA”. Эти функции использовались в SCADA системе Complex1 в виде алгоблоков для создания внутрисистемных вычислений на основе виртуального контроллера. Предоставление библиотеки этих функций позволяет выполнять перенос вычислительных алгоритмов из системы Complex1.

Для адресации к функциям этой библиотеки необходимо использовать путь: <Special.FLibComplex1.*>, где '*' – идентификатор функции в библиотеке.

Ниже приведено описание каждой функции библиотеки. Для каждой функции производилась оценка времени исполнения. Измерение производилось на системе со следующими параметрами: Athlon 64 3000+ (ALTLinux 3.0(32бит)) путём замера общего времени исполнения функции при вызове её 1000 раз. Выборка производилась по наименьшему значению из пяти вычислений. Время заключается в угловые скобки и измеряется в микросекундах.

1. Сигнал (alarm) <111>

Описание: Установка признака сигнализации в случае выхода значения переменной за указанную границу.

Формула:

```
out = if(val>max || val<min) true;
      else false;
```

2. Условие '<' (cond_lt) <239>

Описание: Операция ветвления в соответствии с условием “<".

Формула:

```
out = if(in1<(in2_1*in2_2*in2_3*in2_4)) in3_1*in3_2*in3_3*in3_4;
      else in4_1*in4_2*in4_3*in4_4;
```

3. Условие '>' (cond_gt) <240>

Описание: Операция ветвления в соответствии с условием “>".

Формула:

```
out = if(in1>(in2_1*in2_2*in2_3*in2_4)) in3_1*in3_2*in3_3*in3_4;
      else in4_1*in4_2*in4_3*in4_4;
```

4. Полное условие (cond_full) <513>

Описание: Полная проверка условия, включая больше, меньше и равно.

Формула:

```
out = if(in1<(in2_1*in2_2*in2_3*in2_4)) in3_1*in3_2*in3_3*in3_4;
      else if( in1>(in4_1*in4_2*in4_3*in4_4) in5_1*in5_2*in5_3*in5_4;
      else in6_1*in6_2*in6_3*in6_4;
```

5. Дискретный блок (digitBlock) <252>

Описание: Функция содержит алгоритм управления сборками дискретных сигналов для задвижек и насосов, содержащих: признаки “Open”, “Close” и команды “Open”, “Close”, “Stop”. Функция поддерживает работу с импульсными командами, т.е. может снимать сигнал через указанный промежуток времени.

Параметры:

ID	Параметр	Тип	Режим
cmdOpen	Команда «Открыть»	Bool	Выход
cmdClose	Команда «Закрыть»	Bool	Выход
cmdStop	Команда «Стоп»	Bool	Выход
stOpen	Сотояние «Открыт»	Bool	Вход
stClose	Состояние «Закрыт»	Bool	Вход
tCmd	Command hold time (s)	Целый	Вход
frq	Период обсчёта (мс)	Целый	Вход

6. Деление (div) <526>

Описание: Производит деление сборок переменных.

Формула:

```
out = (in1_1*in1_2*in1_3*in1_4*in1_5 + in2_1*in2_2*in2_3*in2_4*in2_5 + in3) /
      (in4_1*in4_2*in4_3*in4_4*in4_5 + in5_1*in5_2*in5_3*in5_4*in5_5 + in6);
```

7. Экспонента (exp) <476>

Описание: Вычисление экспоненты над группой переменных.

Формула:

```
out = exp (in1_1*in1_2*in1_3*in1_4*in1_5 +
           (in2_1*in2_2*in2_3*in2_4*in2_5+in3) /
           (in4_1*in4_2*in4_3*in4_4*in4_5+in5) )
```

8. Расход (flow) <235>

Описание: Вычисление расхода газа.

Формула:

```
f = K1*((K3+K4*x)^K2);
```

9. Итератор (increment) <181>

Описание: Итерационное вычисление с указанием приращения. Коэффициент приращения для разных направлений различный.

Формула:

```
out = if( in1 > in2 ) in2 + in3*(in1-in2); else in2 - in4*(in2-in1);
```

10. Задержка (lag) <121>

Описание: Задержка изменения переменной. Практически это фильтр без привязки ко времени.

Формула:

```
y = y - Klag*( y - x );
```

11. Простое умножение(mult) <259>

Описание: Простое умножение с делением.

Формула:

```
out=(in1_1*in1_2*in1_3*in1_4*in1_5*in1_6)/(in2_1*in2_2*in2_3*in2_4);
```

12. Умножение+деление(multDiv) <468>

Описание: Разветвленное умножение+деление.

Формула:

```
out = in1_1*in1_2*in1_3*in1_4*in1_5*(in2_1*in2_2*in2_3*in2_4*in2_5+
      (in3_1*in3_2*in3_3*in3_4*in3_5)/(in4_1*in4_2*in4_3*in4_4*in4_5));
```

13. ПИД регулятор (pid) <745>

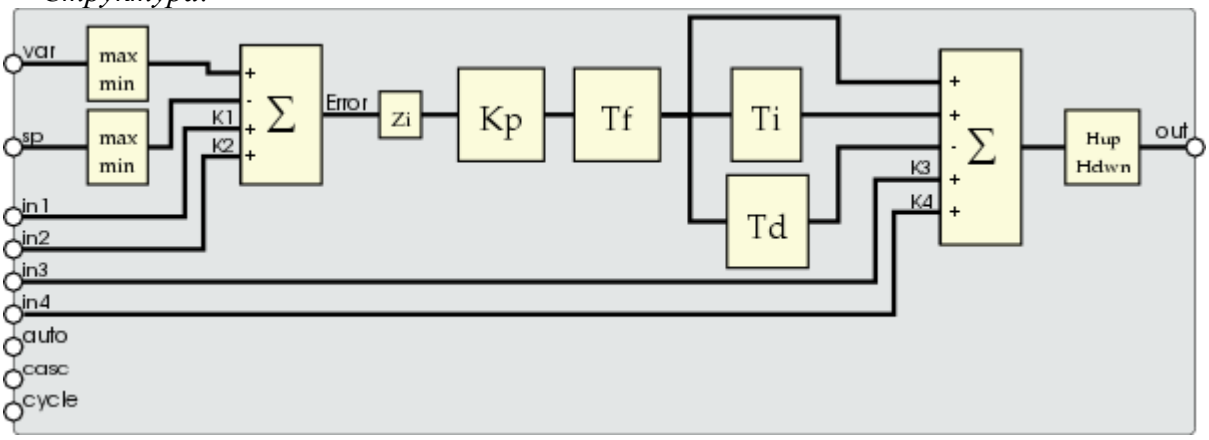
Описание: Пропорционально-интегрально-дифференциальный регулятор.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
var	Переменная	Веществен.	Вход	0
sp	Задание	Веществен.	Выход	0
max	Макс. шкалы	Веществен.	Вход	100
min	Мин. шкалы	Веществен.	Вход	0
manIn	Ручной ввод (%)	Веществен.	Вход	0
out	Выход (%)	Веществен.	Возврат	0
auto	Автомат	Логический	Вход	0
casc	Каскад	Логический	Вход	0
Kp	Kp	Веществен.	Вход	1
Ti	Ти (мс)	Целый	Вход	1000
Kd	Kd	Веществен.	Вход	1
Td	Тд (мс)	Целый	Вход	0
Tzd	Т задержки производной (мс)	Целый	Вход	0
Hup	Верхняя граница выхода (%)	Веществен.	Вход	100
Hdwn	Нижняя граница выхода (%)	Веществен.	Вход	0
Zi	Нечувствительность (%)	Веществен.	Вход	1
followSp	Следить заданием за переменной в ручном.	Логический	Вход	1
K1	Коэф. входа 1	Веществен.	Вход	0
in1	Вход 1	Веществен.	Вход	0
K2	Коэф. входа 2	Веществен.	Вход	0
in2	Вход 2	Веществен.	Вход	0
K3	Коэф. входа 3	Веществен.	Вход	0
in3	Вход 3	Веществен.	Вход	0
K4	Коэф. входа 4	Веществен.	Вход	0
in4	Вход 4	Веществен.	Вход	0

ID	Параметр	Тип	Режим	По умолчанию
f_frq	Частота вычисления (Гц)	Вещественный	Вход	1

Структура:



14. Степень (pow) <564>

Описание: Возведение в степень.

Формула:

```
out = (in1_1*in1_2*in1_3*in1_4*in1_5)^(in2_1*in2_2*in2_3*in2_4*in2_5 +
      (in3_1*in3_2*in3_3*in3_4*in3_5)/(in4_1*in4_2*in4_3*in4_4*in4_5));
```

15. Выбор (select) <156>

Описание: Выбор одного из четырёх вариантов.

Формула:

```
out = if( sel = 1 ) in1_1*in1_2*in1_3*in1_4;
      if( sel = 2 ) in2_1*in2_2*in2_3*in2_4;
      if( sel = 3 ) in3_1*in3_2*in3_3*in3_4;
      if( sel = 4 ) in4_1*in4_2*in4_3*in4_4;
```

16. Простой сумматор (sum) <404>

Описание: Простое суммирование с умножением.

Формула:

```
out = in1_1*in1_2+in2_1*in2_2+in3_1*in3_2+in4_1*in4_2+
      in5_1*in5_2+in6_1*in6_2+in7_1*in7_2+in8_1*in8_2;
```

17. Сумма с делением (sum_div) <518>

Описание: Суммирование с делением группы значений.

Формула:

```
out = in1_1*in1_2*(in1_3+in1_4/in1_5) + in2_1*in2_2*(in2_3+in2_4/in2_5)+
      in3_1*in3_2*(in3_3+in3_4/in3_5) + in4_1*in4_2*(in4_3+in4_4/in4_5);
```

18. Сумма с умножением (sum_mult) <483>

Описание: Суммирование с умножением группы значений.

Формула:

```
out = in1_1*in1_2*(in1_3*in1_4+in1_5) + in2_1*in2_2*(in2_3*in2_4+in2_5) +
      in3_1*in3_2*(in3_3*in3_4+in3_5) + in4_1*in4_2*(in4_3*in4_4+in4_5);
```

Модуль подсистемы “Специальные” <FLibMath>

Модуль:	FLibMath
Имя:	Библиотека стандартных математических функций.
Тип:	Специальные
Источник:	spec_FLibMath.so
Версия:	0.5.2
Автор:	Роман Савоченко
Описание:	Предоставляет библиотеку стандартных математических функций.
Лицензия:	GPL

Специальный модуль FLibMath предоставляет в систему библиотеку стандартных математических функций.

Для адресации к функциям этой библиотеки необходимо использовать путь: <Special.FLibMath.*>. Где '*' идентификатор функции в библиотеке.

1. Функции

В таблице 1 приведено описание каждой функции библиотеки. Для каждой функции производилась оценка времени исполнения. Измерение производилось на системе со следующими параметрами: Athlon 64 3000+ (ALTLinux 3.0(32бит)), путём замера общего времени исполнения функции, при вызове её 1000 раз. Выборка производилась по наименьшему значению из пяти вычислений.

Таблица 1: Функции библиотеки стандартных математических функций

Id	Имя	Описание	Время (мкс)
abs	Модуль	Мат. функция – модуль от числа.	81
acos	Арккосинус	Мат. функция – арккосинус.	149
asin	Арксинус	Мат. функция – арксинус.	140
atan	Арктангенс	Мат. функция – арктангенс.	109
ceil	Округл. до большего	Мат. функция – округления до большего целого.	96
cos	Косинус	Мат. функция – косинус.	93
cosh	Косинус гиперболический	Мат. функция – косинус гиперболический.	121
exp	Экспонента	Мат. функция – экспонента.	145
floor	Округл. до меньшего	Мат. функция – округления до меньшего целого	95
if	Условие Если	Функция условие – «Если».	92
lg	Десятичный логарифм	Мат. функция – десятичный логарифм.	168
ln	Натуральный логарифм	Мат. функция – натуральный логарифм.	185
pow	Степень	Мат. функция – возведение в степень.	157
rand	Случ. число	Мат. функция – генератор случайных чисел.	147
sin	Синус	Мат. функция – синус.	127
sinh	Синус гиперболический	Мат. функция – синус гиперболический.	199
sqrt	Корень квадратный	Мат. функция – корень квадратный.	94
tan	Тангенс	Мат. функция – тангенс.	153
tanh	Тангенс гиперболический	Мат. функция – тангенс гиперболический.	177

Модуль подсистемы “Специальные” <FLibSYS>

Модуль:	FLibSYS
Имя:	Библиотека функций системного API.
Тип:	Специальные
Источник:	spec_FLibSYS.so
Версия:	0.9.2
Автор:	Роман Савоченко
Описание:	Предоставляет библиотеку системного API среды пользовательского программирования.
Лицензия:	GPL

Специальный модуль FLibSYS предоставляет в систему OpenSCADA статическую библиотеку функций для работы с системой OpenSCADA, на уровне её системного API. Эти функции могут использоваться в среде пользовательского программирования системы OpenSCADA для организации неординарных алгоритмов взаимодействия.

Для адресации к функциям этой библиотеки необходимо использовать путь: <Special.FLibSYS.*>. Где '*' идентификатор функции в библиотеке.

Ниже приведено описание каждой функции библиотеки. Для каждой функции производилась оценка времени исполнения. Измерение производилось на системе со следующими параметрами: Athlon 64 3000+ (ALTLinux 4.0(32бит)) путём замера общего времени исполнения функции при вызове её 1000 раз. Выборка производилась по наименьшему значению из пяти вычислений с округлением до меньшего целого. Время заключается в угловые скобки и измеряется в микросекундах.

1. Общесистемные функции

1.1. Вызов консольных команд и утилит операционной системы (sysCall)

Описание: Осуществляет вызовы консольных команд ОС. Функция открывает широкие возможности пользователю OpenSCADA путём вызова любых системных программ, утилит и скриптов, а также получения посредством них доступа к огромному объёму системных данных. Например команда “ls -l” вернёт детализированное содержимое рабочей директории.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
com	Команда	Строка	Вход	

Пример:

```
using Special.FLibSYS;
test=sysCall("ls -l");
messPut("Example",0,"Example: "+test);
```

1.2. SQL запрос (dbReqSQL)

Описание: Формирование SQL-запроса к БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Объект(Массив)	Возврат	
addr	Адрес БД	Строка	Вход	
req	SQL-запрос	Строка	Вход	

1.3. Узел XML (xmlNode)

Описание: Создание объекта узла XML.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Объект(XMLNodeObj)	Возврат	
name	Имя	Строка	Вход	

Пример:

```
using Special.FLibSYS;
//Создание обьекта "get" узла XML.
req = xmlNode("get");
//Создание обьекта "get" узла XML с созданием атрибутов.
//sub_DAQ/mod_ModBus/cntr_1/prm_1 - путь согласно структуре проекта
req    =    xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm%2fst
%2fen");
```

1.4. Запрос интерфейса управления (xmlCntrReq)

Описание: Запрос интерфейса управления к системе посредством XML. Обычный запрос записывается в виде *<get path="/OPath/%2felem"/>*. При указании станции осуществляется запрос к внешней станции.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
req	Запрос	Объект(XMLNodeObj)	Выход	
stat	Станция	Строка	Вход	

Пример:

```
using Special.FLibSYS;
//Получение признака "Включен/Выключен" параметра "1" контроллера "1" модуля "ModBus".
//sub_DAQ/mod_ModBus/cntr_1/prm_1 - путь согласно структуре проекта
req    =    xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm%2fst
%2fen");
rez = xmlCntrReq(req);
messPut("test",0,"Значение: "+req.text());

//Установка признака "Включен" параметра "1" контроллера "1" модуля "ModBus".
req    =    xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm%2fst
%2fen").setText(1);
rez = xmlCntrReq(req);

//Установка признака "Выключен" параметра "1" контроллера "1" модуля "ModBus".
req    =    xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm%2fst
%2fen").setText(0);
rez = xmlCntrReq(req);
```

1.5. Архив значений (vArh)

Описание: Получение объекта архива значений (VArchObj) путём подключения к архиву по его адресу.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Объект(VArchObj)	Возврат	
name	Имя, адрес к атрибуту параметра с архивом или непосредственно к архиву значений.	Строка	Вход	

Объект VArchObj

Функции:

- *begin(usec, archivator)* — Получение времени начала архива путём возврата секунд и микросекунд <*usec*> для архиватора <*archivator*>.
- *end(usec, archivator)* — Получение времени окончания архива путём возврата секунд и микросекунд <*usec*> для архиватора <*archivator*>.
- *period(usec, archivator)* — Получение периодичности архива путём возврата секунд и микросекунд <*usec*> для архиватора <*archivator*>.
- *get(sec, usec, upOrd, archivator)* – Получение значения из архива на время <*sec*>:<*usec*> с привязкой к верху <*upOrd*> и для архиватора <*archivator*>. Реальное время полученного значения устанавливается в <*sec*>:<*usec*>.
- *set(val, sec, usec)* — Запись значения <*val*> в буфер архива на время <*sec*>:<*usec*>.
- *copy(src, begSec, begUSec, endSec, endUSec, archivator)* — Копирование части исходного <*src*> архива или его буфера в текущий начиная с <*begSec*>:<*begUSec*> и заканчивая <*endSec*>:<*endUSec*> для архиватора <*archivator*>.
- *FFT(tm, size, archivator, tm_usec)* - Выполняет разложение в ряд Фурье с помощью FFT алгоритма. Возвращается массив амплитуд частот для окна значений из архива с временем начала <*tm*>:<*tm_usec*> (секунды:микросекунды), глубиной в историю архива <*size*> (секунд) и для архиватора <*archivator*>.

Пример:

```
using Special.FLibSYS;
val = vArh(strPath2Sep(addr)).get(time,uTime,0,archtor);
return val.isEval() ? "Пусто" : real2str(val,prec);
```

1.6. Буфер архива значений (vArhBuf)

Описание: Получение объекта буфера архива значений (VArchObj) для выполнения промежуточных операций над кадрами данных.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Объект(VArchObj)	Возврат	
tp	Тип значений архива (0-Boolean, 1-Integer, 4-Real, 5-String)	Целый	Вход	1
sz	Максимальный размер буфера	Целый	Вход	100
per	Периодичность буфера (в микросекундах)	Целый	Вход	1000000
hgrd	Режим «Жесткая сетка времени»	Логический	Вход	0
hres	Режим «Высокого разрешения времени (микросекунды)"	Логический	Вход	0

2. Функции для работы с астрономическим временем

2.1. Строка времени (tmFStr) <3047>

Описание: Преобразует абсолютное время в строку нужного формата. Запись формата соответствует POSIX-функции strftime.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Строка полной даты	Строка	Возврат	
sec	Секунды	Целый	Вход	0
form	Формат	Строка	Вход	%Y-%m-%d %H:%M:%S

Пример:

```
using Special.FLibSYS;
test=tmFStr(SYS.time(),"%d %m %Y");
messPut("Example",0,"tmFStr(): "+test);
```

2.2. Полная дата (tmDate) <973>

Описание: Возвращает полную дату в секундах, минутах, часах и т.д, исходя из абсолютного времени в секундах от эпохи 1.1.1970.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
fullsec	Полные секунды	Целый	Вход	0
sec	Секунды	Целый	Выход	0
min	Минуты	Целый	Выход	0
hour	Часы	Целый	Выход	0
mday	День месяца	Целый	Выход	0
month	Месяц	Целый	Выход	0
year	Год	Целый	Выход	0
wday	День недели	Целый	Выход	0
yday	День в году	Целый	Выход	0
isdst	Daylight saving time	Целый	Выход	0

Пример:

```
using Special.FLibSYS;
curMin=curHour=curDay=curMonth=curYear=0;
tmDate(tmTime(),0,curMin,curHour,curDay,curMonth,curYear);
messPut("test",0,"Текущая минута: "+curMin);
messPut("test",0,"Текущий час : "+curHour);
messPut("test",0,"Текущий день: "+curDay);
messPut("test",0,"Текущий месяц: "+curMonth);
messPut("test",0,"Текущий год: "+curYear);
```

2.3. Абсолютное время (tmTime) <220>

Описание: Возвращает абсолютное время в секундах от эпохи 1.1.1970 и микросекундах, если <usec> установлен в неотрицательное значение.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
sec	Секунды	Целый	Возврат	0
usec	Микросекунды	Целый	Выход	-1

2.4. Конвертация времени из символьного представлении во время в секундах, от эпохи 1.1.1970 (tmStrPTime) <2600>

Описание: Возвращает время в секундах от эпохи 1.1.1970, исходя из строковой записи времени, в соответствии с указанным шаблоном. Например, шаблону "%Y-%m-%d %H:%M:%S" соответствует время «2006–08–08 11:21:55». Описание формата шаблона можно получить из документации на POSIX-функцию “strptime”.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
sec	Секунды	Целый	Возврат	0
str	Строка даты	Строка	Вход	
form	Формат записи даты	Строка	Вход	%Y-%m-%d %H:%M:%S

Пример:

```
using Special.FLibSYS;
curMin=curHour=curDay=curMonth=curYear=0;
tmDate(tmTime(),0,curMin,curHour,curDay,curMonth,curYear);
test = tmStrPTime(""+curYear+"-"+(curMonth+1)+"-"+curDay+" 9:0:0","%Y-%m-%d %H:%M:%S");
messPut("Example",0,"tmStrPTime(): "+test);
```

2.5. Планирование времени в формате Cron (tmCron)

Описание: Возвращает время спланированное в формате стандарта Cron начиная от базового времени или от текущего если базовое не указано.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
res	Результат	Целый	Возврат	0
str	Запись в стандарте Cron	Строка	Вход	* * * * *
base	Базовое время	Целый	Вход	0

3. Функции работы с сообщениями

3.1. Запрос сообщений (messGet)

Описание: Запрос системных сообщений.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Объект(Массив)	Возврат	
btm	Время начала	Целое	Вход	
etm	Время конца	Целое	Вход	
cat	Категория сообщения	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	
arch	Архиватор	Строка	Вход	

3.2. Генерация сообщения (messPut)

Описание: Формирование системного сообщения.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
cat	Категория сообщения	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	
mess	Текст сообщения	Строка	Вход	

Пример:

```
rnd_sq_gr11_lineClr="red";
Special.FLibSYS.messPut("Example",1,"Event: "+rnd_sq_gr12_leniClr);
```

4. Функции работы с строками

4.1. Получение размера строки (strSize) <114>

Описание: Используется для получения размера.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целый	Возврат	
str	Строка	Строка	Вход	

Пример:

```
Special.FLibSYS.messPut("Example",1,"ReturnString: "+strSize("Example"));
```

4.2. Получение части строки (strSubstr) <413>

Описание: Используется для получения части строки.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
str	Строка	Строка	Вход	
pos	Позиция	Целый	Вход	0
n	Количество	Целый	Вход	-1

Пример:

```
using Special.FLibSYS;
test=strSubstr("Example", 0, strSize("Example")-1);
messPut("Example",1,"ReturnString: "+test);
```

4.3. Вставка одной строки в другую (strInsert) <1200>

Описание: Используется для вставки одной строки в другую.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
str	Строка	Строка	Выход	
pos	Позиция	Целый	Вход	0
ins	Вставляемая строка	Строка	Вход	

4.4. Замена части строки другой (strReplace) <531>

Описание: Используется для замены части сроки другой строкой.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
str	Строка	Строка	Выход	
pos	Позиция	Целый	Вход	0
n	Количество	Целый	Вход	-1
repl	Заменяющая строка	Строка	Вход	

4.5. Разбор строки по разделителю (strParse) <537>

Описание: Используется в разборе строки по разделителю.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
str	Строка	Строка	Вход	
lev	Уровень	Целый	Вход	
sep	Разделитель	Строка	Вход	"."
off	Смещение	Целый	Выход	

Пример:

```
using Special.FLibSYS;
ExapleString="Example:123";
test=strParse(ExapleString,1,":");
messPut("Example",0,"strParse(): "+test);
```

4.6. Разбор пути (strParsePath) <300>

Описание: Используется в разборе пути на элементы.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
path	Путь	Строка	Вход	
lev	Уровень	Целый	Вход	
off	Смещение	Целый	Выход	

Пример:

```
using Special.FLibSYS;
test=strParsePath(path,0,"/");
messPut("Example",1,"strParsePath(): "+test);
```

4.7. Путь в строку с разделителем (strPath2Sep)

Описание: Используется для преобразования пути в строку с разделителем.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	
sep	Разделитель	Строка	Вход	"."

Пример:

```
//Преобразуем значение "/ses_AGLKS/pg_so" атрибута "path" в значение "ses_AGLKS.pg_so"
```

```
using Special.FLibSYS;
test = strPath2Sep(path);
messPut("Example",0,"path: "+path);
messPut("Example",0,"strPath2Sep(): "+test);
```

4.8. Кодирование строки в HTML (strEnc2HTML)

Описание: Используется для кодирования строки для использования в исходнике HTML.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

4.9. Кодирование текста в бинарный вид (strEnc2Bin)

Описание: Используется для кодирования текста в бинарный вид, из формата <00 A0 FA DE>.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

4.10. Декодирование текста из бинарного вида (strDec4Bin)

Описание: Используется для декодирования текста из бинарного вида в формат <00 A0 FA DE>.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

4.11. Преобразование вещественного в строку (real2str)

Описание: Используется для преобразования вещественного в строку.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
val	Значение	Вещественное	Вход	
prc	Точность	Целое	Вход	4
tp	Тип	Строка	Вход	“f”

4.12. Преобразование целого в строку (int2str)

Описание: Используется для преобразования целого в строку.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
val	Значение	Целое	Вход	
base	База, поддерживаются: 8, 10, 16	Целое	Вход	10

4.13. Преобразование строки в вещественное (str2real)

Описание: Используется для преобразования строки в вещественное.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
val	Значение	Строка	Вход	

4.14. Преобразование строки в целое (str2int)

Описание: Используется для преобразования строки в целое.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Целое	Возврат	
val	Значение	Строка	Вход	
base	Основа	Целый	Вход	0

5. Функции работы с вещественным

5.1. Разделение float на слова (floatSplitWord) <56>

Описание: Разделение float (4 байтов) на слова (2 байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Значение	Вещественное	Вход	
w1	Слово 1	Целый	Выход	
w2	Слово 2	Целый	Выход	

5.2. Объединение float из слов (floatMergeWord) <70>

Описание: Объединение float (4 байтов) из слов (2 байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	

Модуль подсистемы “Специальные” <SystemTests>

Модуль:	SystemTests
Имя:	Тесты системы OpenSCADA.
Тип:	Специальные
Источник:	spec_SystemTests.so
Версия:	1.5.0
Автор:	Роман Савоченко
Описание:	Предоставляет группу тестов системы OpenSCADA.
Лицензия:	GPL

Специальный модуль SystemTests содержит набор тестов, предназначенных для тестирования различных подсистем и узлов системы OpenSCADA. Тесты выполнены в виде функций пользовательского API. Следовательно тесты можно запускать как единоразово, во вкладке "Исполнить" страницы объекта функции, так и из пользовательских процедур, передавая в них нужные аргументы.

Кроме механизмов обычного исполнения функций пользовательского API предусмотрен автономный механизм. Этот механизм представлен отдельной задачей, исполняющейся с периодом в одну секунду, в которой осуществляется вызов функций тестов в соответствии с настройками в конфигурационном файле.

Конфигурационные поля тестов помещаются в секцию модуля SystemTests подсистемы «Специальные». Формат конфигурационных полей: **<prm id="Test Id" on="1" per="10" />**

Где:

- id - идентификатор теста;
- on - признак “Тест включен”;
- per - период повторения теста (секунд).

Кроме основных атрибутов осуществляется отражение входных параметров функций тестов на одноимённые атрибуты тега "prm". Например, атрибут "name" функции "Param" можно указать в теге "prm".

Допускается указание множества тегов "prm" для одного или разных тестов с одинаковыми или различными параметрами, указывая тем самым на отдельный запуск теста с указанными параметрами. Приведём пример описания всех доступных тестов:

```
<?xml version="1.0" encoding="UTF-8" ?>
<OpenSCADA>
 <station id="DemoStation">
  <node id="sub_Special">
   <node id="mod_SystemTests">
    <prm id="Param" on="0" per="5" name="LogicLev.experiment.F3"/>
    <prm id="XML" on="0" per="10" file="/etc/oscada.xml"/>
    <prm id="Mess" on="0" per="10" categ="" arhtor="DBArch.test3" depth="10"/>
  <prm id="SOAttach" on="0" per="20" name="../../lib/openscada/daq_LogicLev.so"
mode="0" full="1"/>
    <prm id="Val" on="0" per="1" name="LogicLev.experiment.F3.var" arch_len="5"
arch_per="1000000"/>
    <prm id="Val" on="0" per="1" name="System.AutoDA.CPULoad.load" arch_len="10"
arch_per="1000000"/>
    <prm id="DB" on="0" per="10" type="MySQL"
addr="server.diya.org;roman;123456;oscadaTest" table="test" size="1000"/>
    <prm id="DB" on="0" per="10" type="DBF" addr="./DATA/DBF" table="test.dbf"
size="1000"/>
```

```
    <prm id="DB" on="0" per="10" type="SQLite" addr="./DATA/test.db" table="test"
size="1000"/>
    <prm id="DB" on="0" per="10" type="FireBird"
addr="server.diya.org:/var/tmp/test.fdb;roman;123456" table="test" size="1000"/>
    <prm id="TrOut" on="0" per="1" addr="TCP:127.0.0.1:10001" type="Sockets"
req="time"/>
    <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:10001" type="Sockets"
req="time"/>
    <prm id="TrOut" on="0" per="1" addr="UNIX:./oscada" type="Sockets" req="time"/>
    <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:daytime" type="Sockets"
req="time"/>
    <prm id="SysContrLang" on="0" per="10" path="/Archive/FSArch/mess_StatErrors/
%2fprm%2fst"/>
    <prm id="ValBuf" on="0" per="5"/>
    <prm id="Archive" on="0" per="30" arch="test1" period="1000000"/>
    <prm id="Base64Code" on="0" per="10"/>
   </node>
  </node>
 </station>
</OpenSCADA>
```

1. Параметр (Param)

Описание: Тест DAQ параметров. Вычитывает атрибуты и конфигурационные поля параметра.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
name	Адрес DAQ параметра	Строка	Вход	System.AutoDA.CPULoad

2. Разбор XML (XML)

Описание: Тест разбора файла XML. Парсит и отображает структуру указанного файла.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
file	XML файл	Строка	Вход	

3. Сообщения (Mess)

Описание: Тест архива сообщения. Периодически вычитывает новые сообщения из архива, для указанного архиватора.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
arhtor	Архиватор	Строка	Вход	FSArch.StatErrors
categ	Шаблон категории сообщения	Строка	Вход	
depth	Глубина сообщения (с)	Целый	Вход	10

4. Подключение SO (SOAttach)

Описание: Тест подключения/отключения модулей.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
name	Путь к модулю	Строка	Вход	
mode	Режим (1-подключ.;-1-отключ.;0-изменение)	Целый	Вход	0
full	Полное подключение(при старте)	Bool	Вход	1

5. Атрибут параметра (Val)

Описание: Тест значений атрибута параметра. Выполняет периодический опрос последнего значения указанного атрибута, а также опрос архива на указанную глубину.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
name	Путь к атрибуту параметра	Строка	Вход	System.AutoDA.CPULoad.load
arch_len	Глубина запроса к архиву значений (с)	Целый	Вход	10
arch_per	Период запроса к архиву значений (мкс)	Целый	Вход	1000000

6. Тест БД (DB)

Описание: Полный тест БД. Выполняет:

- создание/открытие БД;
- создание/открытие таблицы;
- создание множества записей (строк) предопределённой структуры;
- модификация множества записей;
- получение и проверка значений множества записей;
- модификация структуры записи и таблицы;
- удаление записей;
- закрытие/удаление таблицы;
- закрытие/удаление БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
type	Тип БД	Строка	Вход	SQLite
addr	Адрес БД	Строка	Вход	./DATA/test.db
table	Таблица БД	Строка	Вход	test
size	Количество записей	Целый	Вход	1000

7. Транспорт (TrOut)

Описание: Тест выходных и/или входных транспортов. Выполняет тестирование исходящего транспорта путём отправления запроса к указанному входящему транспорту.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
addr	Адрес	Строка	Вход	TCP:127.0.0.1:10001
type	Модуль транспорта	Строка	Вход	Sockets
req	Текст запроса	Строка	Вход	

8. Язык управления системой (SysContrLang)

Описание: Тест языка управления системой. Производит запрос элементов языка посредством полного пути. Полный путь к элементу языка имеет вид </Archive/%2fbd%2fm_per>. Полный путь состоит из двух вложенных путей. Первый </d_Archive/> это путь к узлу дерева контроля. Второй </bd/m_per> это путь к конкретному элементу узла.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
path	Путь к элементу языка	Строка	Вход	/Archive/BaseArh/mess_StatErrors/%2fprm%2fst

9. Буфер значений (ValBuf)

Описание: Тесты буфера значений. Содержит 13 тестов всех аспектов буфера значений (подсистема "Архивы").

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	

10. Архив значений (Archive)

Описание: Тесты размещения в архиве значений. Содержит 7(8) тестов архиватора значений на проверку корректности функционирования последовательного механизма упаковки.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
arch	Архив значений	Строка	Вход	
period	Период значений (мкс)	Целый	Вход	1000000

11. Base64 кодирование (Base64Code)

Описание: Тесты кодирования Mime Base64 алгоритмом.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	

Модуль подсистемы “Пользовательские интерфейсы” <QTStarter>

Модуль:	QTStarter
Имя:	QT GUI пускатель
Тип:	Пользовательские интерфейсы
Источник:	ui_QTStarter.so
Версия:	1.6.0
Автор:	Роман Савоченко
Описание:	Предоставляет QT GUI пускатель. QT-пускатель является единственным и обязательным компонентом для всех GUI модулей основанных на библиотеке QT.
Лицензия:	GPL

Модуль <QTStarter> предоставляет в систему OpenSCADA пускатель QT GUI модулей. Отдельный модуль для запуска QT GUI модулей понадобился по причине необходимости организации однопоточного исполнения всех компонентов и централизованной инициализации главного объекта QT-библиотеки - QApplication.

Для запуска QT GUI модулей используется расширенный интерфейс вызова функций модулей. Данный интерфейс подразумевает экспортирование функций внешними модулями. В нашем случае QT GUI модули должны экспортировать следующие функции:

- *QIcon icon();* - Передаёт объект иконки вызываемого модуля.
- *QMainWindow *openWindow();* - Создаёт объект главного окна данного QT GUI модуля и передаёт его пускателю. Может возвращать NULL в случае невозможности создания нового окна.

Для идентификации QT GUI модуль должен определять информационный элемент модуля "SubType" как "QT". Исходя из этого признака, "Стартер" с ним работает.

После получения объекта главного окна "Стартер" добавляет свою панель управления и пункт меню в это окно и запускает его. Панель управления Стартера содержит иконки для вызова всех доступных QT GUI модулей. Для исключения добавления панели управления или пункта меню модуль, содержащий окно может указать свойства "QTStarterToolDis" или "QTStarterMenuDis" соответственно.

Для указания QT GUI модулей, запускаемых при старте, модуль стартера содержит конфигурационное поле StartMod. В данном поле записываются идентификаторы запускаемых модулей через ';'. Конфигурационное поле StartMod можно описать в конфигурационном файле, а также в системной таблице БД через диалог конфигурации модуля (рис.1).

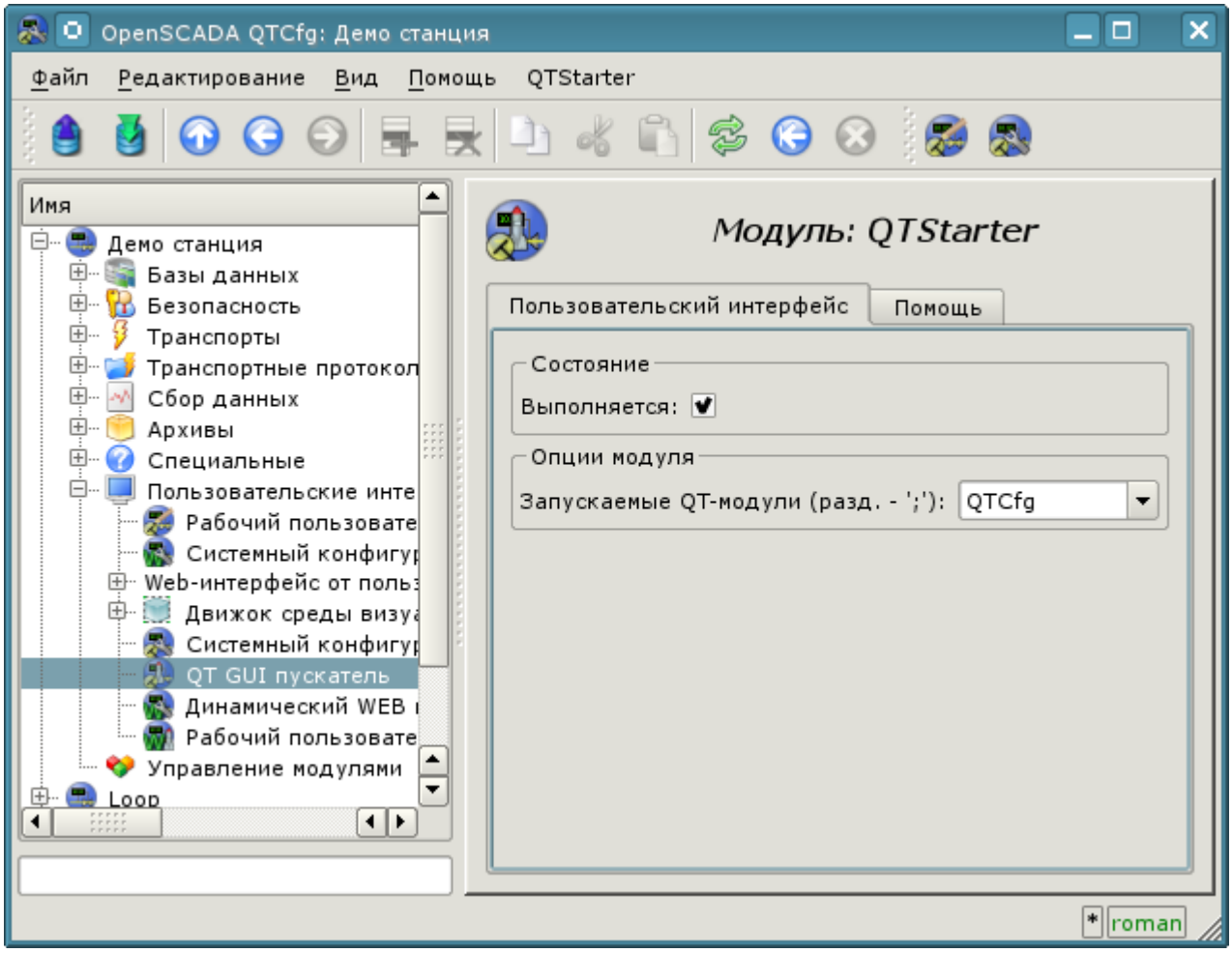


Рис.1. Страница конфигурации модуля.

В случае закрытия окон всех QT GUI модулей "Стартер" создаёт своё диалоговое окно, которое предлагает выбрать доступные QT GUI модули или завершить работу системы OpenSCADA. Вид диалогового окна приведен на рис.2.

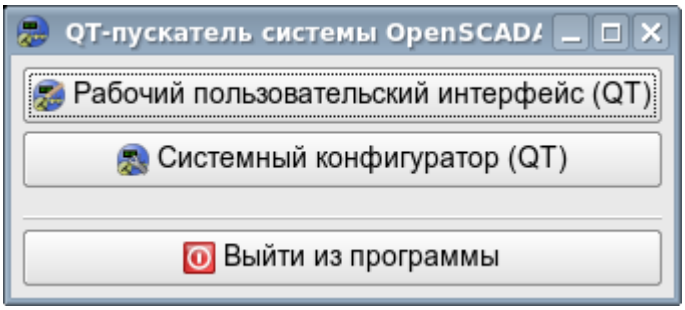


Рис.2. Диалоговое окно "Стартера".

Модуль подсистемы “Пользовательские интерфейсы” <QTCfg>

Модуль:	QTCfg
Имя:	Системный конфигуратор (QT)
Тип:	Пользовательские интерфейсы
Источник:	ui_QTCfg.so
Версия:	1.9.0
Автор:	Роман Савоченко
Описание:	Предоставляет основанный на QT конфигуратор системы OpenSCADA.
Лицензия:	GPL

Модуль <QTCfg> предоставляет конфигуратор системы OpenSCADA. Конфигуратор реализован на основе многоплатформенной библиотеки графического пользовательского интерфейса (GUI) фирмы TrollTech – QT <http://www.trolltech.com/qt/>.

В основе модуля лежит язык интерфейса управления системой OpenSCADA, а значит предоставляется единый интерфейс конфигурации. Обновление модуля может потребоваться только в случае обновления спецификации языка интерфейса управления.

Рассмотрим рабочее окно конфигуратора на рис. 1.

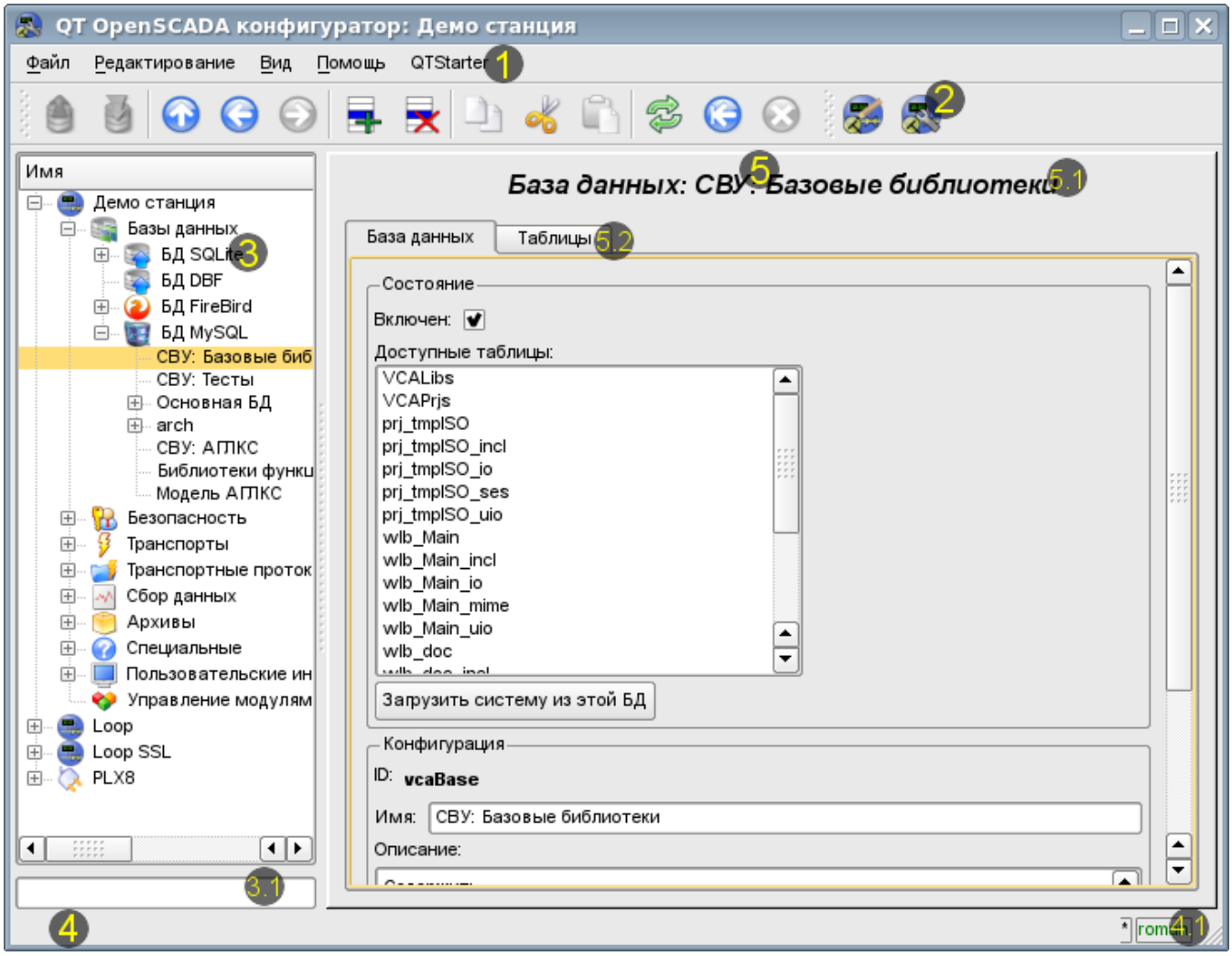


Рис.1. Рабочее окно конфигуратора

Рабочее окно конфигуратора состоит из следующих частей:

- 1 – Меню — содержит выпадающее меню конфигуратора.
- 2 – Панель инструментов — содержит кнопки быстрого управления.
- 3 – Навигатор — предназначен для прямой навигации по дереву управления.
  - 3.1 – Поле ввода текста для поиска элемента в текущей ветви дерева.
- 4 – Строка статуса — отображение состояний конфигуратора.
  - 4.1 – Индикатор/выбор пользователя — отображает текущего пользователя. По двойному клику открывается диалог выбора пользователя. А также индикатор факта внесения изменений в конфигурацию.
- 5 – Рабочее поле. Поделено на части:
  - 5.1 – Имя узла – содержит имя текущего узла.
  - 5.2 – Табулятор рабочих областей – в табулятор помещаются корневые страницы (области управления) узла. Области управления следующих уровней помещаются на информационные панели.

Меню конфигуратора содержит следующие пункты:

- *Файл* — группа общих команд:
  - *Загрузить из БД* — выполняет загрузку выбранного объекта или ветви объектов из БД.
  - *Сохранить в БД* — выполняет сохранение выбранного объекта или ветви объектов в БД.
  - *Закрыть* — закрыть окно конфигуратора.
  - *Выход* — завершение работы системы OpenSCADA.
- *Редактирование* — команды редактирования:
  - *Добавить* — добавить новый объект в контейнер.
  - *Удалить* — удалить выбранный объект.
  - *Копировать элемент* — копирование выбранного объекта.
  - *Вырезать элемент* — вырезание выбранного объекта. Исходный объект удаляется после вставки.
  - *Вставить элемент* — вставка скопированного или вырезанного элемента.
- *Вид* — команды навигации и управления видом:
  - *Вверх* — подняться вверх по дереву.
  - *Предыдущая* — открыть предыдущую открываемую страницу.
  - *Следующая* — открыть следующую открываемую страницу.
  - *Обновить* — обновить содержимое текущей страницы.
  - *Запустить* — запустить периодическое обновление содержимого текущей страницы с интервалом одна секунда.
  - *Остановить* — остановить периодическое обновление содержимого текущей страницы с интервалом одна секунда.
- *Помощь* — команды вызова помощи:
  - *Про* — информация о модуле и системе OpenSCADA.
  - *Про Qt* — информация о библиотеке Qt.
  - *Что это* — команда запроса информации о элементах интерфейса.

Панель инструментов содержит следующие кнопки быстрого управления (слева на право):

- *Загрузить из БД* — выполняет загрузку выбранного объекта или ветви объектов из БД.
- *Сохранить в БД* — выполняет сохранение выбранного объекта или ветви объектов в БД.
- *Вверх* — подняться вверх по дереву.
- *Предыдущая* — открыть предыдущую открываемую страницу.
- *Следующая* — открыть следующую открываемую страницу.
- *Добавить* — добавить новый объект в контейнер.
- *Удалить* — удалить выбранный объект.
- *Копировать элемент* — копирование выбранного объекта.
- *Вырезать элемент* — вырезание выбранного объекта. Исходный объект удаляется после вставки.
- *Вставить элемент* — вставка скопированного или вырезанного элемента.
- *Обновить* — обновить содержимое текущей страницы.

- *Запустить* — запустить периодическое обновление содержимого текущей страницы с интервалом одна секунда.
- *Остановить* — остановить периодическое обновление содержимого текущей страницы с интервалом одна секунда.
- Кнопки вызова модулей графических интерфейсов на библиотеке QT

В дереве навигации поддерживается контекстное меню следующего содержания:

- *Загрузить из БД* — выполняет загрузку выбранного объекта или ветви объектов из БД.
- *Сохранить в БД* — выполняет сохранение выбранного объекта или ветви объектов в БД.
- *Добавить* — добавить новый объект в контейнер.
- *Удалить* — удалить выбранный объект.
- *Копировать элемент* — копирование выбранного объекта.
- *Вырезать элемент* — вырезание выбранного объекта. Исходный объект удаляется после вставки.
- *Вставить элемент* — вставка скопированного или вырезанного элемента.
- *Обновить элементы дерева* — выполняет обновления содержимого дерева навигации.

Элементы управления делятся на базовые, команды, списки, таблицы и изображения. Все элементы отображаются в последовательности, строго соответствующей их расположению в описании языка интерфейса управления.

1. Конфигурация

Для настройки собственного поведения в неочевидных ситуациях модулем предоставляется возможность настройки отдельных параметров посредством интерфейса управления OpenSCADA (рис. 2). Таковыми параметрами являются:

- Начальный путь конфигуратора — позволяет определить, какую страницу открывать при запуске конфигуратора.
- Начальный пользователь конфигуратора — указывает, от имени какого пользователя открывать конфигуратор без запроса пароля.
- Ссылка на страницу конфигурации перечня внешних OpenSCADA станций, используемая для предоставления возможности удалённой конфигурации.

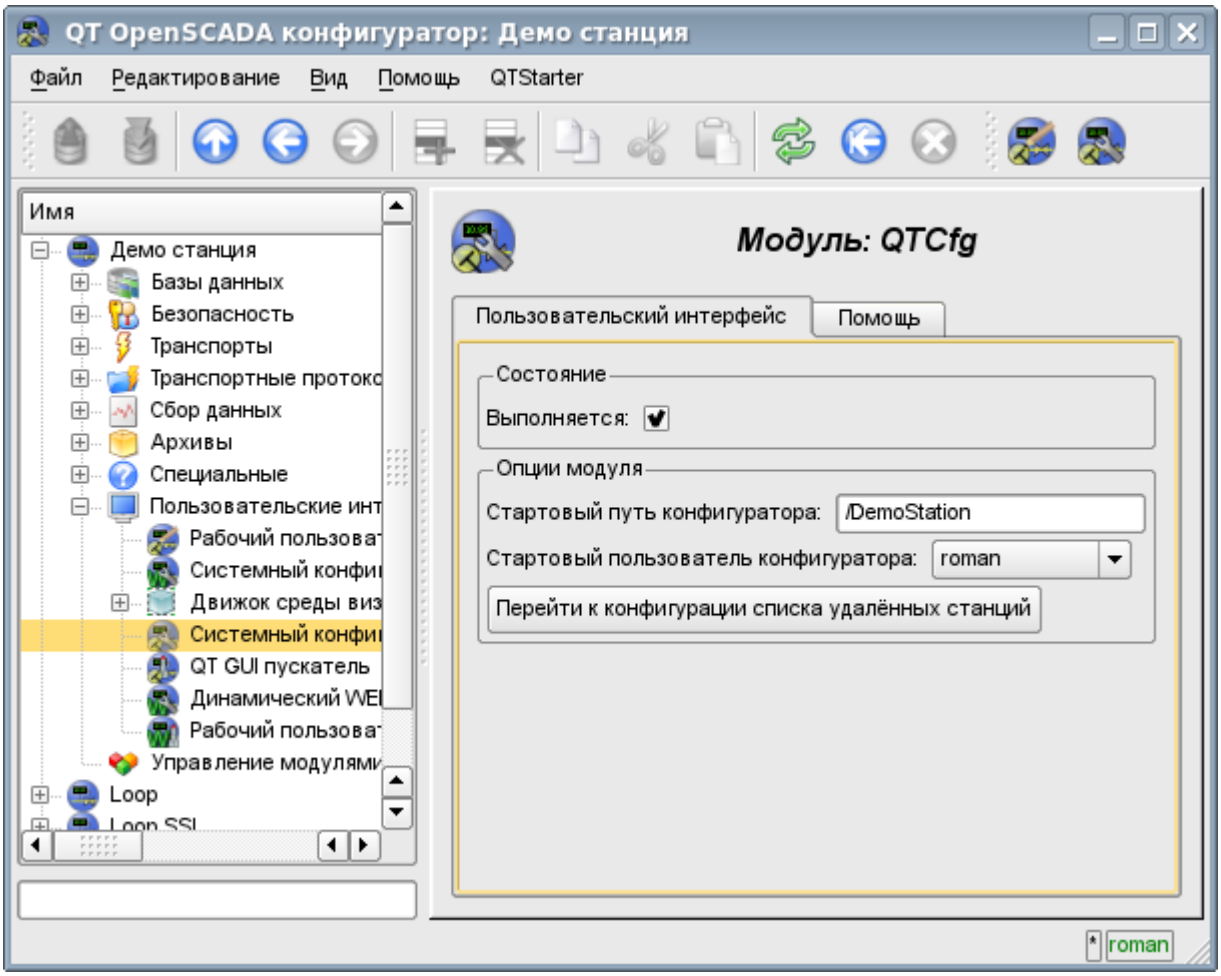


Рис.2. Страница конфигурации самого конфигуратора.

2. Базовые элементы

В число базовых элементов входят: информационные элементы, поля ввода значений, элементы выбора из списка, флаги. В случае отсутствия имени элемента базовый элемент присоединяется к предыдущему базовому элементу. Пример группы базовых элементов с присоединением приведён на рис.3.

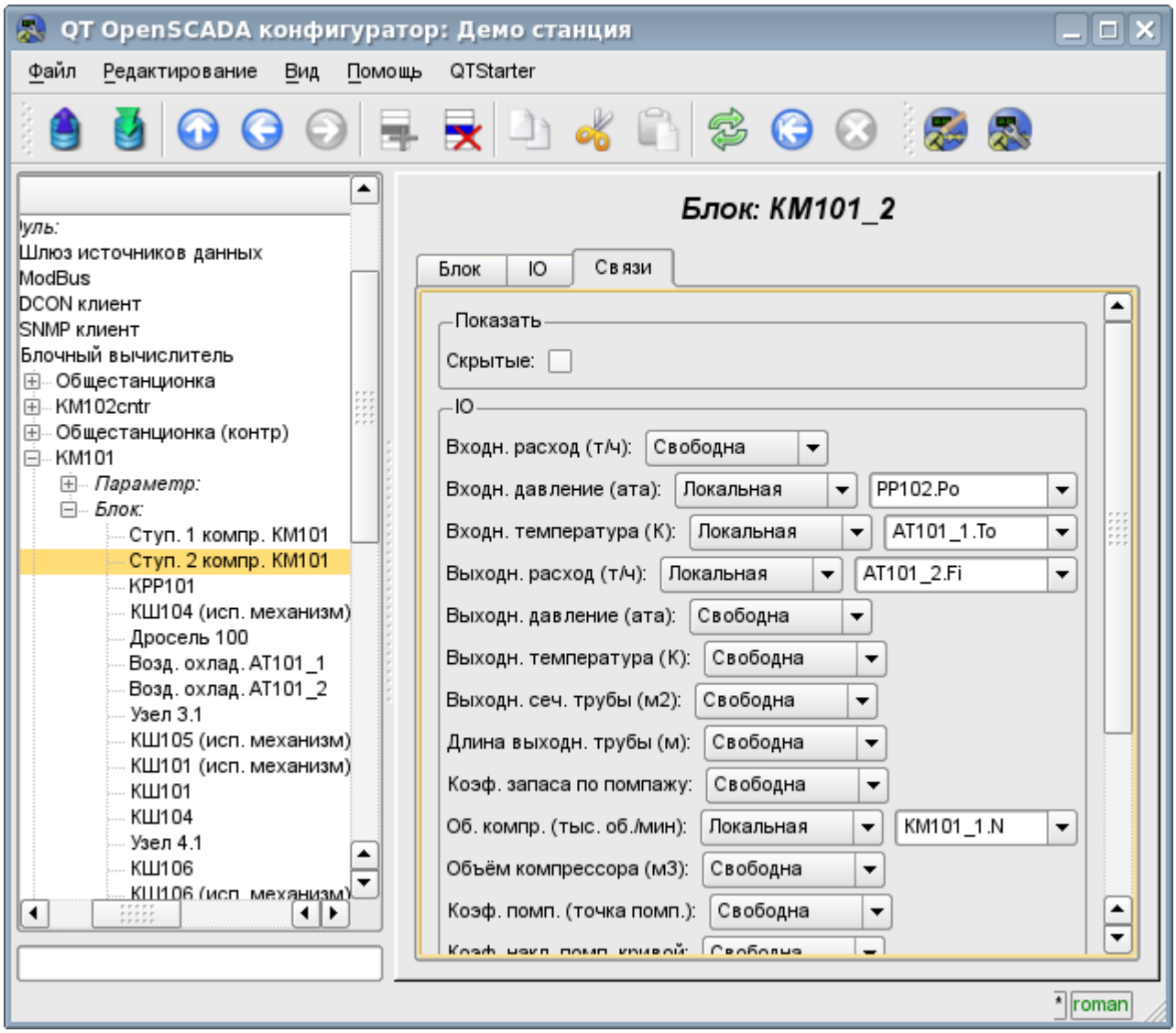


Рис.3. Присоединение базовых элементов.

3. Команды

Команды – это элементы для передачи определённых указаний действия узлу и организации ссылок на страницы. Команды могут содержать параметры. Параметры формируются из базовых элементов. Пример команды с параметрами приведен на рис.4.

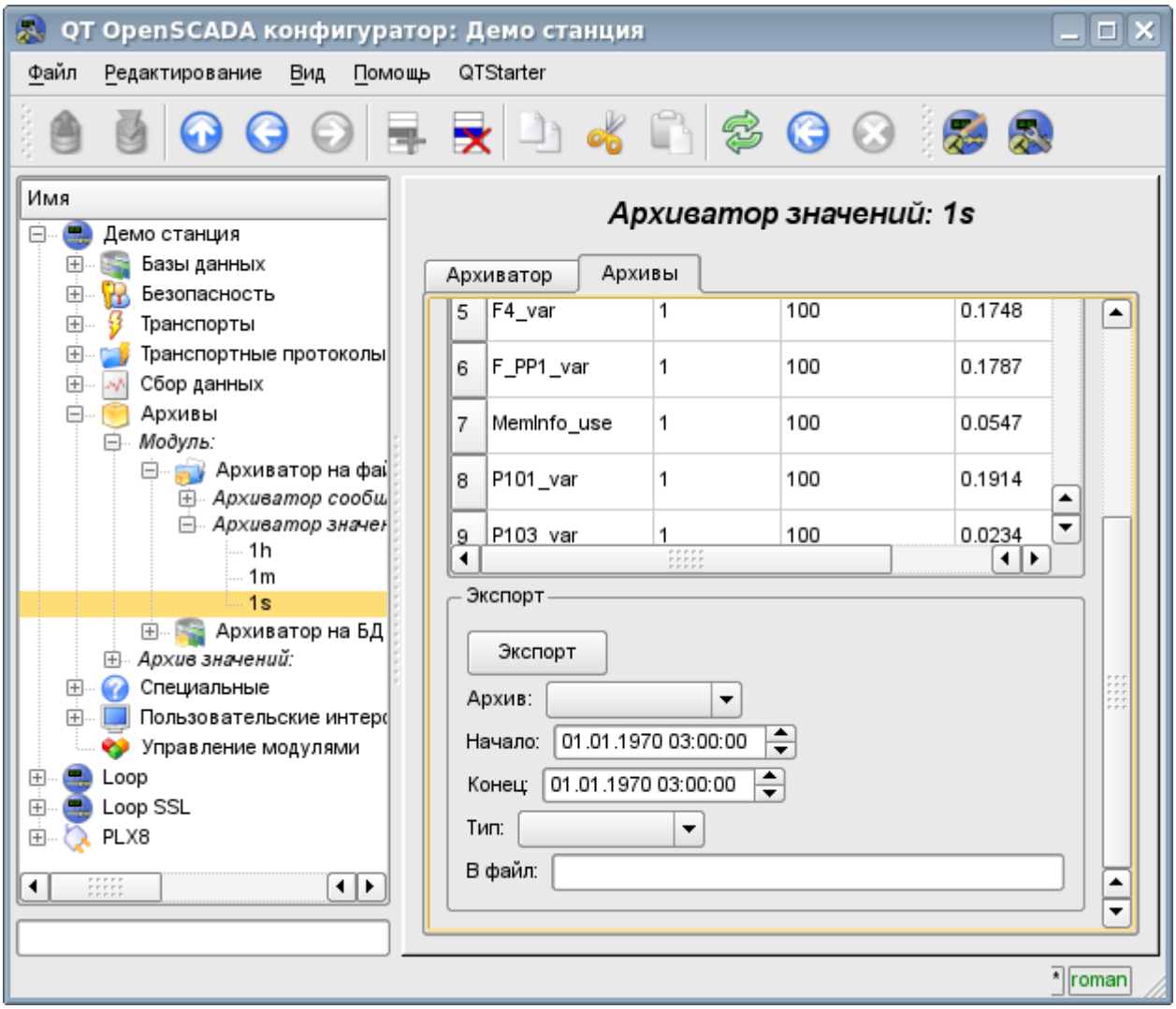


Рис.4. Команда.

4. Списки

Списки содержат группу базовых элементов одного типа. Операции над элементами доступны через контекстное меню списка. Через элементы списка могут выполняться операции перехода на другие страницы. Переход осуществляется посредством двойного клика мышки на элементе списка. Списки могут быть индексированными. Пример списка приведен на рис. 5.

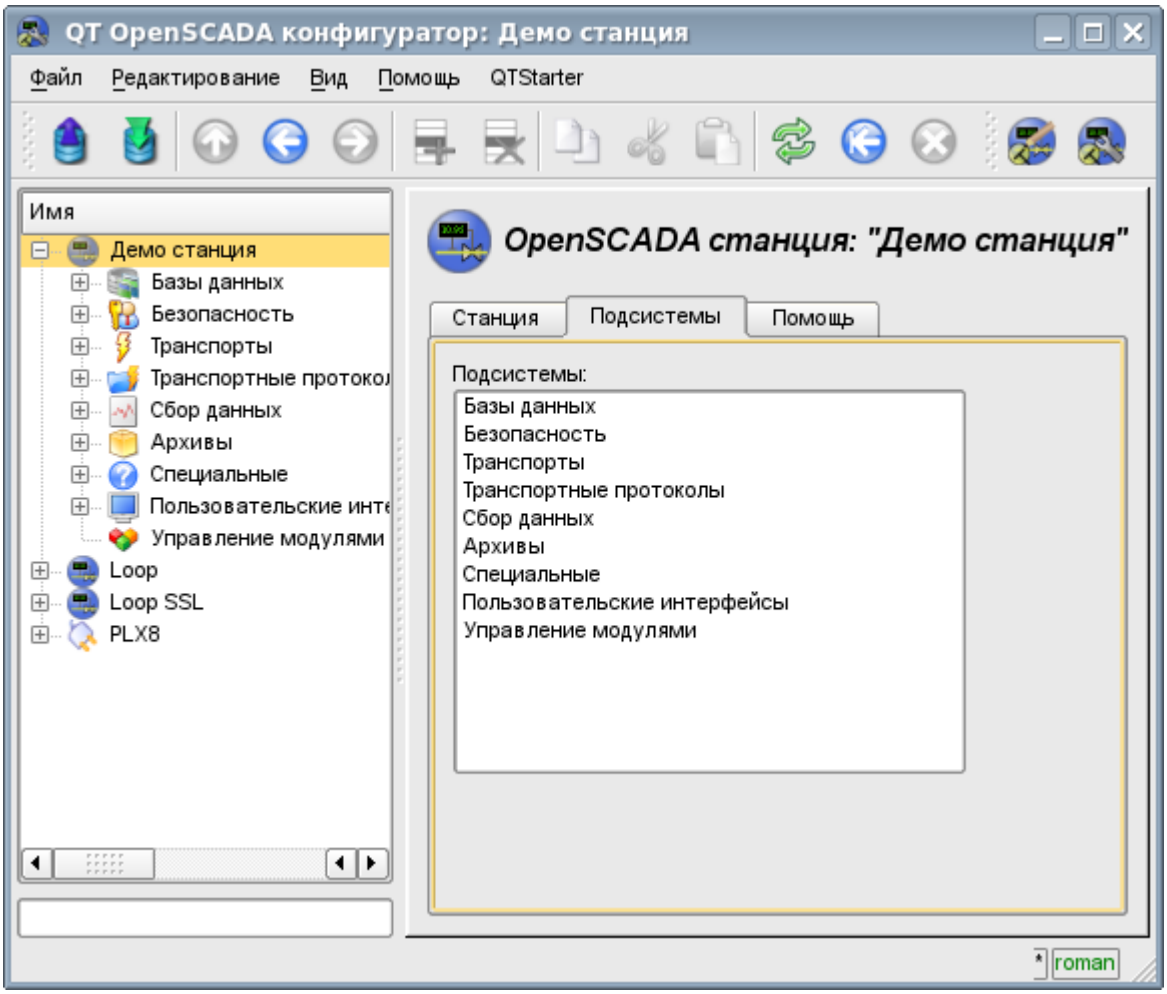


Рис.5. Список.

5. Таблицы

Таблицы содержат значения базовых элементов. Тип базового элемента является индивидуальным для каждой колонки. Пример таблицы приведен на рис. 6. Операции над структурой таблицы для редактируемых таблиц доступны посредством контекстного меню. Редактирование элементов таблицы производится путём двойного клика по нужной ячейке.

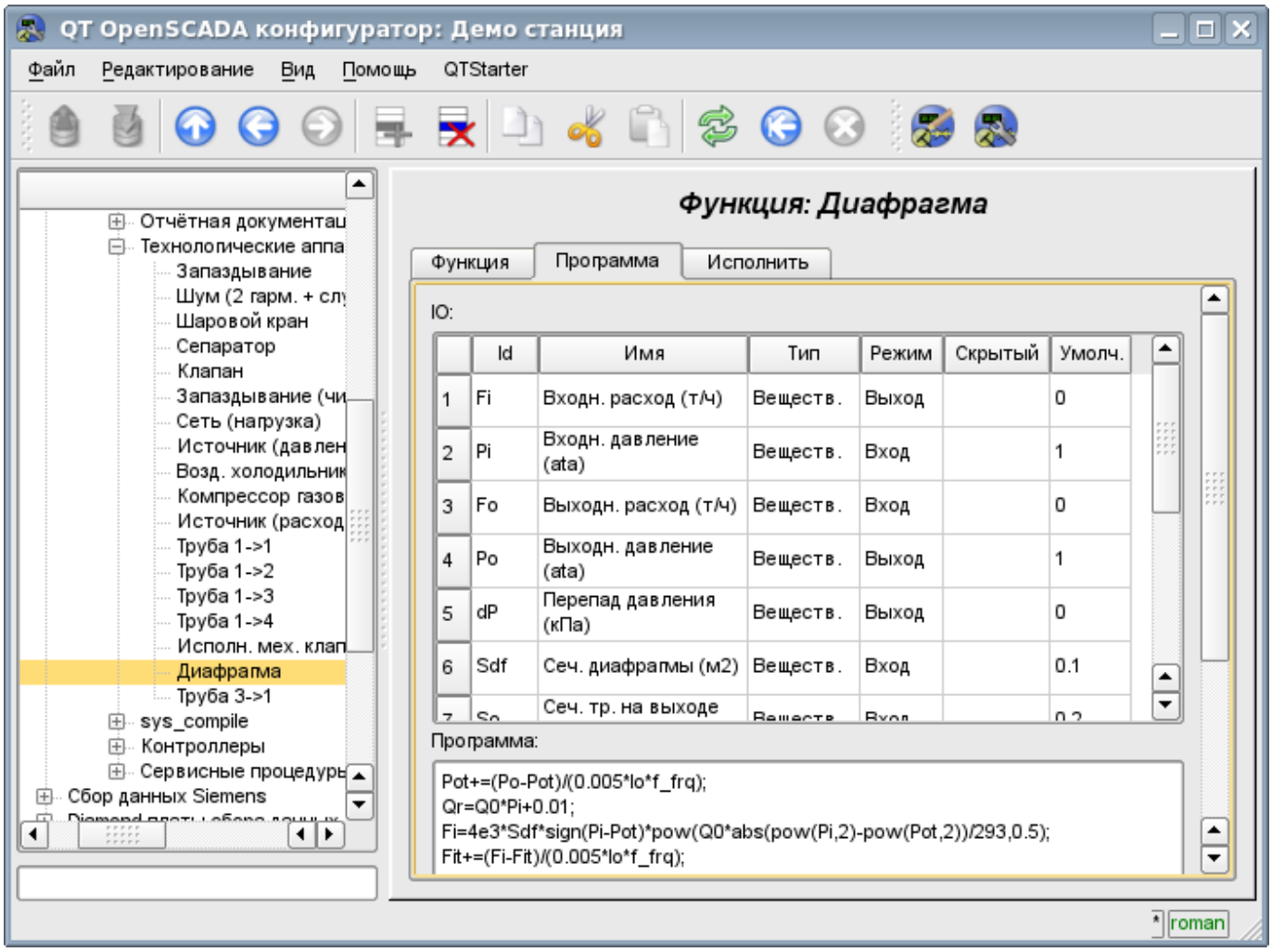


Рис.6. Таблица.

6. Изображения

Изображения призваны передавать графическую информацию в конфигураторы. Пример изображения приведен на рис. 7.

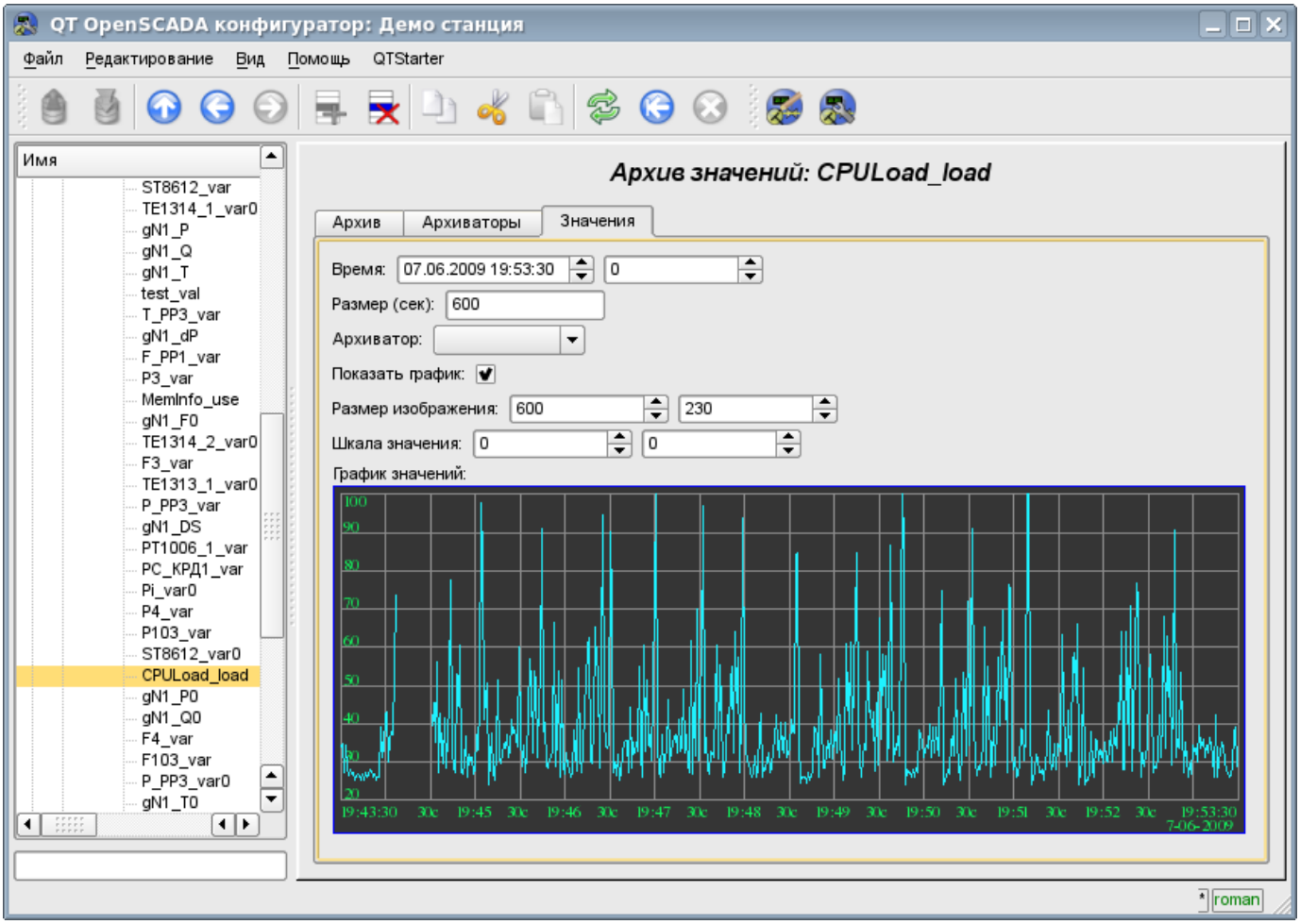


Рис.7. Изображение.

Модуль подсистемы “Пользовательские интерфейсы” <WebCfg>

Модуль:	WebCfg
Имя:	Системный конфигуратор (Web)
Тип:	Пользовательские интерфейсы
Источник:	ui_WebCfg.so
Версия:	1.5.4
Автор:	Роман Савоченко
Описание:	Предоставляет, основанный на Web, конфигуратор системы OpenSCADA.
Лицензия:	GPL

Модуль WebCfg предоставляет конфигуратор системы OpenSCADA. Конфигуратор реализован на основе Web технологий. Для работы конфигуратора достаточно обычного WEB-браузера. Работоспособность модуля <WebCfg> тестировалась в связке с модулями <Transport.Sockets> и <Protocol.HTTP> на следующих Web-браузерах:

- Mozilla;
- Firefox;
- Konqueror;
- Opera;
- IE.

Модуль построен на основе языка интерфейса управления системой OpenSCADA, а значит предоставляет единый интерфейс конфигурации. Обновление модуля может потребоваться только в случае обновления спецификации языка интерфейса управления.

Кроме принадлежности модуля системе OpenSCADA, он также принадлежит, является модулем, модуля транспортного протокола <HTTP>. Собственно, вызов WebCfg производиться из HTTP. Вызов производится посредством расширенного механизма коммуникации через экспортированные, в модуле WebCfg, функции: HttpGet и HttpSet.

Интерфейс модуля реализован на языке XHTML 1.0 Transitional с вкраплениями JavaScript.

Использование модуля начинается с открытия сеанса пользователя, аутентификации пользователя, модулем протокола HTTP (Protocol.HTTP) (рис.1). Для функционирования аутентификации и механизма сохранения сеанса браузер должен разрешать Cookies.

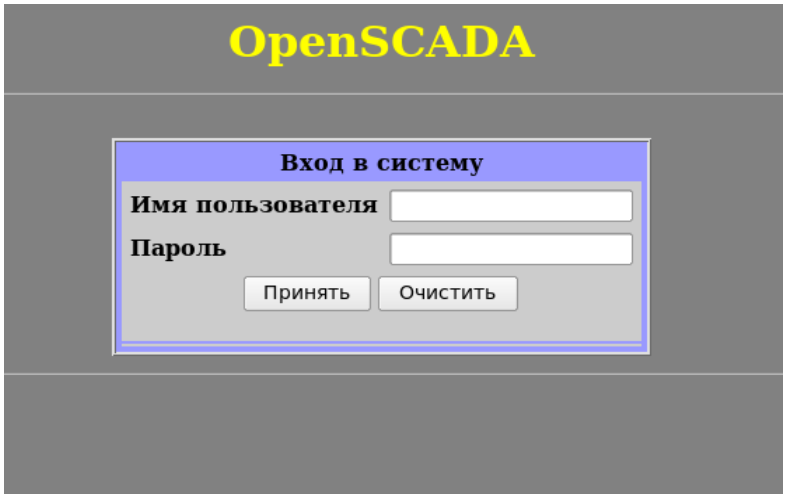


Рис.1. Аутентификация пользователя.

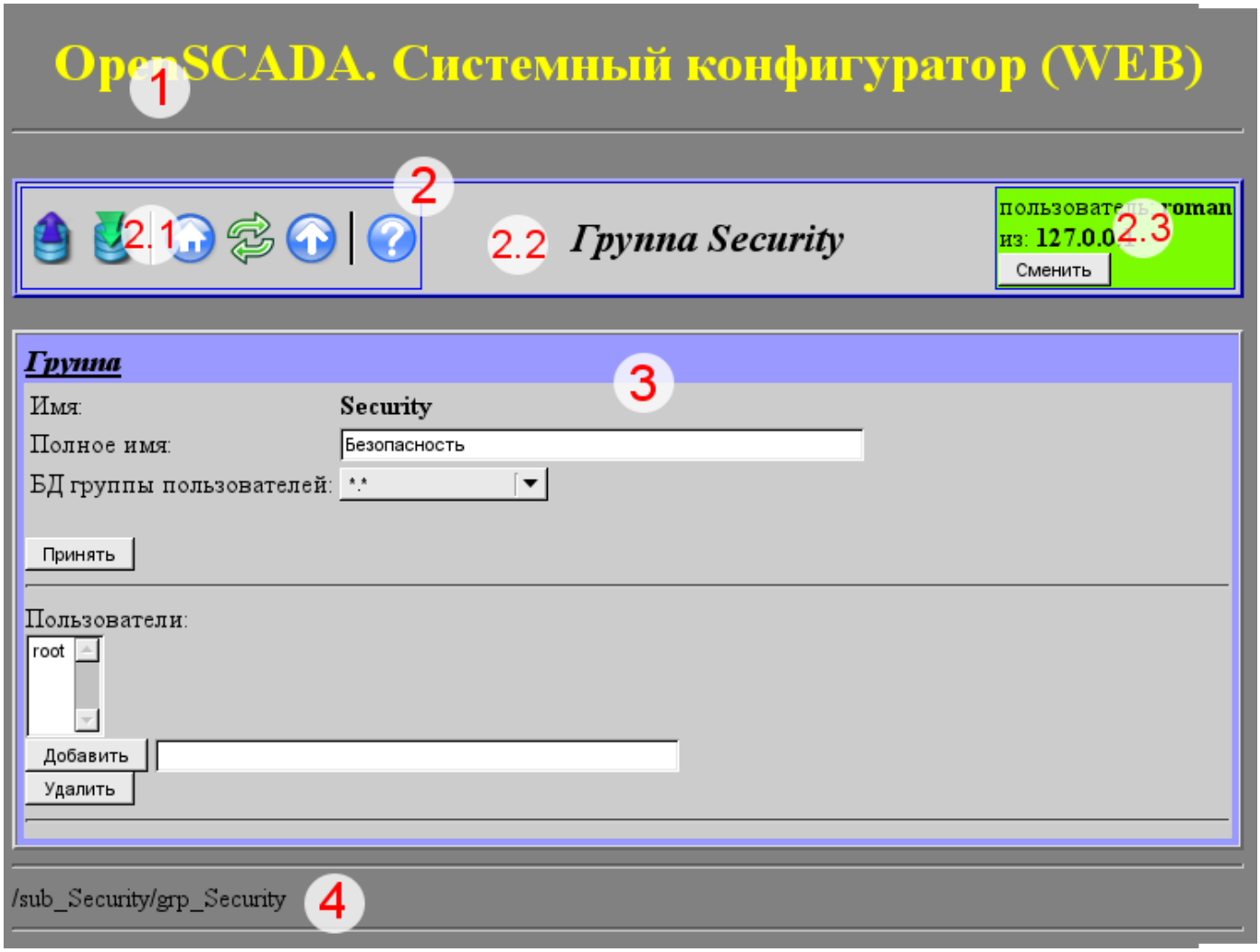


Рис.2. Структура рабочего окна пользователя.

После аутентификации пользователь попадает в рабочее окно (рис.2), которое состоит из следующих частей:

1.Верхний колонтитул – содержит наименование модуля.

2.Панель управления. Состоит из:

2.1.Навигатор – выполняет функции навигации по дереву страниц.

2.2.Наименование узла.

2.3.Пользователь системы – отображает пользователя текущего сеанса, его адрес и позволяет изменить пользователя.

3.Рабочее поле – содержит конфигурационный контент языка интерфейса управления, начиная с корневых вкладок и заканчивая конечными элементами управления.

4.Нижний колонтитул – содержит адрес текущей страницы.

Адресация страниц начинается с элемента второго уровня URL. Это связанно с тем, что элемент первого уровня используется для идентификации модуля пользовательского Web-интерфейса. Например URL: <http://localhost.localdomain:10002/WebCfg//Functions> можно расшифровать как вызов страницы первого уровня “Functions” Web модуля <WebCfg> на хосте localhost.localdomain через порт 10002.

Элементы управления делятся на: базовые, команды, списки, таблицы и изображения. Все четыре типа отображаются отдельными блоками, в не зависимости от их расположения в описании.

1. Базовые элементы

В число базовых элементов входят: информационные элементы, поля ввода значений, элементы выбора из списка, флаги. Для установки новых значений базовых элементов используется групповой метод, для этого на форме существует кнопка “Принять”. В случае отсутствия имени элемента, базовый элемент стыкуется к предыдущему базовому элементу. Пример группы базовых элементов, со стыковкой, приведён на рис.3.

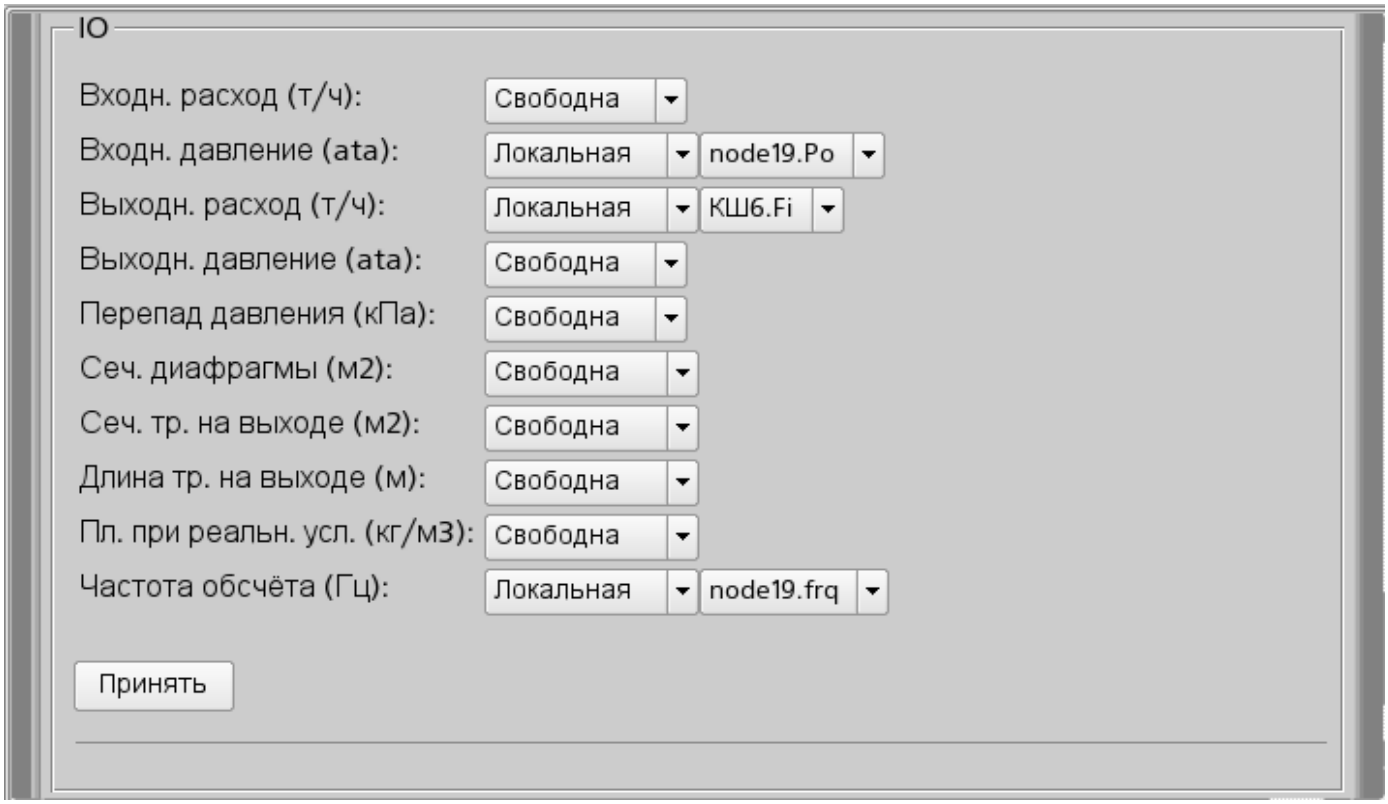


Рис.3. Базовые элементы и их стыковка.

2. Команды

Команды это элементы для передачи определённых действий узлу и организации ссылок на страницы. Команды могут содержать параметры. Параметры формируются из базовых элементов. Пример команды с параметрами приведен на рис.4.

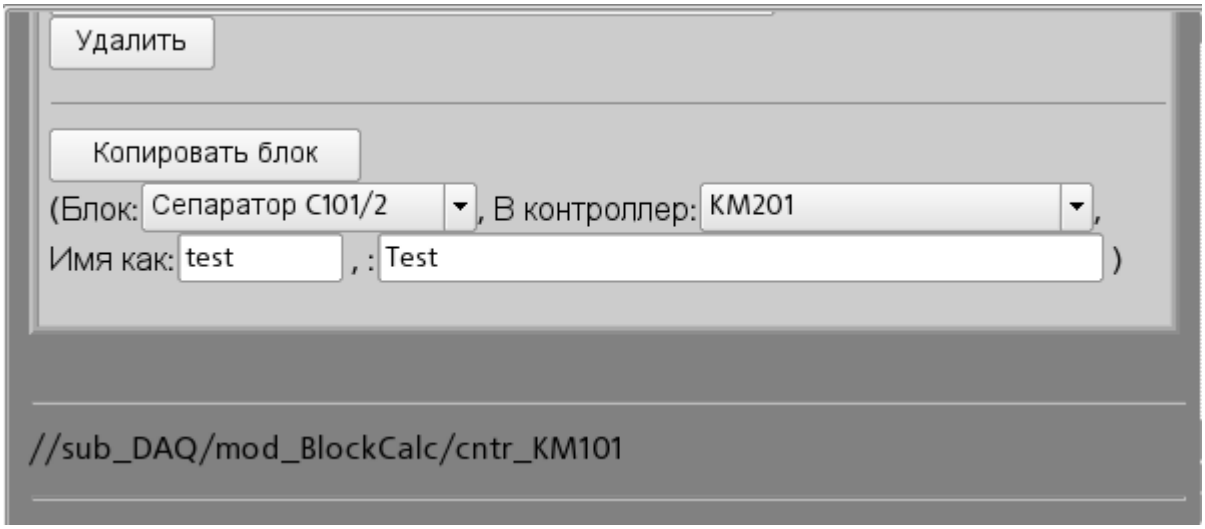


Рис.4. Команда.

3. Списки

Списки содержат группу базовых элементов одного типа. Для операций над элементами списка добавляются дополнительные кнопки. Кроме того, через элементы списка выполняются операции перехода на другие страницы. Для перехода добавляется кнопка “Перейти”. Списки могут быть индексированными. Пример списка с переходом приведен на рис.5.

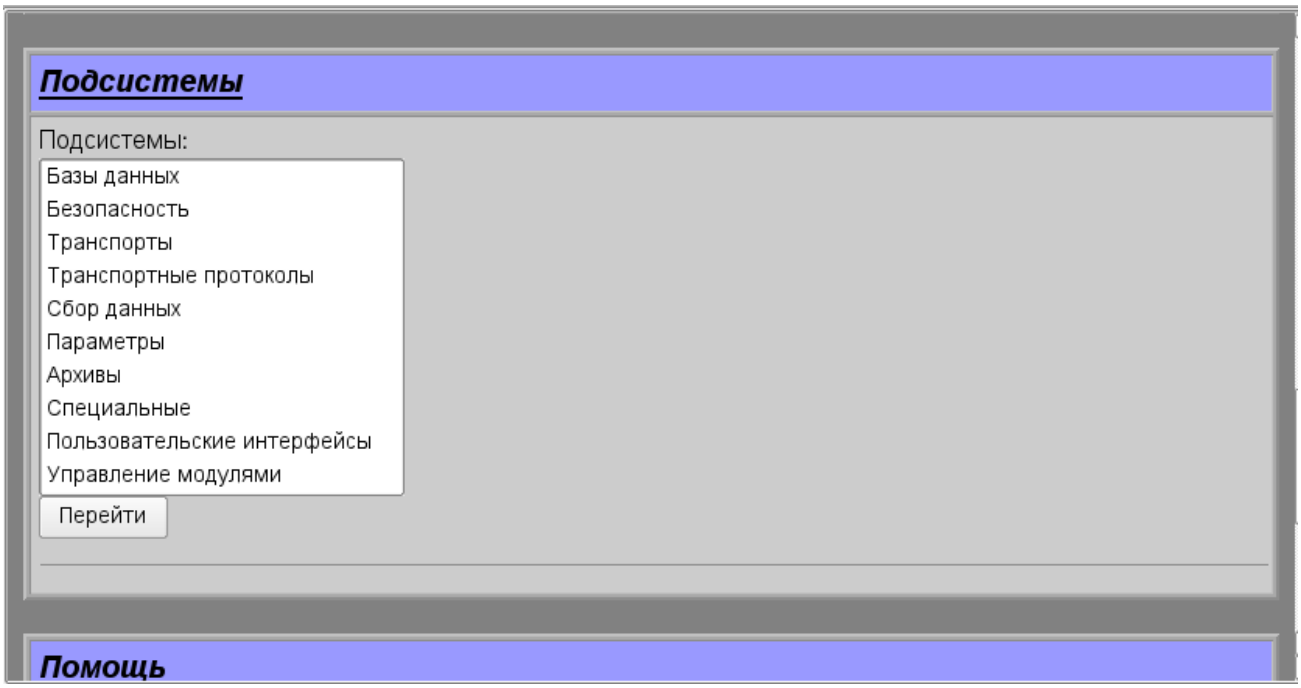


Рис.5. Список.

4. Таблицы

Таблицы содержат значения базовых элементов. Тип базового элемента определяется отдельно для каждой колонки. Пример таблицы приведен. на рис.6.

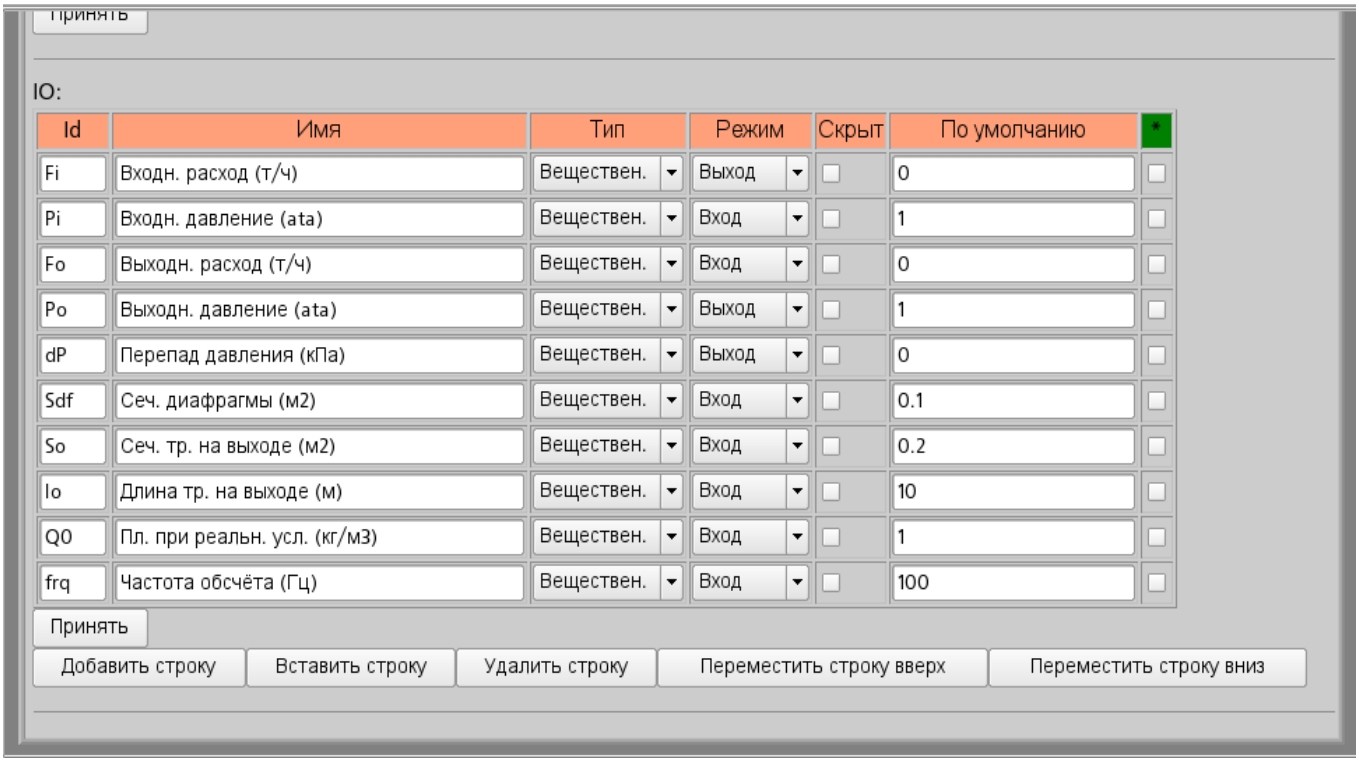


Рис.6. Таблица.

5. Изображения

Изображения призваны передавать графическую информацию в конфигураторы. Пример изображения приведен на рис. 7.

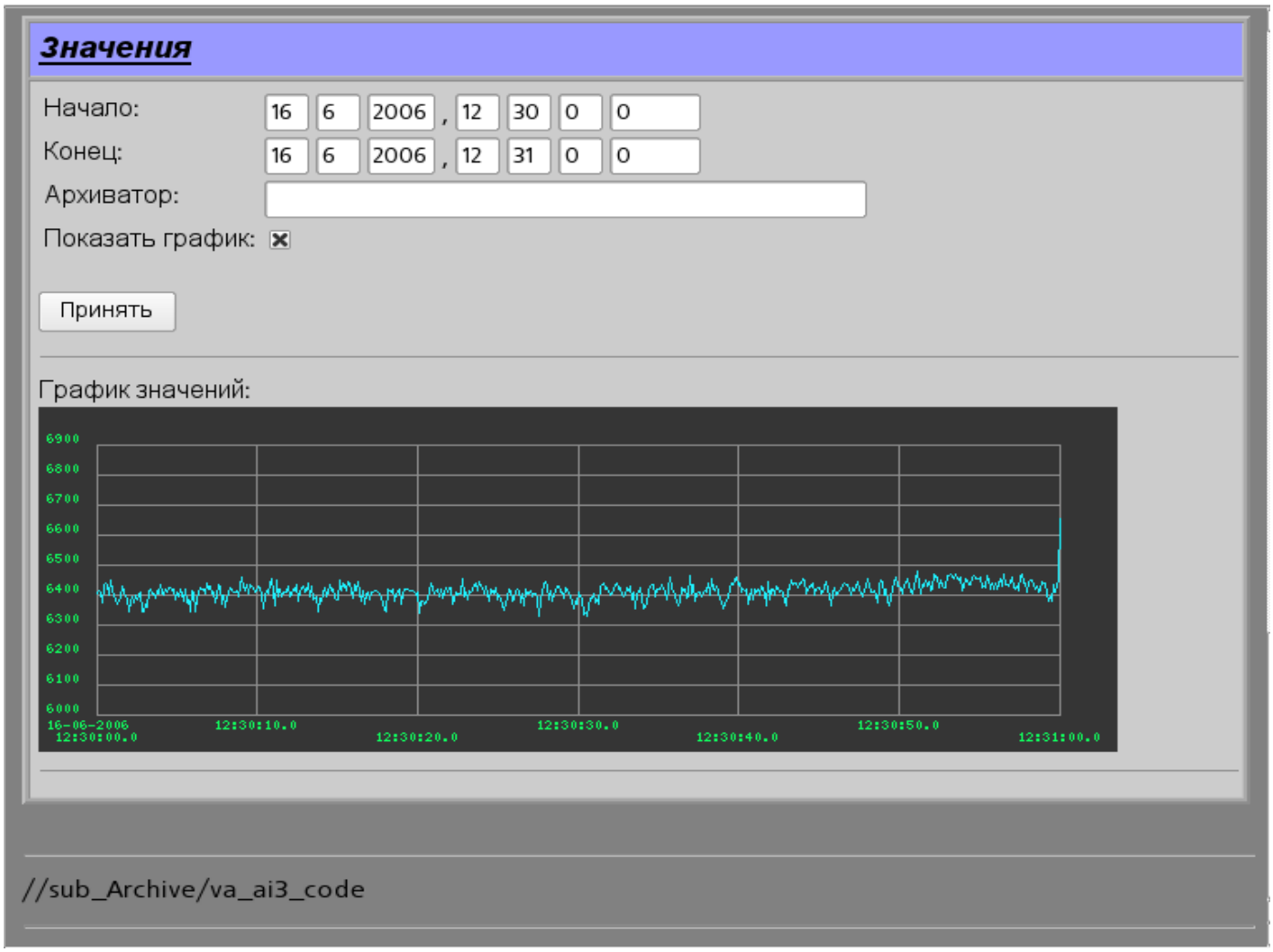


Рис.7. Изображение.

Модуль подсистемы “Пользовательские интерфейсы” <WebCfgD>

Модуль:	WebCfgD
Имя:	Динамический WEB конфигуратор
Тип:	Пользовательские интерфейсы
Источник:	ui_WebCfgD.so
Версия:	0.6.5
Автор:	Роман Савоченко
Описание:	Предоставляет динамический WEB основанный конфигуратор. Использует XHTML, CSS и JavaScript технологии.
Лицензия:	GPL

Модуль <WebCfgD> предоставляет конфигуратор системы OpenSCADA. Конфигуратор реализован на основе таких Web-технологий:

- *HTTP* — протокол передачи гипертекста;
- *XHTML* — расширенный язык разметки гипертекстовых документов;
- *CSS* — каскадные таблицы стилей гипертекстовых документов;
- *JavaScript* — встроенный в гипертекстовый документ язык программирования браузера;
- *DOM* — объектная модель документа внутренней структуры браузера;
- *AJAX* — механизм асинхронных и синхронных запросов из JavaScript к серверу;
- *XML* — расширяемый язык разметки.

Интерфейс конфигуратора формируется в WEB-браузере путём обращения к WEB-серверу и получения от него XHTML-документа по протоколу HTTP. В данном случае в роли WEB-сервера выступает система OpenSCADA, которая поддерживает стандартные коммуникационные механизмы TCP-сетей (модуль Transport.Sockets), протокол передачи гипертекста (модуль Protocol.HTTP), а также шифрование трафика между браузером и сервером (Transport.SSL). Исходя из этого, для получения доступа к интерфейсу конфигурирования OpenSCADA, предоставляемого этим модулем, необходимо в OpenSCADA настроить транспорт (Transport.Sockets или Transport.SSL) в связке с протоколом HTTP (Protocol.HTTP). В поставке с системой OpenSCADA идут конфигурационные файлы содержащие настройки Transport.Sockets для портов 10002 и 10004. Следовательно интерфейс модуля в конфигурации OpenSCADA по умолчанию будет доступен по URL: http://localhost:10002 или http://localhost:10004.

После получения XHTML-документа запускается программа на JavaScript для формирования динамического интерфейса конфигуратора.

В основе модуля лежит язык интерфейса управления системой OpenSCADA, а значит предоставляется единый интерфейс конфигурации. Обновление модуля может потребоваться только в случае обновления спецификации языка интерфейса управления.

Модуль реализовался и тестировался на трёх WEB-браузерах, представителях трёх типов WEB-движков, а именно:

- Mozilla Firefox 3.0.4
- Opera 9.6.2
- Konqueror 3.5.10

Использование модуля начинается с открытия сеанса пользователя, аутентификации, модулем протокола HTTP (Protocol.HTTP). Для функционирования аутентификации и механизма сохранения сеанса браузер должен разрешать Cookies.

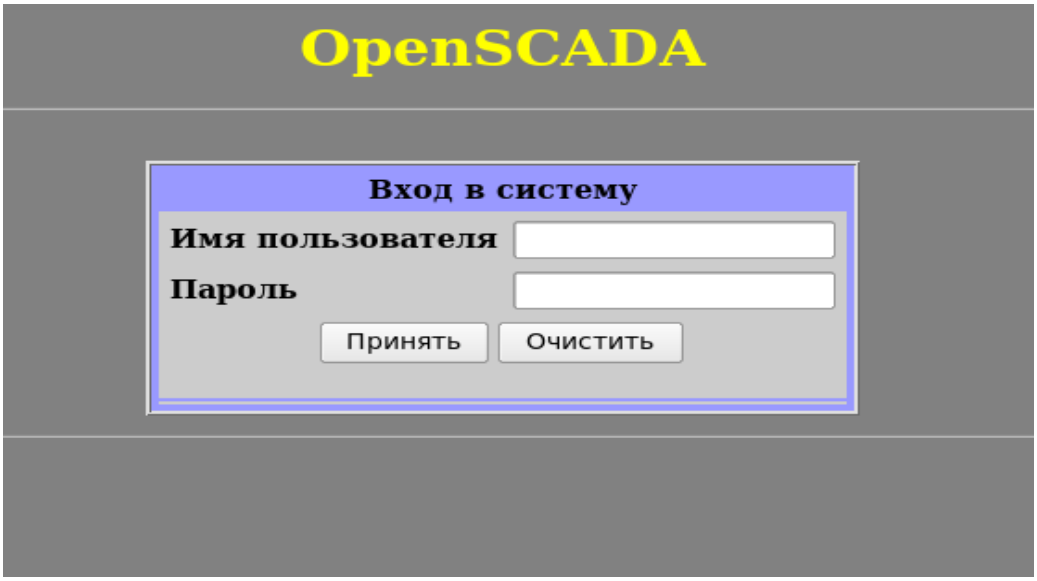


Рис.1. Аутентификация пользователя.

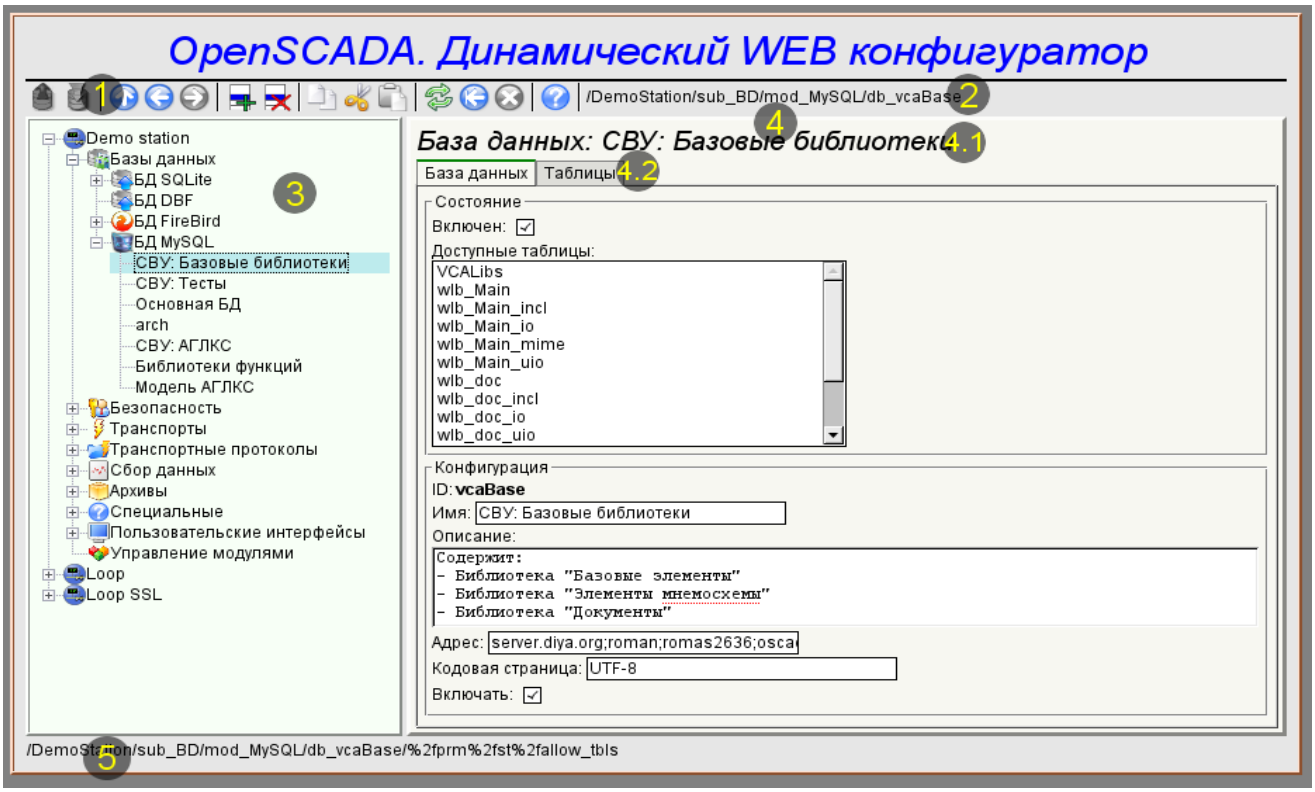


Рис.2. Рабочее окно конфигуратора

Рассмотрим рабочее окно конфигуратора на рис. 2.

Рабочее окно конфигуратора состоит из следующих частей:

- 1 – Панель инструментов — содержит кнопки управления.
- 2 – Адрес открытого узла — отображает текущий выбранный узел.
- 3 – Навигатор — предназначен для прямой навигации по дереву управления.
- 4 – Рабочее поле. Поделено на части:
  - 4.1 – Имя узла – содержит имя текущего узла.
  - 4.2 – Табулятор рабочих областей – в табулятор помещаются корневые страницы (области управления) узла. Области управления следующих уровней помещаются на информационные панели.
- 5 – Строка статуса — отображение состояний конфигуратора.

Панель инструментов содержит следующие кнопки управления (слева на право):

- *Загрузить* — выполняет загрузку выбранного объекта или ветви объектов из БД.

- *Сохранить* — выполняет сохранение выбранного объекта или ветви объектов в БД.
- *На уровень выше* — пдняться вверх по дереву.
- *Предыдущий* — открыть предыдущую открываемую страницу.
- *Следующий* — открыть следующую открываемую страницу.
- *Добавить элемент* — добавить новый объект в контейнер.
- *Удалить элемент* — удалить выбранный объект.
- *Копировать элемент* — копирование выбранного объекта.
- *Вырезать элемент* — вырезание выбранного объекта. Исходный объект удаляется после вставки.
- *Вставить элемент* — вставка скопированного или вырезанного элемента.
- *Перегрузить элемент* — обновить содержимое текущей страницы.
- *Запуск периодического обновления* — запустить периодическое обновление содержимого текущей страницы с интервалом пять секунд.
- *Останов периодического обновления* — остановить периодическое обновление содержимого текущей страницы с интервалом пять секунда.
- *Про...* — информация о данном модуле.

Элементы управления делятся на базовые, команды, списки, таблицы и изображения. Все элементы отображаются в последовательности, строго соответствующей их расположению в описании языка интерфейса управления.

1. Конфигурация

Для настройки собственного поведения в неочевидных ситуациях модулем предоставляется возможность настройки отдельных параметров посредством интерфейса управления OpenSCADA (рис. 3). Таковыми параметрами являются:

- Время жизни сеанса аутентификации(мин) — указывает в течении какого интервала времени бездействия пользователя его сеанс будет сохраняться.
- Ссылка на страницу конфигурации перечня внешних OpenSCADA станций, используемый для предоставления возможности удалённой конфигурации.

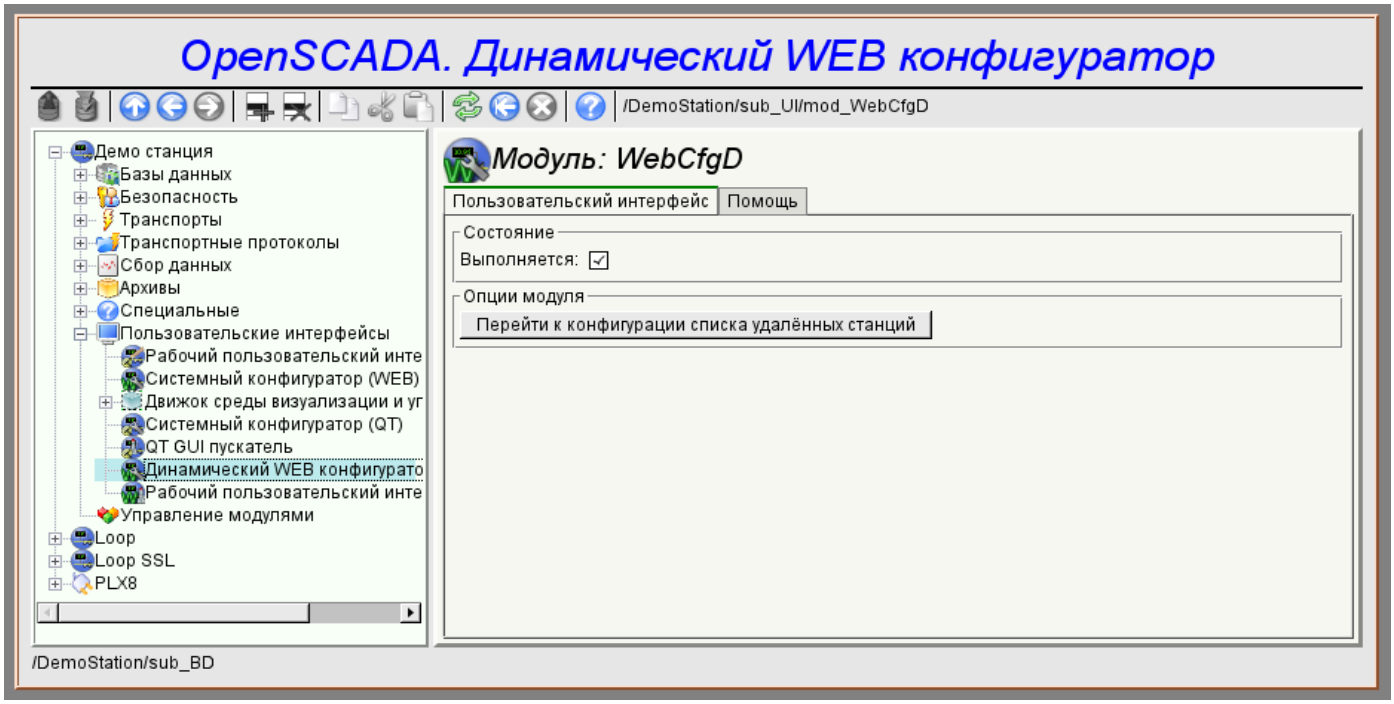


Рис.3. Страница конфигурации самого конфигуратора.

2. Базовые элементы

В число базовых элементов входят: информационные элементы, поля ввода значений, элементы выбора из списка, флаги. В случае отсутствия имени элемента базовый элемент присоединяется к предыдущему базовому элементу. Пример группы базовых элементов с присоединением приведён на рис.4.

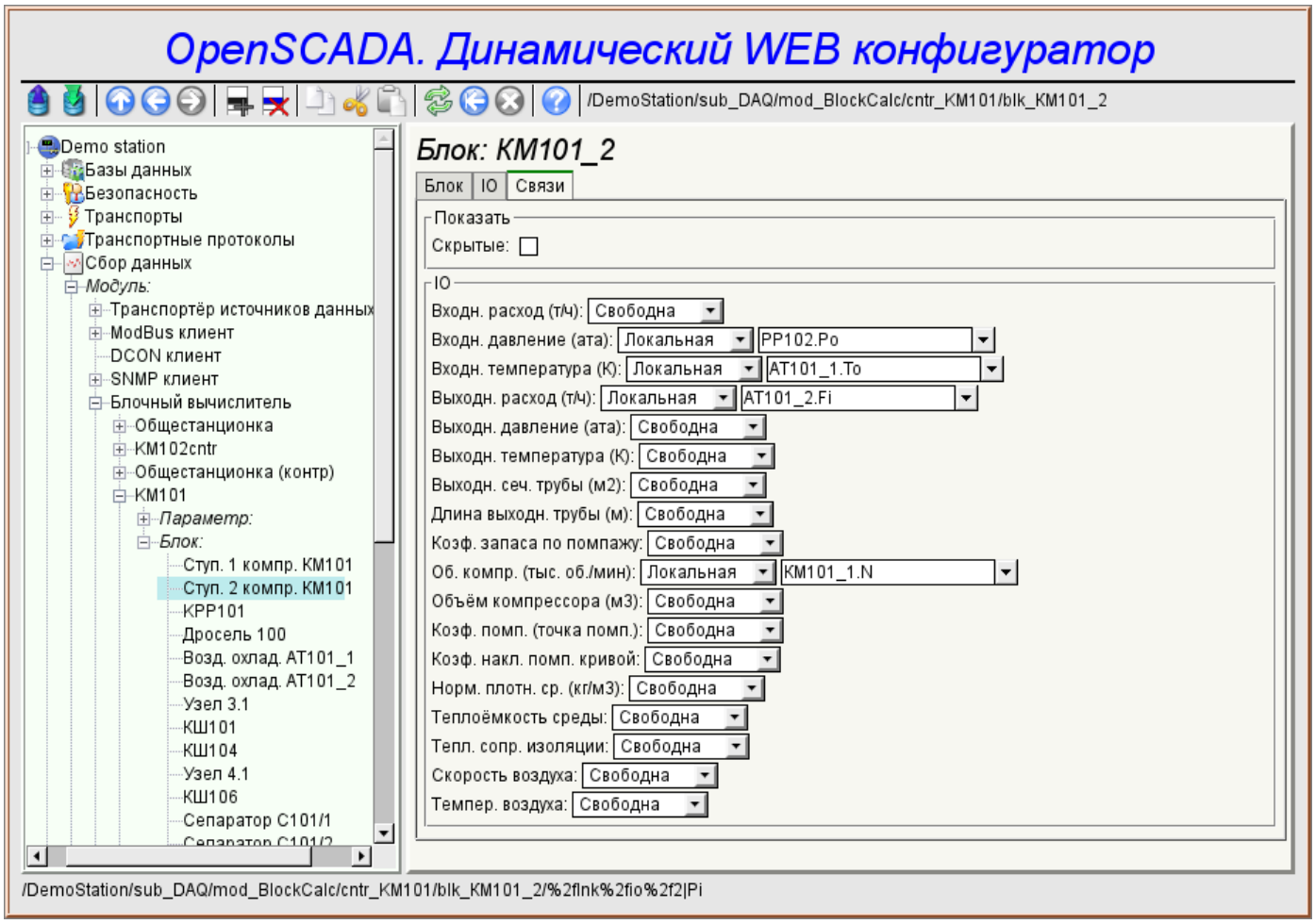


Рис.4. Присоединение базовых элементов.

3. Команды

Команды - это элементы для передачи определённых действий узлу и организации ссылок на страницы. Команды могут содержать параметры. Параметры формируются из базовых элементов. Пример команды с параметрами приведен на рис.5.

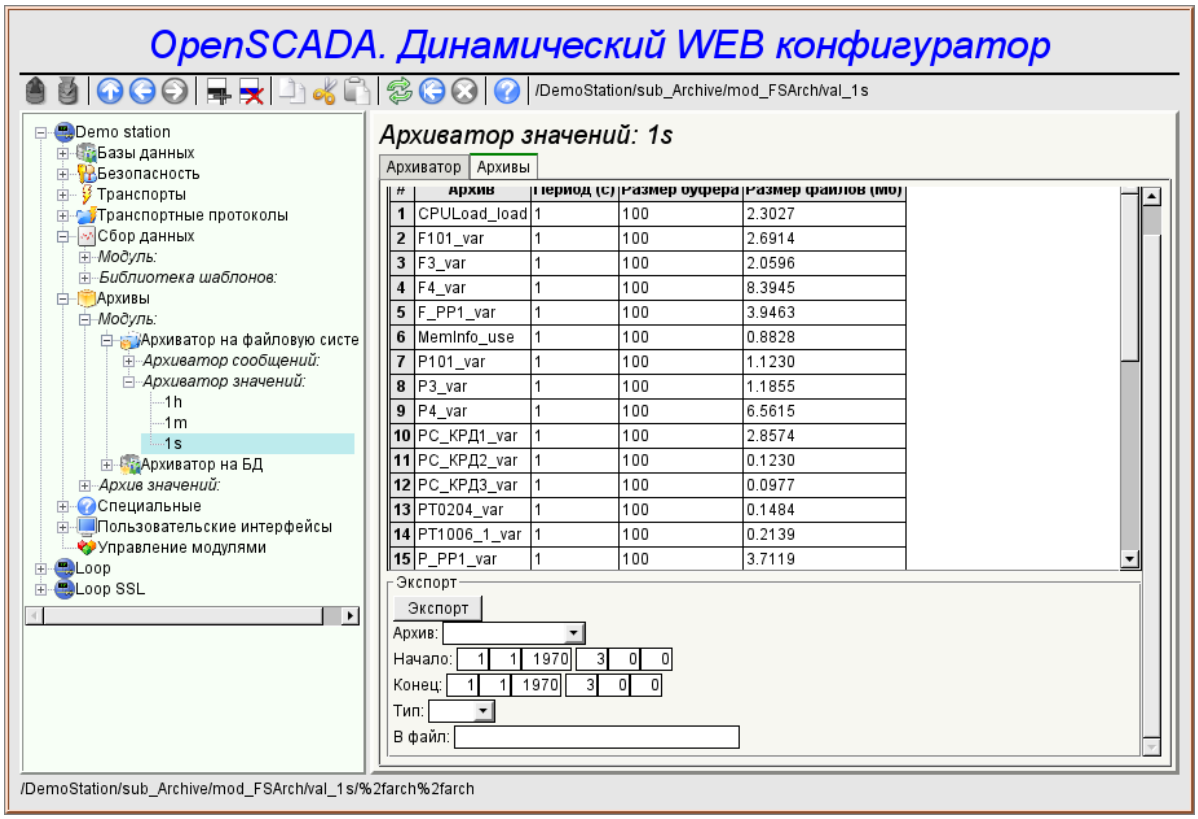


Рис.5. Команда.

4. Списки

Списки содержат группу базовых элементов одного типа. Операции над элементами доступны через контекстное меню по клику мышью на списке. Через элементы списка могут выполняться операции перехода на другие страницы. Списки могут быть индексированными. Пример списка приведен на рис. 6.

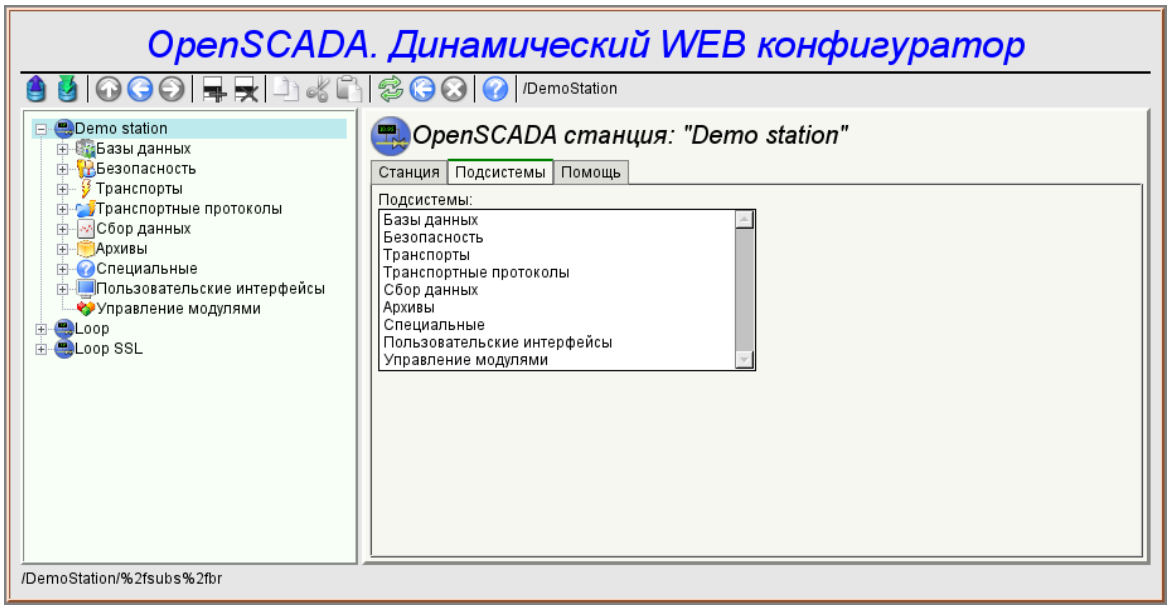


Рис.6. Список.

5. Таблицы

Таблицы содержат значения базовых элементов. Тип базового элемента является индивидуальным для каждой колонки. Пример таблицы приведен на рис. 7. Операции над структурой таблицы для редактируемых таблиц доступны посредством контекстного меню по клику на служебной колонке с номерами строк. Редактирование элементов таблицы производится путём двойного клика по нужной ячейке.

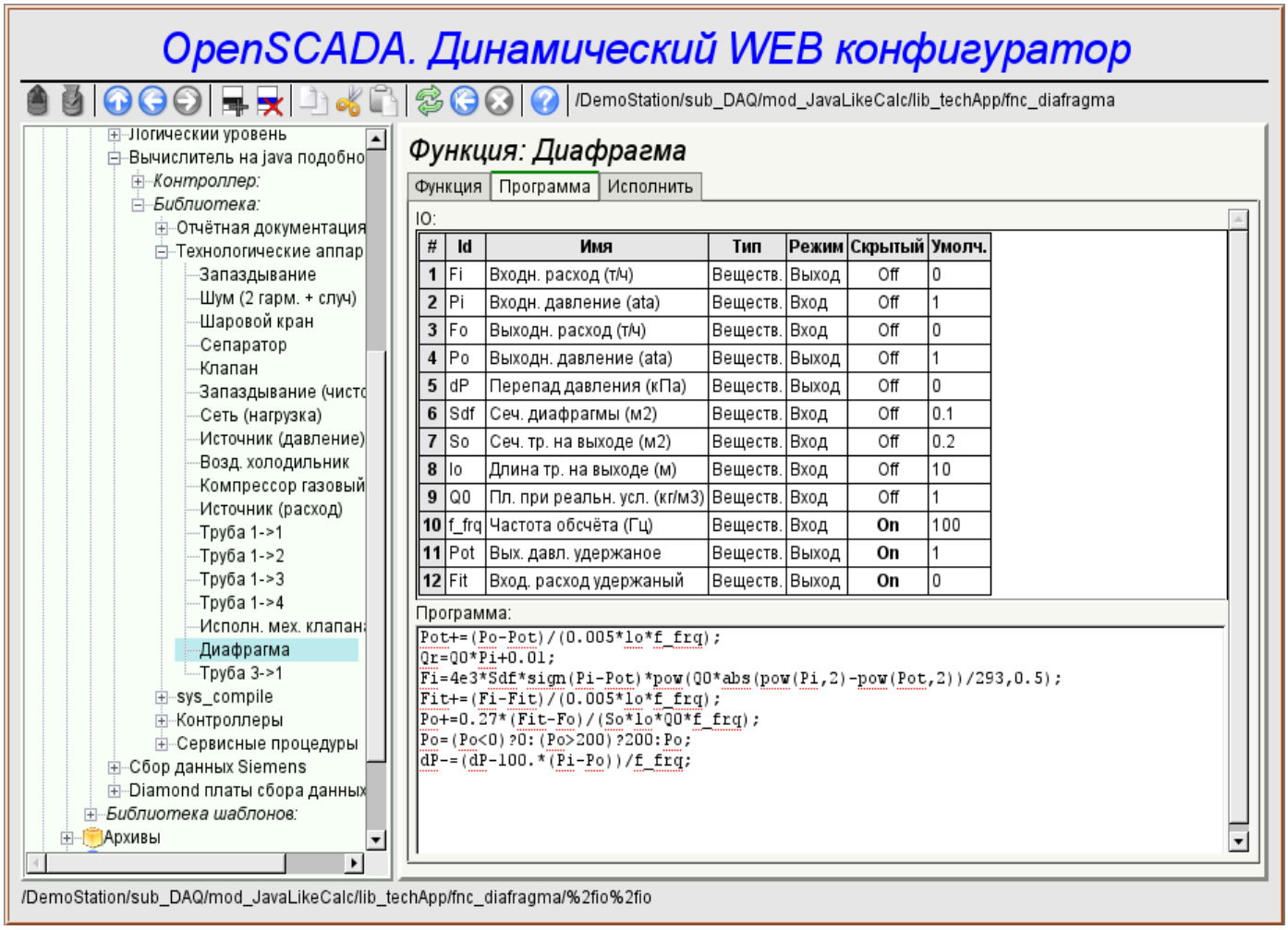


Рис.7. Таблица.

6. Изображения

Изображения призваны передавать графическую информацию в конфигураторы. Пример изображения приведен на рис. 8.

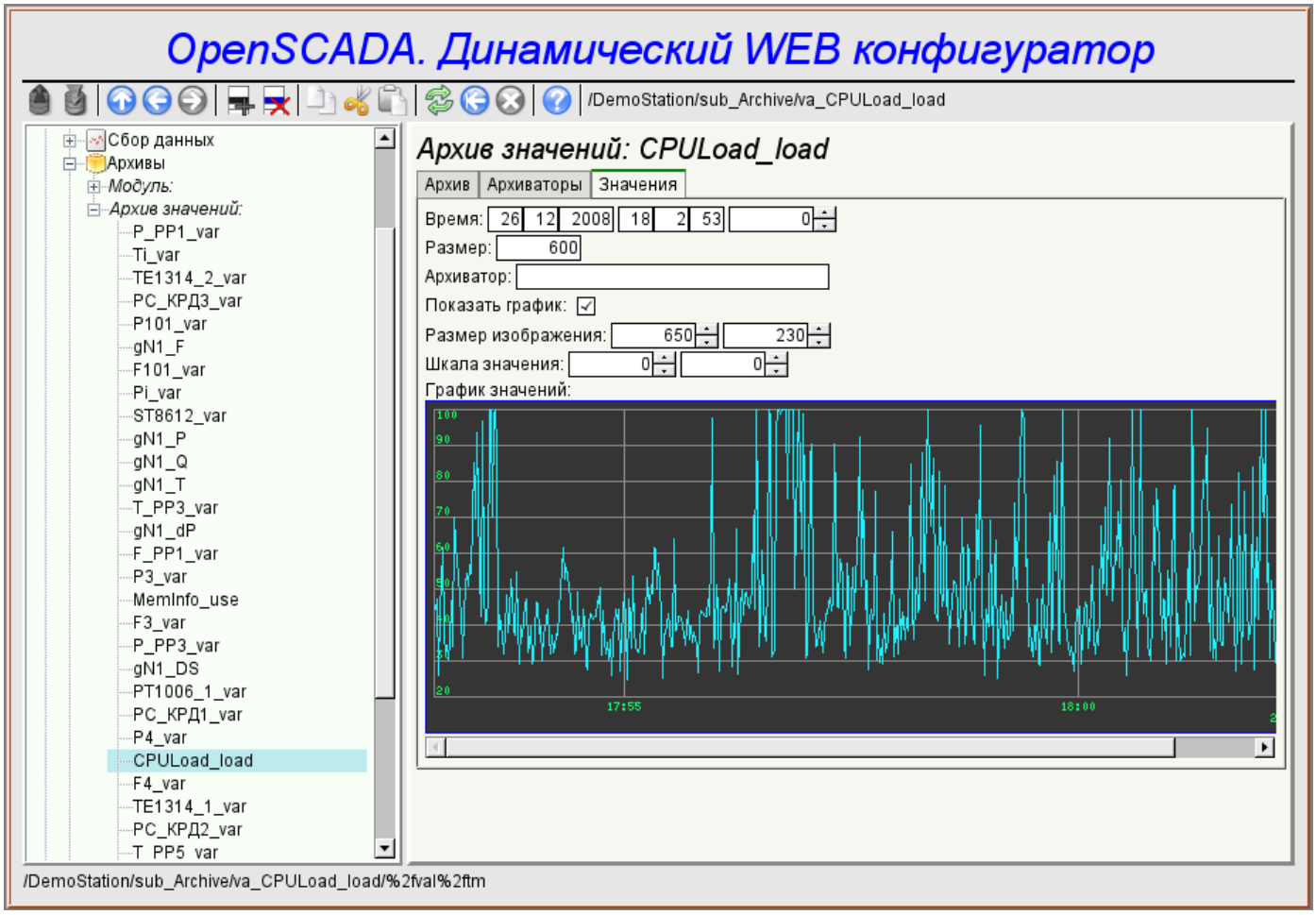


Рис.8. Изображение.

7. Ошибки

Представление конфигуратора может несколько отличаться на разных типах браузеров. Это связано с тем, что в основе данного модуля лежит много достаточно сложных технологий, а также отличий их реализации на разных типах WEB-движков.

Кроме того, каждый Web-браузер содержит собственные ошибки. Часть ошибок была обойдена в процессе реализации, однако часть осталась в виду значительных трудностей в их обходе или-же фактической невозможности это сделать.

В данном разделе содержится таблица перечня обнаруженных ошибок WEB-браузеров, которые проявляются в конфигураторе.

Ошибка	Описание	Исправление
Mozilla FireFox 3.0.4 (устойчив, ошибок немного)		
Смещение всплывающего окна редактируемого комбо-бокса на 5 пикселов вверх-влево.	Проблема связана с тем, что вычисление абсолютной позиции элемента документа не добирает ровно 5 пикселов. Ошибка на 5 пикселов видна в сопоставлении с координатами курсора мыши и положением вновь создаваемого абсолютно-позиционируемого окна. Алгоритм вычисления положения: *for(; obj != null; obj = obj.offsetParent) posX += obj.offsetLeft;*	Для устранения этой ошибки к расчётному значению на этом браузере прибавляется 5 пикселов.
В элементе списка (<select size="10"/>) всегда отображается вертикальный скроллер и никогда не включается горизонтальный.	Данный элемент активно используется для формирования контекстного меню и выпадающего списка редактируемого комбобокса.	Для обхода ошибки браузера пришлось включить список в блок со скролом самого блока.
Не обновляется поле изображения.	Для исключения необходимости перестройки конфигурируемой страницы при обновлении значений полей в дереве объектов структуры страницы, полученной с помощью XMLHttpRequest, создаются свойства со ссылками на объекты тегов полей (addr_lab, addr_val_w). В объектах с именем тега “img” эти свойства браузером не создаются.	Проблема не решена
Opera (устойчив, ошибок немного)		
Не включается скроллер блока страницы. Например, при отображении широких изображений тренда.	Блок является фиксированным с параметрами { overflow: auto; width: 600px; }, однако при превышении размера внутренних элементов скроллер не включается.	Проблема не решена.
Konqueror (очень не устойчив на динамических ресурсах и содержит множество ошибок)		
Стабильные падения браузера.	Браузер неоднократно и стабильно падает в моменты вычисления JavaScript и при операциях с внешними окнами.	Проблема не решена.
Не возвращается скролл дерева навигации.	Если дерево навигации развернуть до появления вертикального скрола, затем прокрутить его вниз, после чего свернуть крупную ветку, то вертикальный скролл исчезает, а часть дерева остаётся невидимой за верхней частью блока. Т.е. содержимое блока не обновляется.	Проблема не решена.

Ошибка	Описание	Исправление
Не обновляются изображения	В полях изображений для обновления изображения из сервера изменяется свойство “src”. Браузер этого не чувствует, иногда даже обновляет размер рамки, а изображение не обновляет. Методы для предотвращения кеширования изображения применяются, но не помогают.	Проблема не решена.
Схватывание изображений кнопок	JavaScript модуля используют не ассинхронные, а синхронные запросы к серверу для сохранения последовательности действий. В моменты такого запроса, если он был вызван событием от изображения (изображение является кнопкой), изображение схватывается как будто для переноса, даже по кратковременным кликам мыши.	Проблема не решена.
Невозможно вставить новый элемент в дерево объектов, полученное как результат XMLHttpRequest	Для контроля за модификацией структуры конфигурационной страницы использовалось приведение текущего дерева структуры к новой, только что полученной из XMLHttpRequest. В момент вставки нового элемента в дерево структуры посредством insertBefore() происходит ошибка «DOM error 4". Если вставка производится в дерево, созданное с нуля (а не из XMLHttpRequest), эта ошибка не происходит. Похоже проблема заключается в отличии объекта “document” владельца этих деревьев. В такое дерево невозможно добавить узел, созданный как document.createElement(). Вставляются только соданные как mytree.ownerDocument.createElement().	Процедура проверки структуры была упрощена и сведена к определению факта изменения.
Не работают механизмы формирования контекстного меню на Konqueror 4.	Обычно для формирования контекстного меню используется обработчик oncontextmenu, на Firefox и IE или onmousedown с обработкой правой клавиши на в остальных браузерах. В Konqueror вообще oncontextmenu не работает, а onmousedown работает только в Konqueror 3.5.	Проблема не решена.

Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>

Модуль:	VCAEngine
Имя:	Движок среды визуализации и управления
Тип:	Пользовательские интерфейсы
Источник:	ui_VCAEngine.so
Версия:	1.0.0
Автор:	Роман Савоченко
Описание:	Основной движок среды визуализации и управления.
Лицензия:	GPL

Модуль VCAEngine предоставляет движок среды визуализации и управления (СВУ) в систему OpenSCADA. Сам модуль не реализует визуализации СВУ, а содержит данные в соответствии с идеологией "Модель/данные – Интерфейс". Визуализация данных этого модуля выполняется модулями визуализации СВУ, например, модулем Vision и WebVision.

Среда визуализации и управления (СВУ) является неотъемлемой составляющей SCADA системы. Она применяется на клиентских станциях с целью доступного предоставления информации об объекте управления и выдачи управляющих воздействий на объект. В различных практических ситуациях и условиях могут применяться СВУ, построенные на различных принципах визуализации. Например, это могут быть библиотеки виджетов QT, GTK+, wxWidgets или гипертекстовые механизмы на основе технологий HTML, XHTML, XML, CSS и JavaScript или сторонние приложения визуализации, реализованные на различных языках программирования Java, Python и т.д. Любой из этих принципов имеет свои преимущества и недостатки, комбинация которых может стать непреодолимым препятствием в возможности использования СВУ в том или ином практическом случае. Например, технологии вроде библиотеки QT позволяют создавать высокореактивные СВУ, что несомненно важно для станций оператора управления технологическим процессом (ТП). Однако, необходимость инсталляции данного клиентского ПО в отдельных случаях может сделать использование его невозможным. С другой стороны, Web-технологии не требуют инсталляции на клиентские системы и являются предельно многоплатформенными (достаточно создать ссылку на Web-сервер в любом Web-браузере), что наиболее важно для различных инженерных и административных станций, но реактивность и надёжность таких интерфейсов ниже, что практически исключает их использования на станциях оператора ТП.

Система OpenSCADA имеет предельно гибкую архитектуру, которая позволяет создавать внешние интерфейсы, в том числе и пользовательские, на любой основе и на любой вкус. Например, среда конфигурации системы OpenSCADA доступна как на QT библиотеке, так и на Web-основе.

В тоже время независимое создание реализаций СВУ на различной основе может повлечь за собой невозможность использования данных конфигурации одной СВУ в другой. Что неудобно и ограничено с пользовательской стороны, а также накладно в плане реализации и последующей поддержки. С целью избежать этих проблем, а также создать в кратчайшие сроки полный спектр различных типов СВУ основан проект создания концепции СВУ. Результатом этого проекта и стал данный модуль движка(модели данных) СВУ, а также модули непосредственной визуализации Vision и WebVision.

1. Назначение

Данный модуль движка(модели данных) СВУ предназначен для формирования логической структуры СВУ и исполнения сеансов отдельных экземпляров проектов СВУ. Также модуль предоставляет все необходимые данные конечным визуализаторам СВУ как посредством локальных механизмов взаимодействия OpenSCADA, так и посредством интерфейса управления OpenSCADA для удалённого доступа.

Финальная версии этого модуля СВУ, построенная на основе данного модуля, обеспечит:

- три уровня сложности в формировании интерфейса визуализации, позволяющие органично осваивать и применять инструментарий по методике от простого к сложному:
  - формирование из шаблонных кадров путём назначения динамики (без графической конфигурации);
  - графическое формирование новых кадров путём использования готовых элементов визуализации из библиотеки (мнемосхемы);
  - формирование новых кадров, шаблонных кадров и элементов отображение в библиотеки.
- построение интерфейсов визуализации различной сложности, начиная от простых плоских интерфейсов мониторинга и заканчивая полноценными иерархическими интерфейсами, используемыми в SCADA системах;
- предоставление различных способов формирования и конфигурации пользовательского интерфейса, основанных на различных интерфейсах графического представления (QT, Web, Java ...) или-же посредством стандартного интерфейса управления системой OpenSCADA;
- смену динамики в процессе исполнения;
- построение новых шаблонных кадров на уровне пользователя и формирование специализированных под область применения библиотек кадров (например включение кадров параметров, графиков и других элементов с увязкой их друг с другом) в соответствии с теорией вторичного использования и накопления;
- построение новых пользовательских элементов визуализации и формирование специализированных под область применения библиотек кадров в соответствии с теорией вторичного использования и накопления;
- описание логики новых шаблонных кадров и пользовательских элементов визуализации как простыми связями, так и лаконичным, полноценным языком пользовательского программирования;
- возможность включения в пользовательские элементы визуализации функций (или кадров вычисления функций) объектной модели OpenSCADA, практически связывая представление с алгоритмом вычисления (например, визуализируя библиотеку моделей аппаратов ТП для последующего визуального построения моделей ТП);
- разделение данных пользовательских интерфейсов и интерфейсов представления этих данных, позволяющее строить интерфейс пользователя в одной среде, а исполнять во многих других (QT, Web, Java ...);
- возможность подключения к исполняющемуся интерфейсу для наблюдения и коррекции действий (например, при обучении операторов и контроля в реальном времени за его действиями);
- визуальное построение различных схем с наложением логических связей и последующим централизованным исполнением в фоне (визуальное построение и исполнение математических моделей, логических схем, релейных схем и иных процедур);
- предоставление функций объектного API в систему OpenSCADA может использоваться для управления свойствами интерфейса визуализации из пользовательских процедур;
- построение серверов кадров, элементов визуализации и проектов интерфейсов визуализации с возможностью обслуживания множественных клиентских соединений;
- простая организация клиентских станций на различной основе (QT, Web, Java ...) с подключением к центральному серверу;
- полноценный механизм разделения полномочий между пользователями, позволяющий создавать и исполнять проекты с различными правами доступа к его компонентам;

- гибкое формирование правил сигнализаций и уведомления с учётом и поддержкой различных способов уведомления;
- поддержка пользовательского формирования палитры и шрифтовых предпочтений для интерфейса визуализации;
- поддержка пользовательского формирования карт событий под различное оборудование управления и пользовательские предпочтения;
- поддержка профилей пользователей, позволяющая определять различные свойства интерфейса визуализации (цветовая гамма, шрифтовые особенности, предпочтительные карты событий);
- гибкое хранение и распространение библиотек виджетов, кадров и проектов интерфейсов визуализации в БД, поддерживаемых системой OpenSCADA; практически пользователю нужно только зарегистрировать полученную БД с данными.

2. Конфигурация и формирование интерфейсов СВУ

Сам модуль не содержит инструмента визуального формирования интерфейсов СВУ, основанного на одном из механизмов. Такие инструменты могут предоставляться модулями конечной визуализации СВУ, например, модулем Vision такой инструмент предоставляется.

Хотя визуального инструмента формирования СВУ модулем не предоставляется, для управления логической структурой предоставляется интерфейс, реализованный на основе интерфейса управления OpenSCADA, а значит доступный для использования в любом конфигураторе системы OpenSCADA. Диалоги этого интерфейса рассмотрены далее в контексте рассмотрения архитектуры модуля и его данных.

3. Архитектура

Любая СВУ может работать в двух режимах – разработки и исполнения. В режиме разработки формируется интерфейс СВУ, его компоненты и определяются механизмы взаимодействия. В режиме исполнения выполняется формирование интерфейса СВУ и производится взаимодействие с конечным пользователем на основе разработанных СВУ.

Интерфейс СВУ формируется из кадров, каждый из которых, в свою очередь, формируется из элементов примитивов или пользовательских элементов интерфейса. При этом пользовательские элементы интерфейса также формируются из примитивов или других пользовательских элементов. Таким образом обеспечивается иерархичность и повторное использования уже разработанных компонентов.

Кадры и пользовательские элементы размещаются в библиотеках виджетов. Из элементов этих библиотек формируются проекты интерфейсов конечной визуализации СВУ. На основе же этих проектов формируются сеансы визуализации.

Описанная структура СВУ приведена на рис. 3.

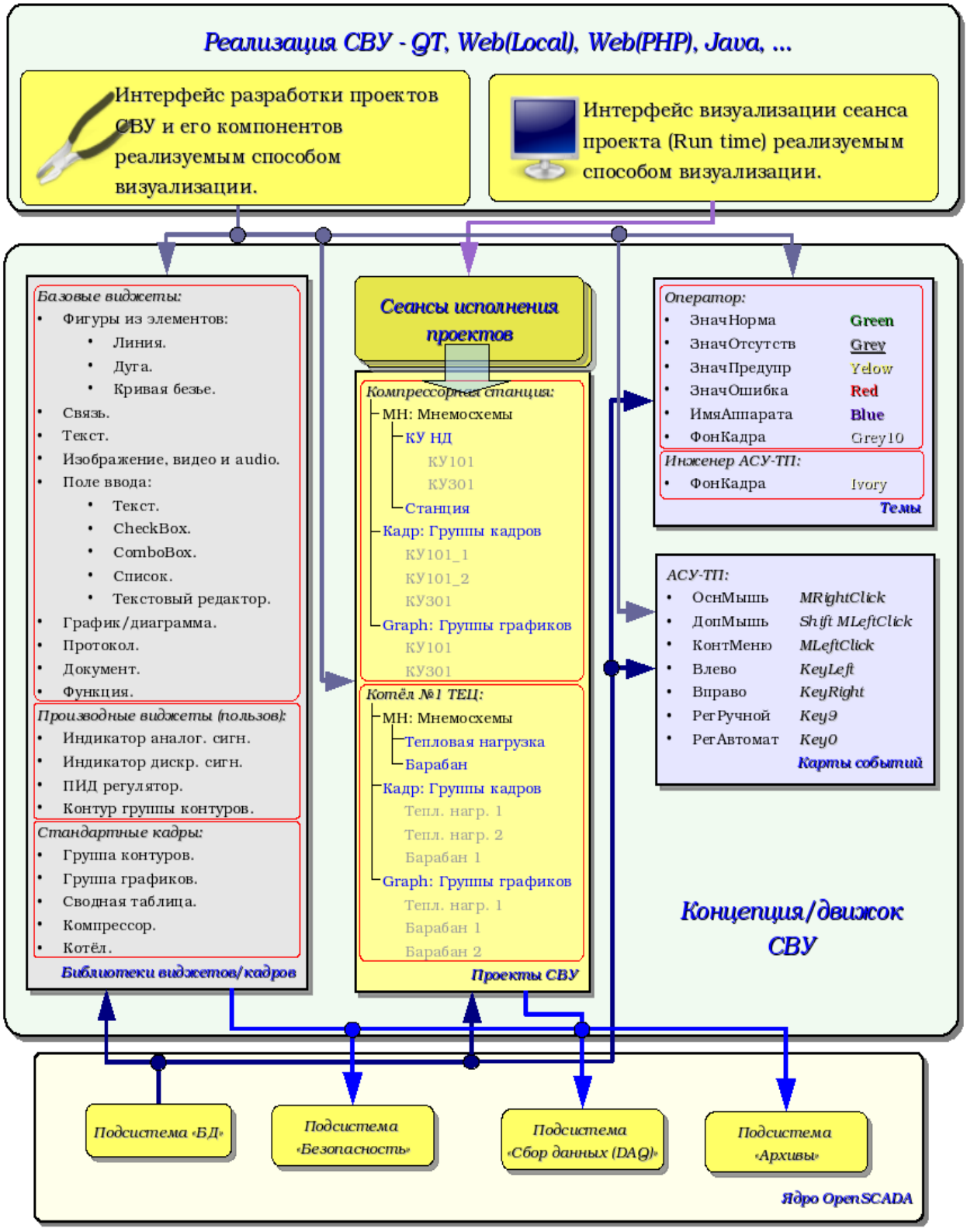


Рис.3 Обобщённая структура СВУ.

Данная архитектура СВУ позволяет реализовать поддержку трёх уровней сложности процесса разработки интерфейсов управления:

- Формирования интерфейса ВУ(визуализации и управления) с помощью библиотеки шаблонных кадров путём помещения шаблонов кадров в проект и назначения динамики.
- В дополнении к первому уровню производится формирование собственных кадров на основе библиотеки производных и базовых виджетов. Возможно как прямое назначение динамики в виджете, так и последующее её назначение в проекте.
- В дополнении ко второму уровню производится самостоятельное формирование

производных виджетов, новых шаблонных кадров, а также кадров с использованием механизма описания логики взаимодействия и обработки событий на одном из языков пользовательского программирования системы OpenSCADA.

3.1. Кадры и элементы отображения(виджеты)

Кадр - это окно, непосредственно предоставляющее информацию пользователю в графической и/или текстовой форме. Группа взаимосвязанных кадров формирует цельный пользовательский интерфейс ВУ.

Содержимое кадра формируется из элементов отображения(виджетов). Виджеты могут быть базовыми примитивами (различные плоские фигуры, текст, тренд и т.д.) и производными (сформированные из базовых или других производных виджетов). Все виджеты группированы по библиотекам. В процессе работы пользователь может формировать собственные библиотеки производных виджетов.

Собственно кадр также является виджетом, который используется в роли конечного элемента визуализации. А это значит, что библиотеки виджетов могут хранить и заготовки кадров, и шаблоны результирующих страниц пользовательского интерфейса.

Кадры и виджеты являются пассивными элементами, которые обычно не содержат связей с динамикой и другими кадрами, а только предоставляют информацию о свойствах виджета и характере динамики(конфигурации), подключаемой к свойствам кадра. Активированные кадры, т.е. содержащие ссылки на динамику и активные связи, формируют пользовательский интерфейс и хранятся в проектах. В некоторых случаях возможно прямое назначение динамики в заготовках кадров.

Производные кадры/виджеты могут содержать другие виджеты(вложенные), которые могут быть склеены(связаны) логикой один с другим с помощью одного из языков пользовательского программирования, доступного в системе OpenSCADA (рис.3.1.1).

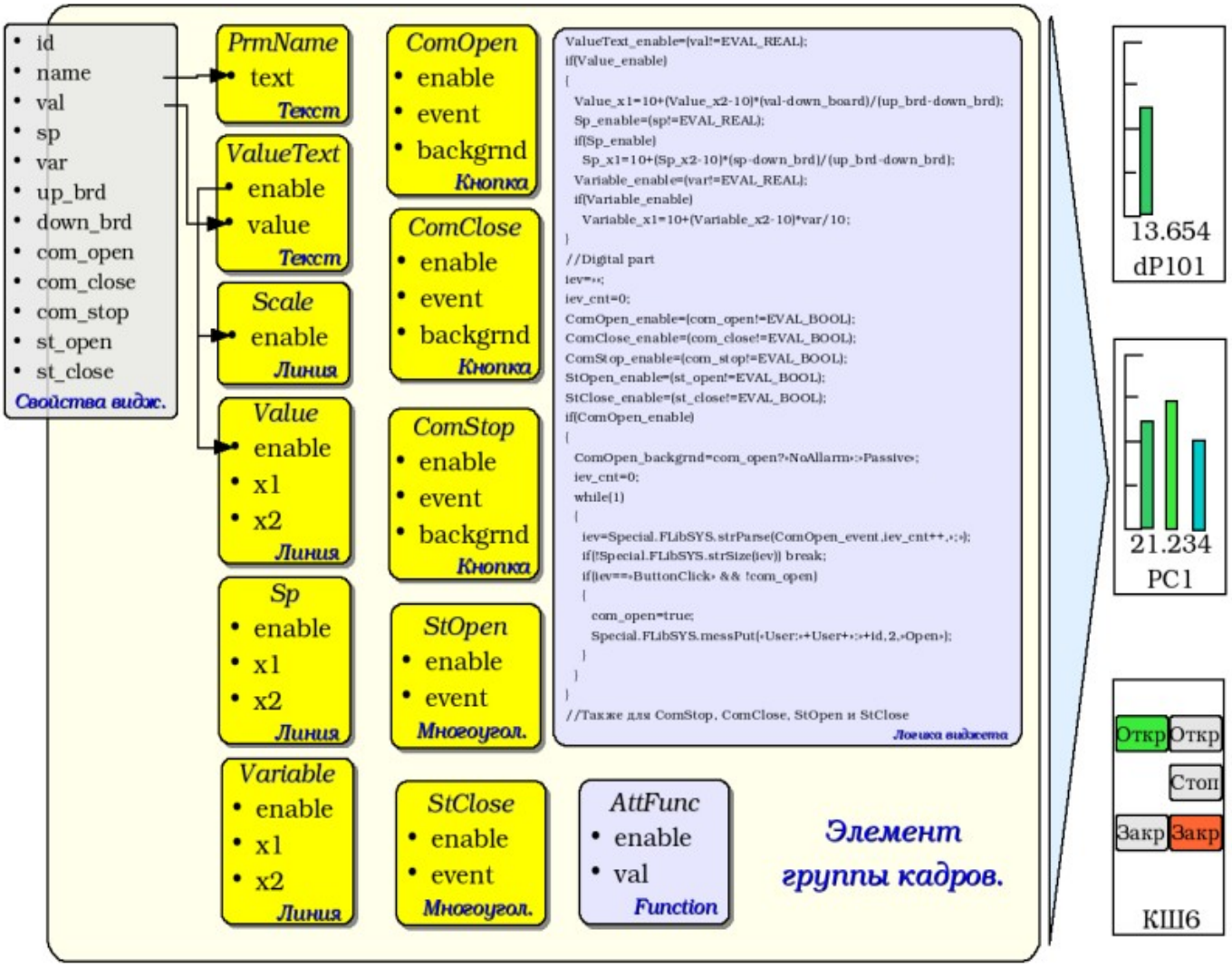


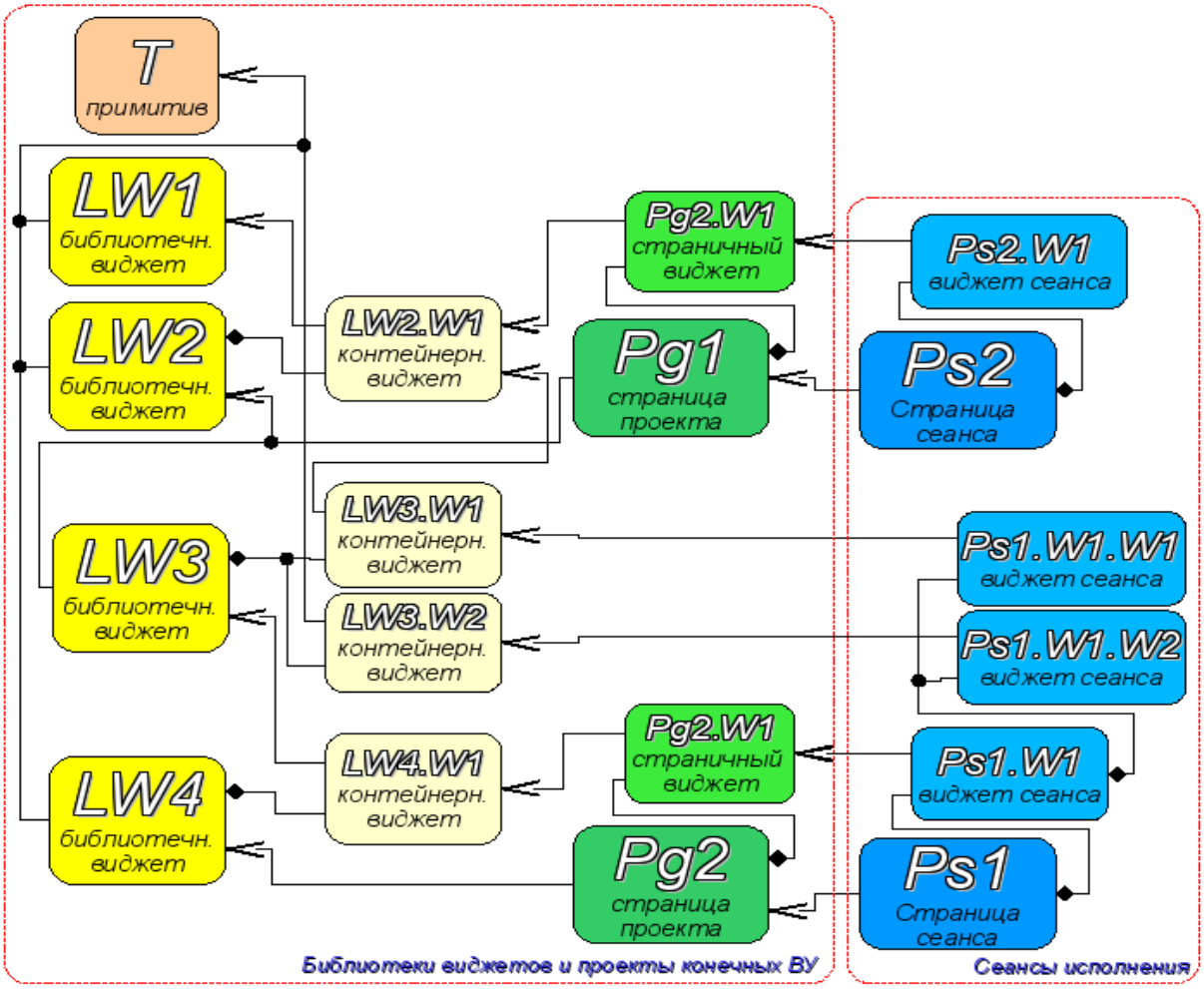
Рис.3.1.1 Пример структуры производного виджета.

Виджет является элементом, посредством которого обеспечивается:

- визуализация оперативной и архивной информации ведения ТП;
- сигнализация про нарушения ведения ТП;
- переключение между кадрами ТП;
- управление технологическим оборудованием и параметрами ведения ТП.

Настройка и связывание виджетов производится посредством их свойств. Родительский виджет и виджеты, содержащиеся в нем, могут дополняться пользовательскими свойствами. В последствии пользовательские и статические атрибуты связываются со свойствами вложенных виджетов посредством внутренней логики. Для отображения динамики (т.е. текущих и архивных данных) свойства виджетов динамизируются, т.е связываются с атрибутами параметров OpenSCADA или свойствами других виджетов. Использование для связывания вложенных виджетов внутренней логикой доступных языков пользовательского программирования системы OpenSCADA снимает вопрос реализации сложной логики визуализации, обеспечивая тем самым высокую гибкость. Практически можно создавать полностью динамизированные кадры со сложными взаимосвязями на уровне пользователя.

Между виджетами на различных уровнях иерархии выстраиваются достаточно сложные наследственные связи, которые определяются возможностью использования одних виджетов другими, начиная с библиотечного виджета и заканчивая виджетом сеанса. Для разъяснения этих особенностей взаимодействия на рис. 3.1.2 изображена исчерпывающая карта «использующего» наследования.



Терминальный виджет – Конечный элемент визуализации, или примитив. На стороне визуализации приобретает соответствующий визуальный образ.

Библиотечный виджет – Хранимый библиотекой виджет. Обязательно наследует визуальный образ терминального виджета и переопределяет его данные. Наследование терминального виджета может быть как прямое, так и посредством нескольких промежуточных элементов.

Контейнерный виджет библиотеки – Фактически является ссылкой на другой виджет в библиотеке (LW2.W1 -> LW1) или ссылку контейнера библиотеки (LW3.W1 -> LW2.W1).

Страница проекта – Элемент интерфейса визуализации и управления (ВУ) - страница, используется для построения иерархического интерфейса ВУ конечного пользователя.

Страничный виджет – Элемент страницы, доопределяющий данные библиотечного виджета к нуждам страницы проекта.

Страница сеанса – Страница сеанса для исполнения страницы проекта в контексте цельного интерфейса ВУ.

Виджет сеанса – Элемент конечной визуализации. Выстраиваются в иерархическую зависимость, соответствующую наследованию терминальных виджетов в контейнерных виджетах библиотеки виджетов и проекта.

Рис.3.1.2 Карта «использующего» наследования компонентов концепции/движка

На уровне сеансов виджет содержит кадр значений процедуры обсчёта. Этот кадр инициируется и используется в случае наличия процедуры обсчёта. В момент инициализации создаётся перечень параметров процедуры и выполняется компиляция процедуры с этими параметрами в модуле, реализующем выбранный язык программирования и закодированным полным именем виджета. Скомпилированная функция подключается к кадру значений процедуры обсчёта. Далее выполняется вычисление с периодичностью сеанса.

Вычисление и обработка виджета в целом выполняется в следующей последовательности:

- выбирается события, доступные на момент вычисления из атрибута “event” виджета;
- события загружаются в параметр “event” кадра вычисления;
- загружаются значения по входным связям в кадр вычисления;
- загружаются значения специальных переменных в кадр вычисления (f_frq, f_start и f_stop);
- загружаются значения выбранных параметров виджета в кадр вычисления;
- вычисление;
- выгрузка значений кадра вычисления в выбранные параметры виджета;
- выгрузка события из параметра “event” кадра вычисления;
- обработка событий и передача необработанных на уровень выше.

3.2. Проект

Непосредственная конфигурация и свойства конечного интерфейса визуализации содержаться в проекте интерфейса визуализации СВУ. Может быть создано множество проектов интерфейсов визуализации.

Каждый проект включает кадры из библиотек кадров/виджетов. Кадр предоставляет инструмент для привязки динамики к описанным в нём свойствам. Все свойства кадра могут быть связаны с динамикой или разрешены константами, а могут выступать в роли шаблона для формирования производных страниц. Фактически каждый кадр может содержать множество страниц с собственной динамикой. Данный механизм позволяет предельно упростить процесс создания однотипных кадров инженером АСУ-ТП или пользователем системы OpenSCADA для простого мониторинга. Примером таких однотипных кадров могут быть: группы контуров, группы графиков, протоколы и различные сводные таблицы. Мнемосхемы технологических процессов редко подпадают под такую схему и будут формироваться прямо в описании кадра.

Для предоставления возможности создания сложных иерархических интерфейсов ВУ кадры, помещённые в проект, могут группироваться путём именования в иерархическом виде и соответствующей визуализации в виде дерева. В придачу к этому предусматривается механизм ассоциативного описания вызова кадров посредством регулярных выражений.

Пример иерархического представления компонентов проекта классического интерфейса ВУ технологического процесса с описанием выражений стандартных вызовов приведен на рис. 3.2.

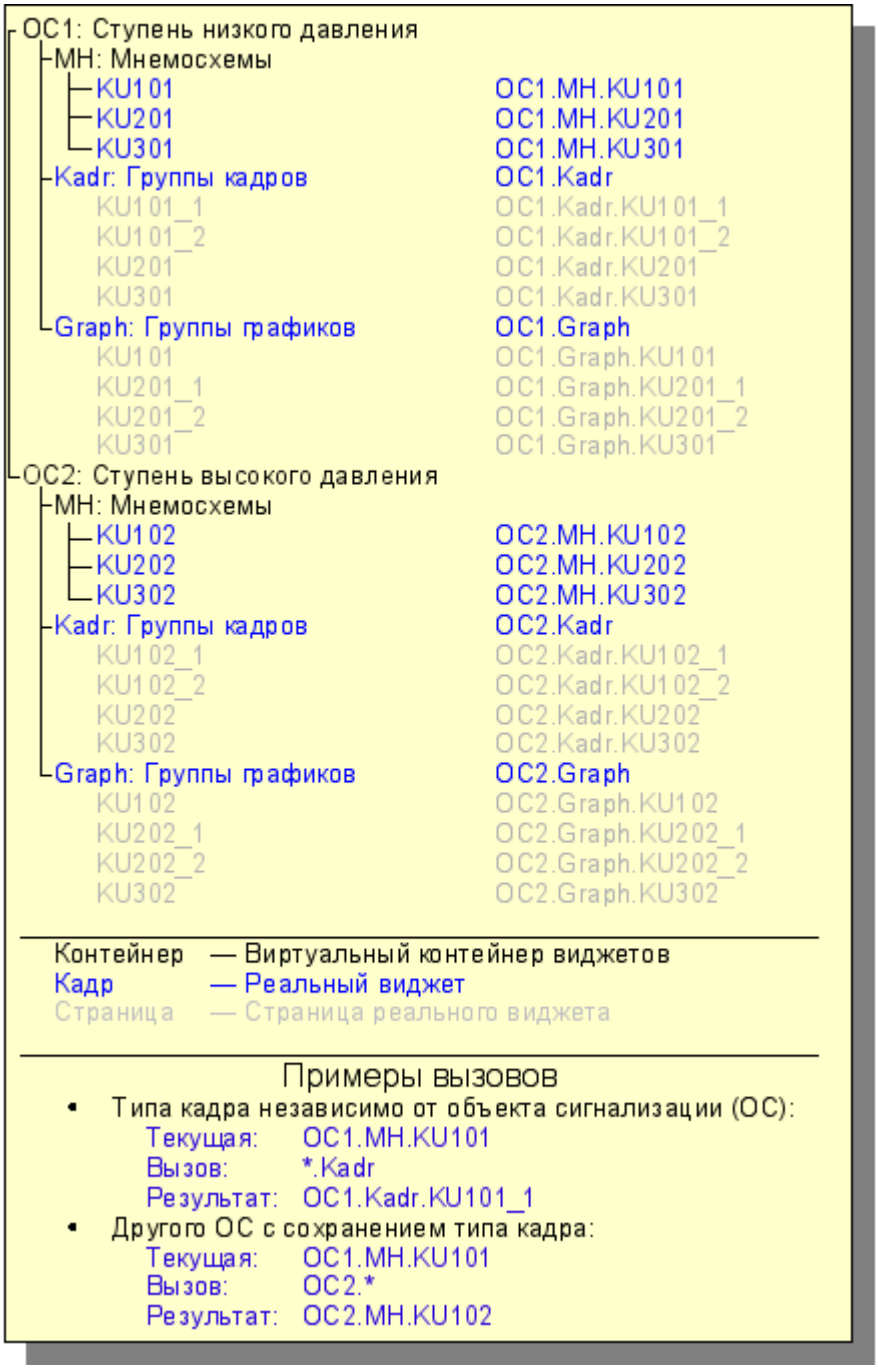


Рис.3.2 Иерархическое представления компонентов проекта классического интерфейса ВУ технологического процесса.

В соответствии с рис.3.1.2 объекты сессии проекта наследуются от абстрактного объекта "Widget" и используют соответствующие объекты проекта. Так, сессия ("Session") использует проект ("Project") и формирует развёрнутое дерево на основе него. Страница проекта "Page" прямо используется страницей сессии "SessPage". Остальные объекты ("SessWdg") разворачиваются в соответствии с иерархией элементов страницы (рис.3.1.2).

В дополнение к стандартным свойствам абстрактного виджета ("Widget") элементы страницы и сами страницы сессии получают свойства: хранения кадра значений вычислительной процедуры, обсчёта процедур и механизм обработки событий. Страницы сессии в дополнение ко всему содержат контейнер следующих по иерархии страниц. Сессия в целом обсчитывается с указанной периодичностью и в последовательности:

- "Страница верхнего уровня" -> "Страница нижнего уровня"
- "Виджет нижнего уровня" -> "Виджет верхнего уровня"

Такая политика позволяет обходить страницы в соответствии с иерархией, а событиям в виджетах всплывать на верх за одну итерацию.

В сессии реализована поддержка специальных свойств страниц:

- *Container* - страница является контейнером для нижележащих страниц;
- *Template* - страница является шаблоном для нижележащих страниц;
- *Empty* - пустая, неактивная, страница; это свойство используется совместно со свойством *Container* для организации логических контейнеров.

На основе этих свойств реализованы следующие типы страниц:

- *Standard* - Стандартная страница (не установлено ни одно из свойств). Является полноценной конечной страницей.
- *Container* - Полноценная страница со свойством контейнера (*Container*).
- *Logical container* - Логический контейнер, фактически не являющийся страницей (*Container*|*Empty*). Выполняет свойство промежуточного и группирующего элемента в дереве страниц.
- *Template* - Страница шаблон (*Template*). Чистая шаблонная страница используется для описания общих свойств и доопределения их в частном порядке во вложенных страницах.
- *Container and template* — Страница шаблон и контейнер (*Template*|*Container*). Совмещает функции шаблона и контейнера.

Переключение, открытие, замещение и навигация по страницам реализованы на основе обработки событий по сценарию в атрибуте активного виджета "evProc". Сценарий этого атрибута записывается в виде списка команд с синтаксисом: **<event>:<evSrc>:<com>:<prm>**. Где:

- *event* - ожидаемое событие;
- *evSrc* - путь вложенного виджета-источника события;
- *com* - команда сессии;
- *prm* - параметр команды.

Реализованы следующие команды:

- *open* - Открытие страницы. Открываемая страница указывается в параметре <prm> как на прямую, так и в виде шаблона (например: /pg_so/1/*/*).
- *next* - Открытие следующей страницы. Открываемая страница указывается в параметре <prm> в виде шаблона (например: /pg_so/*/*/$).
- *prev* - Открытие предыдущей страницы. Открываемая страница указывается в параметре <prm> в виде шаблона (например: /pg_so/*/*/$).

Специальные символы шаблона расшифровываются следующим образом:

- *pg_so* - прямое имя требуемой страницы с префиксом. Требует обязательного соответствия и используется для идентификации предыдущей открытой страницы;
- *1* - имя новой страницы в общем пути без префикса. Игнорируется при обнаружении предыдущей открытой страницы;
- * - страница берётся с имени предыдущей открытой страницы или подставляется первая доступная страница, если предыдущая открытая страница отсутствует;
- *$* - указывает на место открытой страницы, относительно которой необходимо искать следующую или предыдущую.

Для понимания работы механизма шаблонов приведём несколько реальных примеров:

- *Переключение объекта сигнализации:*
  Команда: open:/pg_so/2/*/*
  Было: /pg_so/pg_1/pg_mn/pg_1
  Стало: /pg_so/pg_2/pg_mn/pg_1
- *Переключение вида:*
  Команда: open:/pg_so/*/gkadr/*
  Было: /pg_so/pg_1/pg_mn/pg_1
  Стало: /pg_so/pg_1/pg_gkadr/pg_1
- *Следующая/предыдущая страница вида:*
  Команда: next:/pg_so/*/*/$
  Было: /pg_so/pg_1/pg_mn/pg_1
  Стало: /pg_so/pg_1/pg_mn/pg_2

В качестве примера приведём сценарий обеспечения работы главной страницы интерфейса

пользователя:

```
ws_BtPress:/prev:prev:/pg_so/*/*/$
ws_BtPress:/next:next:/pg_so/*/*/$
ws_BtPress:/go_mn:open:/pg_so/*/mn/*
ws_BtPress:/go_graph:open:/pg_so/*/ggraph/*
ws_BtPress:/go_cadr:open:/pg_so/*/gcadr/*
ws_BtPress:/go_view:open:/pg_so/*/gview/*
ws_BtPress:/go_doc:open:/pg_so/*/doc/*
ws_BtPress:/go_resg:open:/pg_so/rg/rg/*
ws_BtPress:/so1:open:/pg_so/1/*/*
ws_BtPress:/so2:open:/pg_so/2/*/*
ws_BtPress:/so3:open:/pg_so/3/*/*
ws_BtPress:/so4:open:/pg_so/4/*/*
ws_BtPress:/so5:open:/pg_so/5/*/*
ws_BtPress:/so6:open:/pg_so/6/*/*
ws_BtPress:/so7:open:/pg_so/7/*/*
ws_BtPress:/so8:open:/pg_so/8/*/*
ws_BtPress:/so9:open:/pg_so/9/*/*
ws_BtPress:*:open:/pg_control/pg_terminator
```

В связке с вышеописанным механизмом на стороне визуализации (RunTime) построена логика, регулирующая, каким образом открывать страницы. Логика построена на следующих атрибутах базового элемента "Box":

- *pgOpen* - Признак "Страница открыта".
- *pgNoOpenProc* - Признак "Исполнять страницу даже если она не открыта".
- *pgOpenSrc* - Содержит адрес виджета или страницы, открывшей текущую. В случае вложенного контейнерного виджета здесь содержится адрес включаемой страницы. Для открытия страницы из скрипта достаточно здесь указать адрес виджета-источника открытия.
- *pgGrp* - Группа страниц. Используется для связки контейнеров страниц со страницами в соответствии с общей группой.

Логика определения способа открытия страниц работает следующим образом:

- если страница имеет группу "main" или совпадает с группой страницы в главном окне или нет страницы на главном окне, то открывать страницу в главном окне;
- если страница имеет группу, которая совпадает с группой одного из контейнеров текущей страницы, то открыть в этом контейнере;
- если источник открытия страницы совпадает с текущей страницей, то открыть в виде дополнительного окна над текущей страницей;
- передать вызов на запрос открытия дополнительным окнам с обработкой у каждого по первым трем пунктам;
- если никто из родственных окон не открыл новую страницу, то открыть её как родственное окно главного окна.

3.3. Стили

Известно, что человек может иметь индивидуальные особенности в восприятии графической информации. Если эти особенности не учитывать то можно получить неприятие и отторжение пользователя к интерфейсу ВУ. Такое неприятие и отторжение может привести к фатальным ошибкам при управлении ТП, а также травмировать человека постоянной работой с таким интерфейсом. В SCADA системах приняты соглашения, которые регламентируют требования по созданию унифицированного интерфейса ВУ нормально воспринимаемого большинством людей. При этом практически отсутствует учёт особенностей людей с некоторыми отклонениями.

С целью учесть это обстоятельство, и предоставить возможность централизованно и просто изменять визуальные свойства интерфейса, проектом предусматривается реализация менеджера стилей интерфейса визуализации.

Пользователем может быть создано множество стилей, каждый из которых будет хранить цветовые, шрифтовые и другие свойства элементов кадра. Простая смена стиля позволит быстро преобразить интерфейс ВУ, а возможность назначения индивидуальной стиля для пользователя позволит учесть его индивидуальные особенности.

Для реализации этой возможности, при создании кадров, необходимо для свойств цвета, шрифта и других установить параметр "Конфигурация" (таблицы во вкладке "Обработка") в значение "Из стиля" (рис.3.7). А в параметре "Конфигурационный шаблон" указать идентификатор поля стиля. Далее это поле автоматически появится в менеджере стилей и его можно будет там менять. Менеджер стилей доступен на странице конфигурации проекта во вкладке "Стили" (рис. 3.3). На этой вкладке можно создавать новые стили, удалять старые, изменять поля стиля и удалять не нужные.

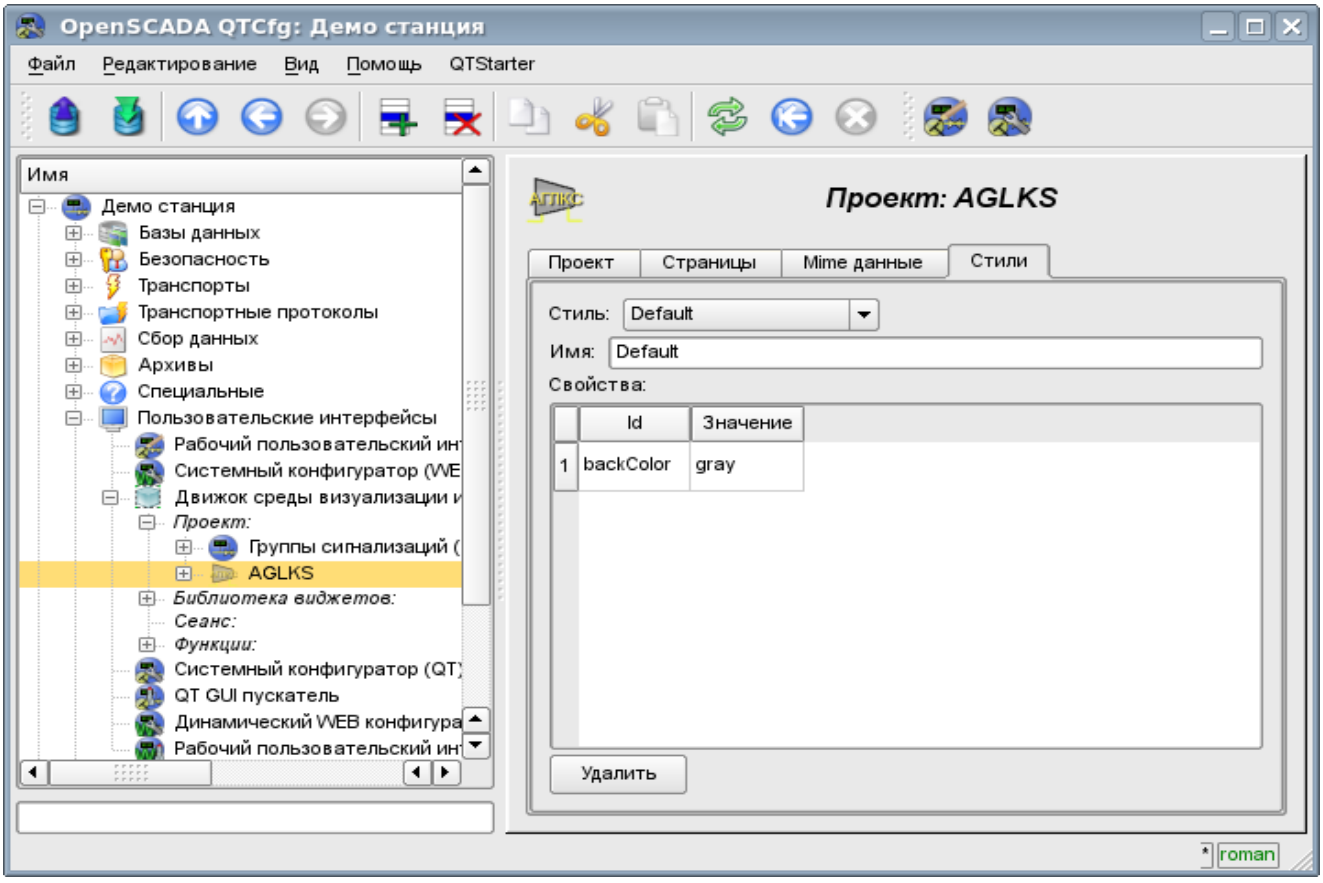


Рис. 3.3 Вкладка "Стили" страницы конфигурации проекта.

В целом стили доступны начиная с уровня проектов. На уровне библиотек виджетов можно только определять поля стилей у виджетов. На уровне проекта при выборе стиля включается работа со стилями, что предполагает доступ к полям стилей вместо непосредственных значений атрибутов. Фактически это означает, что при чтении или записи атрибута виджета указанные операции будут

осуществляться с соответствующим полем выбранного стиля.

При запуске проекта на исполнения будет использован установленный в проекте стиль. В последствии пользователь может выбрать стиль из перечня доступных. Выбранный пользователем стиль будет сохранён и использован при следующем запуске проекта.

3.4. События, их обработка и карты событий

Учитывая спектр задач для которых может использоваться система OpenSCADA, нужно предусмотреть и механизм управления интерактивными пользовательскими событиями. Это связано с тем, что при решении отдельных задач встраиваемых систем устройства ввода и управления могут значительно отличатся. Впрочем достаточно взглянуть на обычную офисную клавиатуру и клавиатуру ноутбука что бы снять любые сомнения о необходимости менеджера событий.

Менеджер событий должен работать, используя карты событий. Карта событий - это список именованных событий с указанием его происхождения. Происхождением события может быть клавиатура, манипулятор мыши, джойстик и т.д. При возникновении события менеджер событий ищет его в активной карте и сопоставляет с именем события. Сопоставленное имя события помещается в очередь на обработку. Виджеты в этом случае должны обрабатывать полученную очередь событий.

Активная карта событий указывается в профиле каждого пользователя или устанавливается по умолчанию.

В целом предусмотрены четыре типа событий:

- события образов СВУ (префикс: ws_), например, событие нажатия кнопки - *ws_BtPress*;
- клавишные события (префикс: key_) - все события от клавиатуры и мыши в виде - *key_presAlt1*;
- пользовательские события (префикс: usr_) генерируются пользователем в процедурах обсчёта виджетов;
- мапированные события (префикс: map_) - события, полученные из карты событий.

Само событие представляет мало информации, особенно если его обработка происходит на уровнях выше. Для однозначной идентификации события и его источника событие в целом записывается следующим образом: "ws_BtPress:/curtime". Где:

ws_BtPress - событие;
/curtime - путь к дочернему элементу, сгенерировавшего событие.

В таблице 3.4 приведён перечень стандартных событий, поддержка которых должна быть обеспечена в визуализаторах СВУ.

Таблица 3.4. Стандартные события

Id	Описание
Клавиатурные события: **key_[pres|rels][Ctrl|Alt|Shift]{Key}**	
*SC#3b	Скан код клавиши.
*#2cd5	Код не именованной клавиши.
*Esc	"Esc".
*BackSpace	Удаления предыдущего символа - "<-".
*Return, *Enter	Ввод - "Enter".
*Insert	Вставка - "Insert".
*Delete	Удаление - "Delete".
*Pause	Пауза - "Pause".
*Print	Печать экрана - "Print Screen".
*Home	Дом - "Home".
*End	Конец - "End".
*Left	Влево - "<-".
*Up	Вверх - '^'.
*Right	Вправо - "->".

Id	Описание
*Down	Вниз - '\/'.
*PageUp	Страницы вверх - "PageUp".
*PageDown	Страницы вниз - "PageDown".
*F1 - *F35	Функциональная клавиша от "F1" до "F35".
*Space	Пробел - ' '.
*Apostrophe	Апостраф - '`'.
*Asterisk	Звёздочка на дополнительном поле клавиатуры - '*'.
*Plus	Плюс на дополнительном поле клавиатуры - '+'.
*Comma	Запятая - ','.
*Minus	Минус - '-'.
*Period	Точка - '.'.
*Slash	Наклонная черта - '\'.
*0 - *9	Цифра от '0' до '9'.
*Semicolon	Точка с запятой - ';'.
*Equal	Равно - '='.
*A - *Z	Клавиши букв латинского алфавита от 'A' до 'Z'.
*BracketLeft	Левая квадратная скобка - '['.
*BackSlash	Обратная наклонная линия - '/'.
*BracketRight	Правая квадратная скобка - ']'.
*QuoteLeft	Левая кавычка - '".
События клавиатурного фокуса.	
ws_FocusIn	Фокус получен виджетом.
ws_FocusOut	Фокус утерян виджетом.
Мышиные события:	
key_mouse[Pres|Rels][Left|Right|Midle]	Нажата/отпущена кнопка мыши.
key_mouseDblClick	Двойное нажатие левой кнопки мыши.
События квитирования на стороне визуализатора.	
ws_alarmLev	Квитирование всех нарушений всеми способами уведомления.
ws_alarmLight	Квитирование всех нарушений уведомления миганием/светом.
ws_alarmAlarm	Квитирование всех нарушений уведомления гудком.
ws_alarmSound	Квитирование всех нарушений уведомления звуком/речью.
События примитива элементарной фигуры **ElFigure***:*	
ws_Fig[n][Left|Right|Midle]	Активация фигуры (заливки) [n] клавишей мыши.
События примитива элементов формы **FormEl***:*	
ws_LnAccept	Установлено новое значение в строке ввода.
ws_TxtAccept	Изменено значение редактора текста.
ws_ChkChange	Состояние флажка изменено.
ws_BtPress	Кнопка нажата.
ws_BtRelease	Кнопка отпущена.
ws_BtToggleChange	Изменена вдавленность кнопки.
ws_CombChange	Изменено значение поля выбора.
ws_ListChange	Изменен текущий элемент списка.
ws_SliderChange	Изменение положения слайдера.
События примитива медиа-контента **Media***:*	
ws_MapAct{n}[Left|Right|Midle]	Активирована медиа-область с номером {n} клавишей мыши.

События являются основным механизмом уведомления и активно используются для осуществления взаимодействия с пользователем. Для обработки событий предусмотрены два механизма: сценарии управления открытием страниц и вычислительная процедура виджета.

Механизм "Сценарии управления открытием страниц" основан на базовом атрибуте виджета "evProc" и детально описан в разделе 3.2.

Механизм "Обработка событий с помощью вычислительной процедуры виджета" основан на атрибуте "event" и пользовательской процедуре вычисления на одном из языков пользовательского программирования OpenSCADA. События по мере поступления аккумулируются в атрибуте "event" до момента вызова вычислительной процедуры. Вычислительная процедура вызывается с указанной периодичностью вычисления виджета и получает значение атрибута "event" в виде списка событий. В процедуре вычисления пользователь может: проанализировать, обработать и исключить обработанные события из списка, а также добавить в список новые события. Оставшиеся после исполнения процедуры события, анализируются на предмет соответствия условиям вызова сценарием первого механизма после чего оставшиеся события передаются на верхний по иерархии виджет для обработки им, при этом осуществляется коррекция пути событий в соответствии с иерархией проникновения события.

Содержимое атрибута "event" является списком событий формата **<event>:<evSrc>**, с событием в отдельной строке. Приведём пример процедуры обработки событий на Java-подобном языке пользовательского программирования OpenSCADA:

```
using Special.FLibSYS;
ev_rez = "";
off = 0;
while(true)
{
      sval = strParse(event,0,"\n",off);
      if( sval == "" ) break;
      else if( sval == "ws_BtPress:/cvt_light" ) alarmSt = 0x1000001;
      else if( sval == "ws_BtPress:/cvt_alarm" ) alarmSt = 0x1000002;
      else if( sval == "ws_BtPress:/cvt_sound" ) alarmSt = 0x1000004;
      else ev_rez+=sval+"\n";
}
event=ev_rez;
```

3.5. Сигнализация

Важным элементом любого интерфейса визуализации является уведомление пользователя про нарушения - сигнализация. Для упрощения восприятия, а также в виду тесной связности визуализации и уведомления (как правило уведомление дополняет визуализацию) решено интегрировать интерфейс уведомления в интерфейс визуализации. Для этого во всех виджетах предусматриваются два дополнительных атрибута (уровня сеанса): "alarm" и "alarmSt". Атрибут "alarm" используется для формирования сигнала виджетом в соответствии с его логикой, а атрибут "alarmSt" используется для контроля за фактом сигнализации ветви дерева сеанса проекта.

Атрибут "alarm" является строкой и имеет следующий формат: *{lev|categ|message|type|tp_arg}* Где:

- *lev* - уровень сигнализации: число от 0 до 255;
- *categ* - категория сигнала: параметр подсистемы сбора, объект, путь или комбинация.
- *message* - сообщение сигнализации; для помещение в строку статуса, отображения в протоколе и помещения в архив сообщений;
- *type* - типы уведомления (визуальное, гудок и речь); формируется в виде целого числа, содержащего флаги способов уведомлений:
  - *0x01* - визуальная;
  - *0x02* - гудок, часто производится через PC-speaker;
  - *0x04* - звуковой сигнал из файла звука или синтез речи; если в <tp_arg> указано имя ресурса звукового файла, то воспроизводится именно он, иначе выполняется синтез речи из текста указанного в <message>.
- *tp_arg* - аргумент типа; используется в случае осуществления звуковой сигнализации для

указания ресурса звукового сигнала (файл звукового формата).

Атрибут "alarmSt" является целым числом, которое отражает максимальный уровень сигнала и факт квитирования ветви дерева сеанса проекта. Формат числа имеет следующий вид:

- первый байт (0-255) характеризует уровень сигнала ветви;
- второй байт указывает тип уведомления (также как и в атрибуте "alarm");
- третий байт указывает тип несквитированного уведомления (также как и в атрибуте "alarm");
- первый бит четвёртого байта имеет специальное назначение, установка этого бита является фактом квитирования уведомлений указанных первым байтом.

Формирование сигнала и получение его визуализатором.

Формирование сигнала производится самим виджетом путём установки собственного атрибута "alarm" нужным образом, и в соответствии с ним устанавливается атрибут "alarmSt" текущего и вышестоящих виджетов. Визуализаторы получают уведомление о сигнале с помощью стандартного механизма уведомления об изменении атрибутов виджетов.

Такой механизм предоставляет возможность формировать интерфейсы сигнализации как на уровне подсистемы "Сбор данных", так и прямо на уровне представления.

Учитывая то, что обработка условий сигнализации осуществляется в виджетах, страницы, содержащие объекты сигнализации, должны исполняться в фоне, не зависимо от открытости их в данный момент. Это осуществляется путём установки флага исполнения страницы в фоне.

Хотя механизм сигнализации и построен в среде визуализации, возможность формирования невизуальных элементов сигнализации остаётся, например, путём создания страницы, которая никогда не будет открываться.

Квитирование

Квитирование производится путём указания корня ветви виджетов и типов уведомления. Это позволяет реализовать квитирование на стороне визуализатора как по группам, например, по объектам сигнализации, так и индивидуально по объектам. При этом можно независимо квитировать разные типы сигнализаций. Установка квитирования производится простой модификацией атрибута "alarmSt".

Пример скрипта для работы с сигналами приведён ниже:

```
//Выделение факта наличия сигнализаций разных способов уведомления
cvt_light_en = alarmSt&0x100;
cvt_alarm_en = alarmSt&0x200;
cvt_sound_en = alarmSt&0x400;
//Выделение факта наличия несквитированных сигнализаций
//разных способов уведомления
cvt_light_active = alarmSt&0x10000;
cvt_alarm_active = alarmSt&0x20000;
cvt_sound_active = alarmSt&0x40000;
//Обработка событий кнопок квитирования и квитирование
//разных способов уведомлений
ev_rez = "";
off = 0;
while(true)
{
  sval = strParse(event,0,"\n",off);
  if( sval == "" ) break;
  else if( sval == "ws_BtPress:/cvt_light" ) alarmSt = 0x1000001;
  else if( sval == "ws_BtPress:/cvt_alarm" ) alarmSt = 0x1000002;
  else if( sval == "ws_BtPress:/cvt_sound" ) alarmSt = 0x1000004;
  else ev_rez+=sval+"\n";
}
event=ev_rez;
```

3.6. Управление правами

Для разделения доступа к интерфейсу ВУ и его составляющим каждый виджет содержит информацию о владельце, его группе и правах доступа. Права доступа записываются, как принято в системе OpenSCADA, в виде триады: *<пользователь><группа><остальные>*, где каждый элемент состоит из трёх признаков доступа. Для элементов СВУ принята следующая их интерпретация:

- *'r'* - право на просмотр виджета;
- *'w'* - право на контроль над виджетом.

В режиме разработки используется простая схема доступа "root.UI:RWRWR_", что означает - все пользователи могут открывать и просматривать библиотеки, их компоненты и проекты; а редактировать могут все пользователи группы "UI" (пользовательские интерфейсы).

В режиме исполнения работают права, описанные в компонентах интерфейса.

3.7. Связывание с динамикой

Для предоставления актуальных данных в интерфейс визуализации должны использоваться данные подсистемы "Сбор данных (DAQ)". Природа этих данных следующая:

1. параметры, содержащие некоторое количество атрибутов;
2. атрибуты параметра могут предоставлять данные четырёх типов: Логический, Целый, Вещественный и Строковый;
3. атрибуты параметра могут иметь историю (архив);
4. атрибуты параметра могут быть на чтение, запись и с полным доступом.

Учитывая первый пункт, нужно обеспечить возможность группового назначения ссылки. Для этого используем концепцию логического уровня.

В соответствии с пунктом 2 связи обеспечивают прозрачное преобразование типов и не требуют специальной конфигурации.

Для удовлетворения возможности доступа к архивам в соответствии с пунктом 3 связи выполняют проверку типа атрибута и, в случае подключения к "Адресу", в значение помещается адрес связи.

В терминах СВУ, динамические связи и конфигурация динамики являются одним процессом, для описания конфигурации которого предусматривается вкладка "Обработка" виджетов (рис.3.7.а). Вкладка содержит таблицу конфигурации свойств атрибутов виджета и текст процедуры вычисления виджета.

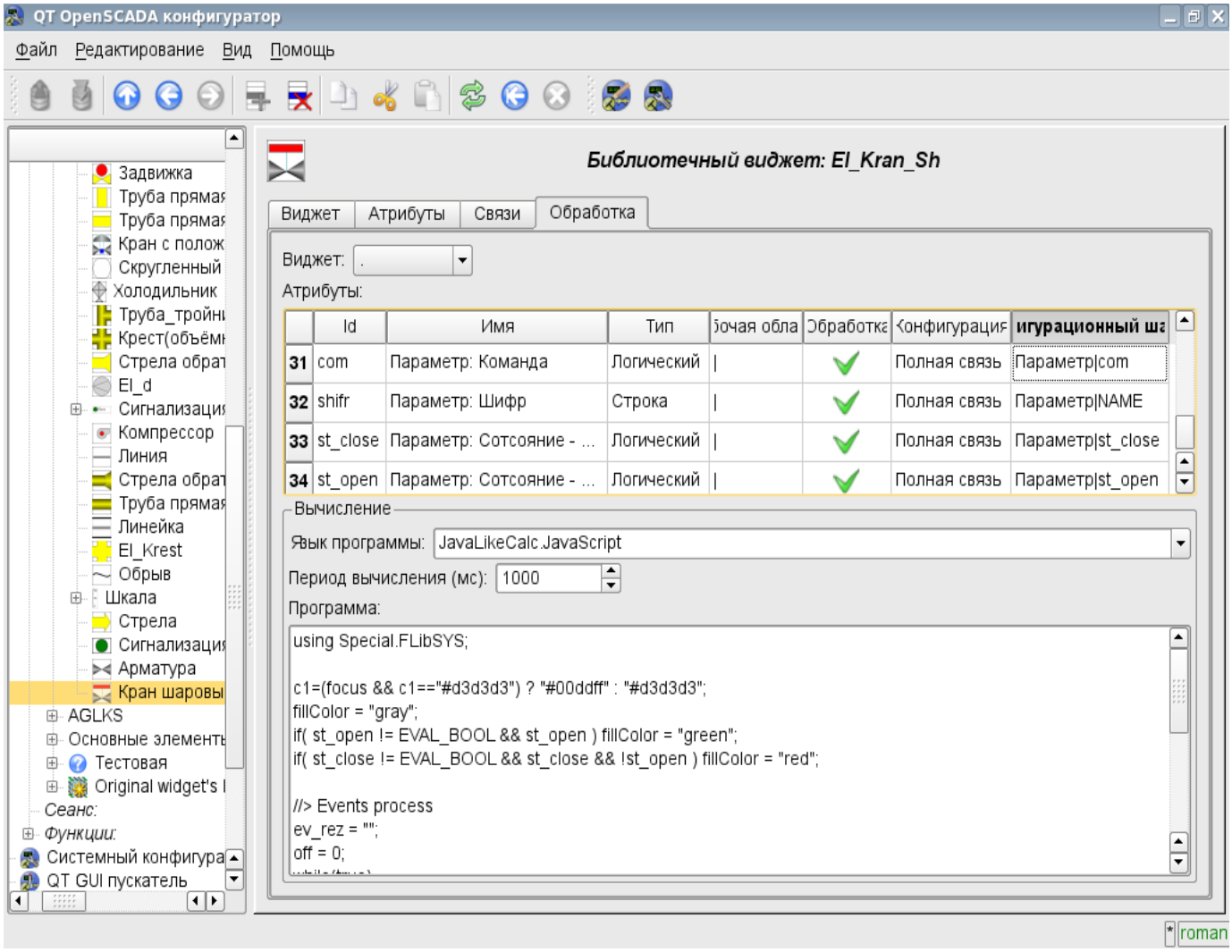


Рис. 3.7.а Вкладка "Обработка" страницы конфигурации виджета.

Кроме полей конфигурации атрибутов в таблице предусматривается колонка "Обработка", для избирательного использования атрибутов виджетов в вычислительной процедуре виджета, и колонки "Конфигурация" и "Конфигурационный шаблон" для описания конфигурации связей.

Если в колонке "Обработка" стоит true, то в вычислительной процедуре становится доступной переменная *{идентификатор виджета}_{идентификатор строки}*, например *cw_value*.

Колонка "Конфигурация" позволяет указать тип связи для атрибута виджета:

- *Постоянная* - во вкладке связей виджета появляется поле указания постоянной, например, особого цвета или заголовка для шаблонных кадров;
- *Входная связь* - связь с динамикой только для чтения;
- *Выходная связь* - связь с динамикой только для записи;
- *Полная связь* - полная связь с динамикой (чтение и запись).

Колонка "Конфигурационный шаблон" позволяет описать группы динамических атрибутов. Например, это могут быть разные типы параметров подсистемы "DAQ". Кроме того, при корректном формировании этого поля работает механизм автоматического назначения атрибутов при указании только параметра подсистемы "DAQ", что упрощает и ускоряет процесс конфигурации. Значение этой колонки имеет следующий формат: **<Параметр>|<Идентификатор>**, где:

- *<Параметр>* - группа атрибута;
- *<Идентификатор>* - идентификатор атрибута, именно это значение сопоставляется с атрибутами параметров DAQ при автоматическом связывании после указания групповой связи.

Установка связей может быть нескольких типов, который определяется префиксом:

- *val:* - Прямая загрузка значения через механизм связей. Например, связь: "val:100" загружает в атрибут виджета значение 100. Часто используется в случае отсутствия конечной точки связи с целью прямой установки значения.
- *prm:* - Связь на атрибут параметра или параметр в целом, для группы атрибутов, подсистемы "Сбор данных". Например, связь "prm:/LogicLev/experiment/Pi/var" осуществляет доступ атрибута виджета к атрибуту параметра подсистемы "Сбор данных".
- *wdg:* - Связь на атрибут другого виджета или виджет в целом для группы атрибутов. Например, связь "wdg:/ses_AGLKS/pg_so/pg_1/pg_ggraph/pg_1/a_bordColor" осуществляет доступ атрибута одного виджета к атрибуту другого. На данный момент этот тип связи не предназначен для установки пользователем вручную, а устанавливается автоматически в режиме динамического связывания!

Обработка связей происходит с периодичностью вычисления виджета в порядке:

- Получение данных входных связей.
- Выполнение вычисления скрипта.
- Передача значений по выходным связям.

На рис. 3.7.b представлена вкладка связей с групповым назначением атрибутов путём указания только параметра, а на рис. 3.7.c с индивидуальным назначением атрибутов.

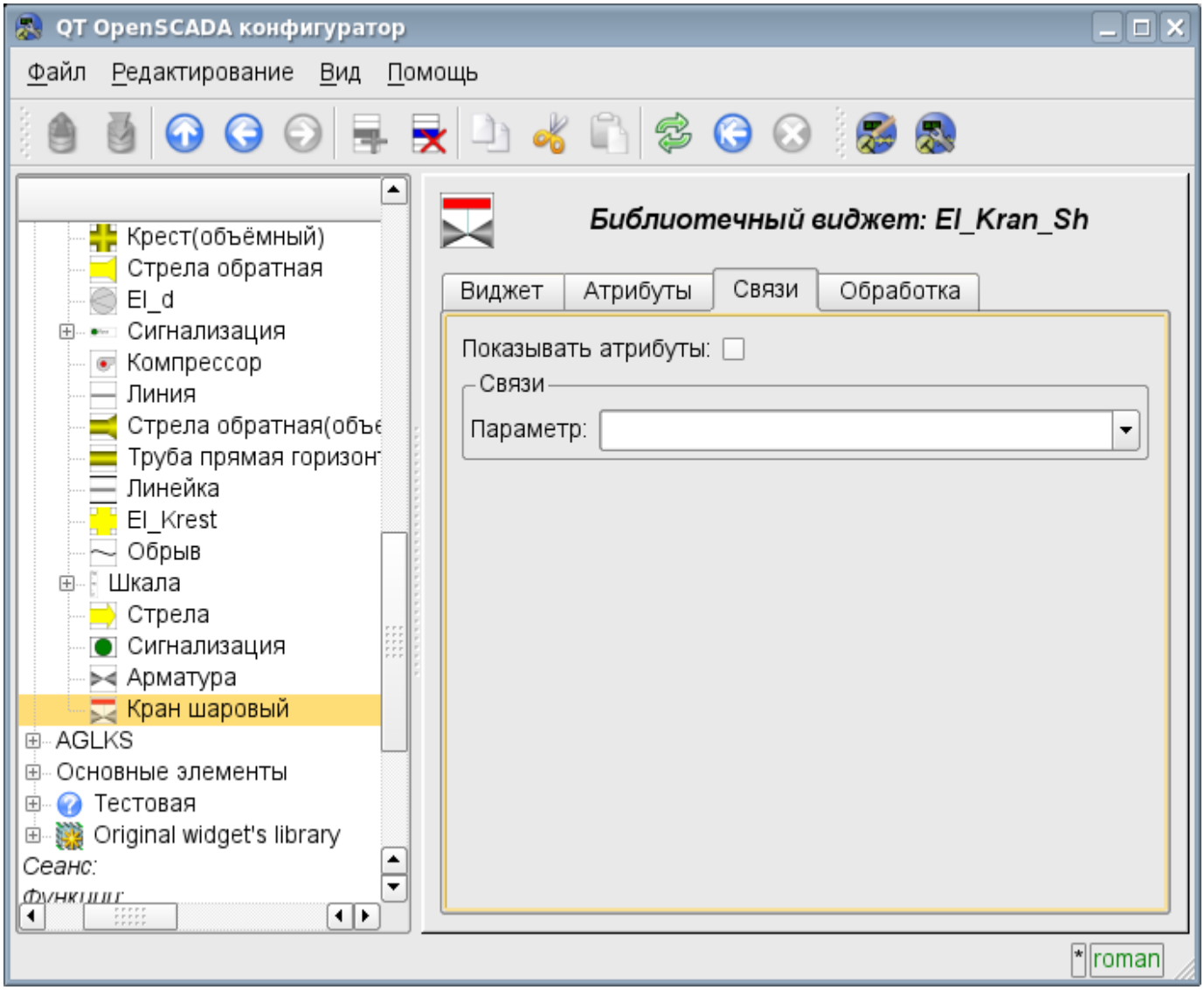


Рис. 3.7.b Вкладка "Связи" страницы конфигурации виджета с групповым назначением атрибутов путём указания только параметра.

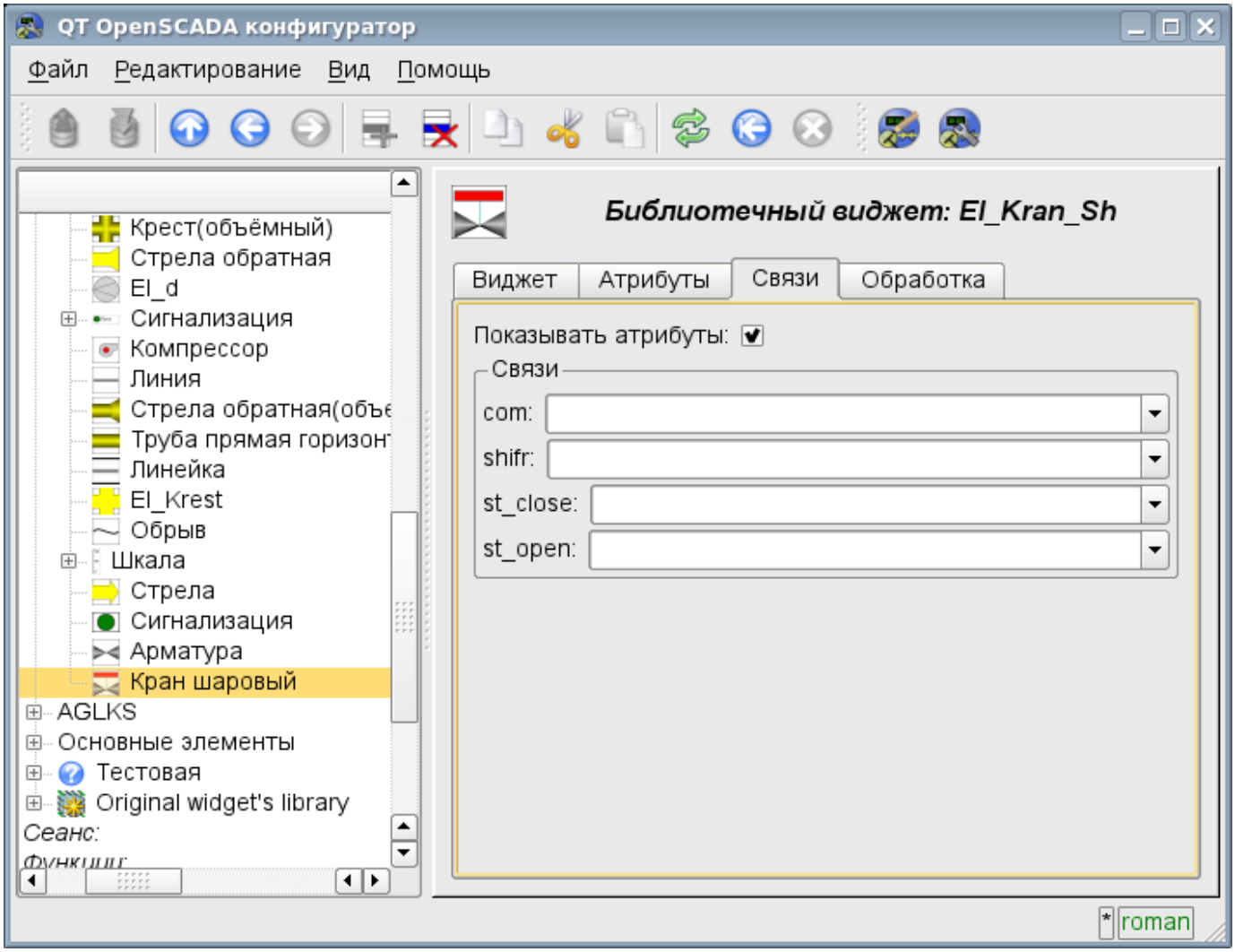


Рис. 3.7.c Вкладка "Связи" страницы конфигурации виджета с индивидуальным назначением атрибутов.

При размещении виджета, содержащего конфигурацию связей, в контейнер виджетов все связи исходного виджета добавляются в список результирующих связей контейнера виджетов (рис. 3.7.d)

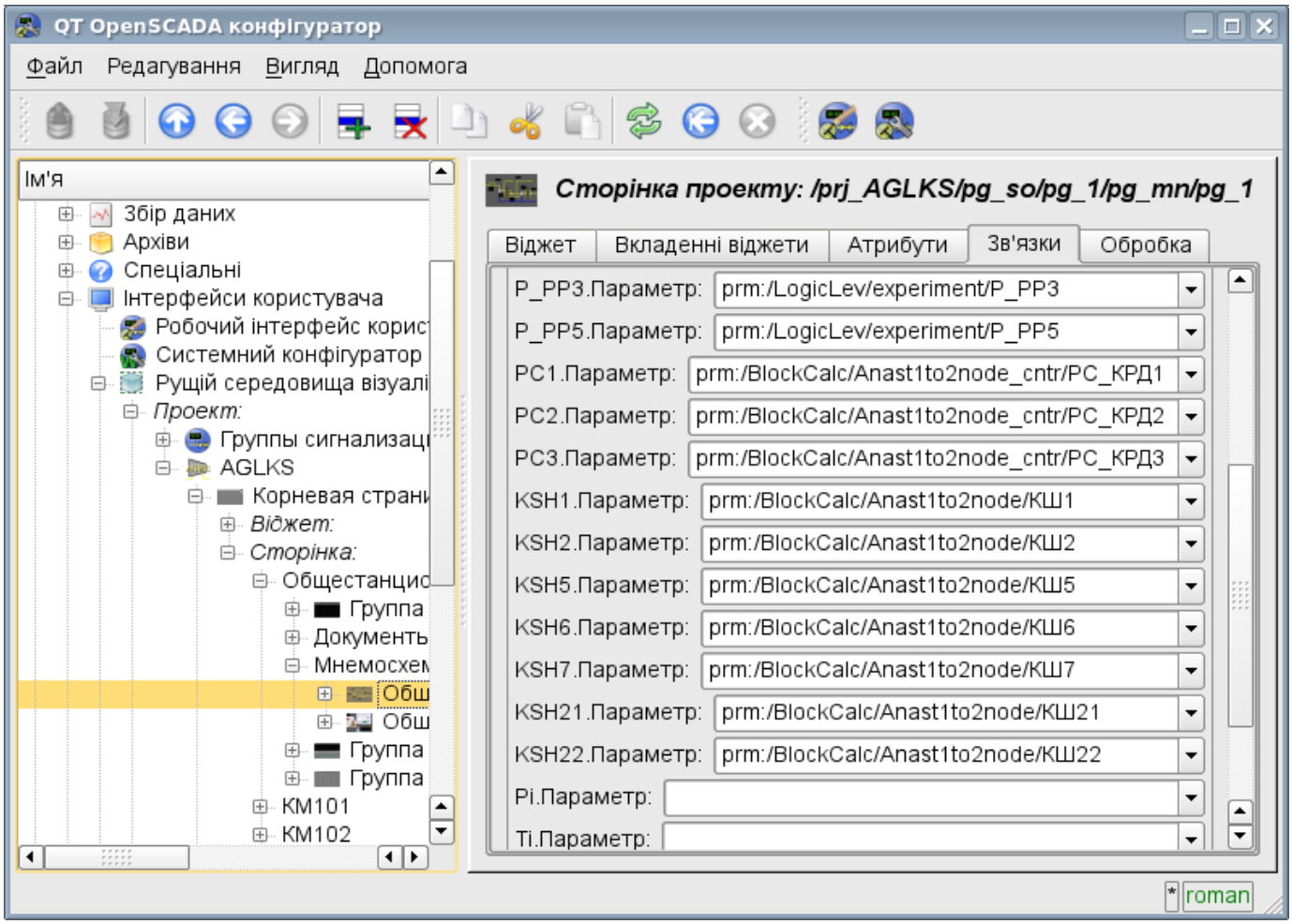


Рис. 3.7.d Вкладка "Связи" страницы конфигурации контейнера виджетов, включающего виджеты со связями.

Из вышесказанного видно, что связи устанавливаются пользователем в процессе конфигурации интерфейса. Однако, для предоставления возможности создания кадров общего назначения с функцией предоставления детализированных данных разных источников одного типа необходим механизм динамической установки связей. Такой механизм предусматривается посредством зарезервированного ключевого идентификатора "**<page>**" группы атрибутов связей у кадров общего назначения и динамическое назначение связей с идентификатором "**<page>**" в процессе открытия кадра общего назначения сигналом от другого виджета.

Рассмотрим пример, когда имется кадр общего назначения "Панель контроля графиком" и множество "Графиков" на разных кадрах. "Панель контроля графиком" имеет связи с шаблонами:

- tSek -> "<page>|tSek"
- tSize -> "<page>|tSize"
- trcPer -> "<page>|trcPer"
- valArch -> "<page>|valArch"

При этом каждый виджет "График" имеет атрибуты tSek, tSize, trcPer и valArch. В случае вызова сигнала открытия "Панели контроля графиком" из любого виджета "График" происходит связывания атрибутов "Панели контроля графиком" в соответствии атрибуту, указанному в шаблоне, с атрибутом виджета "График". Как результат, все изменения на "Панели контроля графиком" будут отражаться на графике посредством связи.

В случае наличия у виджета "График" внешних связей на параметры подсистемы "Сбор данных", связи "Панели контроля графиком" будут устанавливаться на внешний источник. Кроме этого, если у "Панели контроля графиком" будут заявлены связи на отсутствующие непосредственно у виджета "График" атрибуты, то будет производится поиск на наличие таких атрибутов у внешнего

источника, первого на который установлена прямая связь, выполняя, тем самым, дополнение недостающих связей.

Для наглядного изображения этого механизма приведена таблица 3.7.

Таблица 3.7. Механизм динамической линковки.

Атрибуты "Панели контроля графиком" (шаблон динамической связи)	**Атрибуты "Графика"**	**Атрибуты внешнего "Параметра"**	**Результирующая связь или значение связующегося атрибута**
tSek (<page>|tSek)	tSek	-	"График".tSek
tSize (<page>|tSize)	tSize	-	"График".tSize
trcPer (<page>|trcPer)	trcPer	-	"График".trcPer
valArch (<page>|valArch)	valArch	-	"График".valArch
var (<page>|var)	var	var	"Параметр".var
ed (<page>|ed)	-	ed	"Параметр".ed
max (<page>|max)	-	-	EVAL
min (<page>|min)	-	-	EVAL

3.8. Примитивы виджетов

Любой вновь создаваемый виджет основывается на одном из нескольких примитивов(конечный элемент визуализации) путём установки родственной связи как прямо на примитив, так и посредством нескольких промежуточных пользовательских виджетов. Каждый из примитивов содержит механизм (логику) модели данных. Экземпляр виджета хранит значения свойств конфигурирования примитива специально для себя.

В задачи интерфейса визуализации входит поддержка и работа с моделью данных примитивов виджетов. Примитивы виджетов должны быть тщательно проработаны и унифицированы с целью охватить как можно больше возможностей в как можно меньшем количестве слабо связанных друг с другом по назначению примитивов.

В таблице 3.8.а приведён перечень примитивов виджетов(базовых элементов отображения).

Таблица 3.8.а. Библиотека примитивов виджетов(базовых элементов отображения)

Id	Наименование	Функция
ElFigure	Элементарные графические фигуры	Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Предусматривается поддержка следующих элементарных фигур: • Линия. • Дуга. • Кривая безье. • Заливка замкнутого пространства. Для всех фигур, содержащихся в виджете устанавливаются единые свойства толщины, цвета и т.д., но это не исключает возможность указания вышеперечисленных атрибутов для каждой фигуры отдельно.
FormEl	Элементы формы.	Включает поддержку стандартных компонентов формы: • Редактирование строки. • Редактирование текста. • Флажок. • Кнопка. • Поле выбора из списка. • Список. • Слайдер. • Строка прокрутки.
Text	Текст	Элемент текста(метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием.
Media	Медиа	Элемент отображения растровых и векторных изображений различных форматов, проигрывания анимированных изображений, проигрывание аудио фрагментов и просмотр видео-фрагментов. Возможно в него стоит включить и поддержку OpenGL!
Diagram	Диаграмма	Элемент диаграммы с поддержкой возможности отображения нескольких потоков трендов, частотного спектра ... .
Protocol	Протокол	Элемент протокола, визуализатора системных сообщений с поддержкой несколько режимов работы.
Document	Документ	Элемент формирования отчётов, журналов и другой документации на основе доступных в системе данных.
Box	Контейнер	Содержит механизм размещения других виджетов с целью формирования новых более сложных виджетов и страниц конечной визуализации.

Id	Наименование	Функция
Function	Функция API объектной модели OpenSCADA	Невизуальный на стороне исполнения виджет, позволяющий включать вычислительные функции объектной модели OpenSCADA в СВУ.

Каждый примитив и виджет вообще содержит общий набор свойств/атрибутов в составе, приведенном в таблице 3.8.b:

Таблица 3.8.b. Общий набор свойств/атрибутов в виджете

Id	Имя	№	Значение
id	Id	-	Идентификатор элемента. Атрибут только для чтения, призванный предоставить информацию об идентификаторе элемента.
path	Path	-	Путь виджета. Атрибут только для чтения и предоставления информации об расположения элемента.
parent	Parent	-	Предок или родитель виджета. Атрибут только для чтения и предоставления информации об расположении предка, от которого наследован виджет.
owner	Owner	-	Владелец и группа виджета, через разделитель ":". По умолчанию "root:UI".
perm	Access	-	Доступ к виджету в виде выбора восьмеричных чисел вида "RWR_R_". По умолчанию 0664(RWRWR_).
root	Root	1	Идентификатор виджета-примитива (базового элемента), который лежит в основе образа визуализации виджета.
name	Name	-	Имя элемента. Модифицируемое имя элемента.
dscr	Description	-	Описание элемента. Текстовое поле для прикрепления к виджету краткого описания.
en	Enabled	5	Состояние элемента - "Включен". Отключенный элемент не отображается при исполнении.
active	Active	6	Состояние элемента - "Активный". Активный элементы может получать фокус при исполнении, а значит получать клавиатурные и иные события с последующей их обработкой.
geomX	Geometry:x	7	Геометрия, координата 'x' положения элемента.
geomY	Geometry:y	8	Геометрия, координата 'y' положения элемента.
geomW	Geometry:width	9	Геометрия, ширина элемента.
geomH	Geometry:height	10	Геометрия, высота элемента.
geomXsc	Geometry:x scale	13	Масштаб элемента по горизонтали.
geomYsc	Geometry:y scale	14	Масштаб элемента по вертикали.
geomZ	Geometry:z	11	Геометрия, координата 'z' (уровень) элемента на странице. Также определяет порядок передачи фокуса между активными элементами.
geomMargin	Geometry:margin	12	Геометрия, поля элемента.
tipTool	Tip:tool	15	Текст краткой помощи или подсказки по данному элементу. Обычно реализуется как всплывающая подсказка при удержании курсора мыши над элементом.
tipStatus	Tip:status	16	Текст информации о состоянии элемента или руководства к действию над элементом. Обычно реализуется в виде сообщения в строке статуса при удержании курсора мыши над элементом.

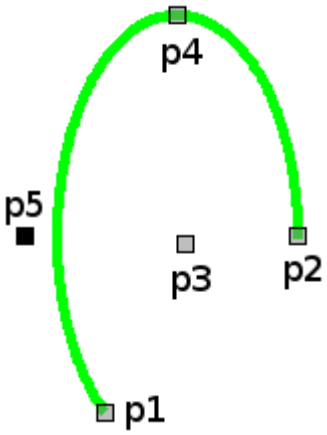
Id	Имя	№	Значение
contextMenu	Context menu	17	Конфигурация собственного контекстного меню элемента. Конфигурация записывается в виде строк пунктов контекстного меню формата: **<Имя пункта>:<ИдСобытия>**. Где: • *<Имя пункта>* - Имя пункта меню. • *<ИдСобытия>* - Идентификатор события, генерируемого виджету (usr_<ИдСобытия>) при выборе пункта меню.
evProc	Events process	-	Атрибут для хранения сценария обработки событий непосредственного управления пользовательским интерфейсом. Сценарий представляет собой список команд интерфейсу визуализации, генерируемых при поступлении события (атрибут **event**).
Дополнительные атрибуты для элементов, помещённых в проект в роли страницы.			
pgOpen	Page:open state	-	Признак "Страница открыта".
pgNoOpenProc	Page:no open process	-	Признак "Исполнять страницу даже если она закрыта".
pgOpenSrc	Page:open source	3	Полный адрес страницы, открывшей данную.
pgGrp	Page:group	4	Группа страницы.
Дополнительные атрибуты режима исполнения.			
event	Event	-	Специальный атрибут для сбора событий виджета в списке, разделённом новой строкой. Данный атрибут доступен только в сеансе. Доступ к атрибуту защищён ресурсом с целью избежания потери событий. Атрибут всегда доступен в скрипте виджета.
load	Load	-1	Виртуальная команда групповой загрузки данных.
focus	Focus	-2	Специальный атрибут индикации факта получения фокуса активным виджетом. Данный атрибут доступен только в сеансе. Атрибут этого виджета и вложенных виджетов доступен в скрипте виджета.
perm	Permition	-3	Виртуальный атрибут проверки прав активного пользователя на просмотр и контроль над виджетом.

3.8.1. Элементарные графические фигуры (ElFigure)

Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Учитывая широкий спектр всевозможных фигур, которые должен поддерживать примитив, и в тоже время являться достаточно простым в использовании и, по возможности, в реализации, решено было ограничить перечень базовых фигур, используемых для построения результирующих графических объектов до таких фигур: линия, дуга, кривая Безье и заливка замкнутых контуров. Основываясь уже на этих базовых фигурах, можно строить производные фигуры, комбинируя базовые. В рамках примитива существует возможность задания прозрачности цвета в диапозоне [0..255], где '0' - полная прозрачность.

Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.1.

Таблица 3.8.1. Набор дополнительных свойств/атрибутов в примитиве ElFigure

Id	Имя	№	Значение
lineWdth	Line:width	20	Ширина линии.
lineClr	Line:color	21	Цвет линии.(Прозрачность цвета задается следующим способом: red-127 | #ff0000-127, где "127" и есть значение прозрачности.)
lineStyle	Line:style	22	Стиль линии (сплошная, пунктир, точечная).
bordWdth	Border:width	23	Ширина бордюра линии. Нулевая ширина указывает на отсутствие бордюра.
bordClr	Border:color	24	Цвет бордюра.
fillColor	Fill:color	25	Цвет заливки.
fillImg	Fill:image	26	Изображение заливки.
orient	Orientation angle	28	Угол поворота содержимого виджета.
elLst	Element's list	27	Список графических примитивов в формате: • Линия. Форма записи в списке: *<line:p1 | (x | y):p2 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[line_style | s{n}]>* • Дуга. Форма записи в списке: *<arc:p1 | (x | y):p2 | (x | y):p3 | (x | y):p4 | (x | y):p5 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[*line_style | *s{n}]>*  • Кривая Безье. Форма записи в списке: *<bezier:p1 | (x | y):p2 | (x | y):p3 | (x | y):p4 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[line_style | s{n}]>* • Заливка. Форма записи в списке: *<fill:p1 | (x | y),p2 | (x | y),...,pn | (x | y):[fillClr | c{n}]:[fillImg | i{n}]>*

Id	Имя	№	Значение
Атрибуты для каждой точки из списка графических фигур **elLst**			
p{n}x	Point {n}:x	30+n*6	Координата 'x' точки {n}.
p{n}y	Point {n}:y	30+n*6+1	Координата 'y' точки {n}.
w{n}	Width {n}	30+n*6+2	Ширина {n}.
c{n}	Color {n}	30+n*6+3	Цвет {n}.
i{n}	Image {n}	30+n*6+4	Изображение {n}.
s{n}	Style {n}	30+n*6+5	Стиль {n}.

3.8.2. Элементы формы (FormEl)

Примитив, предназначенный для предоставления стандартных элементов формы в распоряжение пользователя. Общий перечень атрибутов зависит от типа элемента. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.2.

Таблица 3.8.2. Набор дополнительных свойств/атрибутов в примитиве FormEl

Id	Имя	№	Значение
elType	Element type	20	Тип элемента (Строка редактирования; Редактор текста; Флажок; Кнопка; Выбор из списка; Список; Слайдер; Полоса прокрутки). От его значения зависит перечень дополнительных атрибутов.
Строка редактирования:			
value	Value	21	Содержимое строки.
view	View	22	Вид строки редактирования (Текст; Комбобокс; Целое; Вещественное; Время; Дата; Дата и время).
cfg	Config	23	Конфигурация строки. Формат значения данного поля для различных видов строки: *Текст* - указывается шаблон ввода в формате библиотеки QT. *Комбобокс* - содержит список значений редактируемого комбобокса. *Целое* - содержит конфигурацию поля ввода целочисленного представления в формате: **<Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>**. *Вещественное* - содержит конфигурацию поля ввода вещественного представления в формате: **<Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>:<Число знаков после запятой>**. *Время*, *Дата*, *Дата и время* - формировать дату по шаблону с параметрами: **d** - номер дня (1-31); **dd** - номер дня (01-31); **ddd** - сокращённое наименование дня ("Mon" ... "Sun"); **dddd** - полное наименование дня ("Monday" ... "Sunday"); **M** - номер месяца (1-12); **MM** - номер месяца (01-12); **MMM** - сокращённое имя месяца ("Jan" ... "Dec"); **MMMM** - полное имя месяца ("January" ... "December"); **yy** - последние две цифры года; **yyyy** - год полностью; **h** - час (0-23); **hh** - час (00-23); **m** - минуты (0-59); **mm** - минуты (00-59); **s** - секунды (0-59); **ss** - секунды (00-59); **AP,ap** - отображать AM/PM или am/pm.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Редактор текста:			
value	Value	21	Содержимое редактора.
wordWrap	Word wrap	22	Автоматический перенос текста по словам.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.

Id	Имя	№	Значение
Флажок:			
name	Name	26	Имя/метка флажка.
value	Value	21	Значение флажка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Кнопка:			
name	Name	26	Имя, надпись на кнопке.
value	Value	21	Значение для фиксированной кнопки.
img	Image	22	Изображение на кнопке.
color	Color	23	Цвет кнопки.
colorText	Color:text	27	Цвет текста.
checkable	Checkable	24	Признак функционирования как фиксированная кнопка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Выбор из списка:			
value	Value	21	Текущее значение списка.
items	Items	22	Перечень элементов списка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Список:			
value	Value	21	Выбранное значение списка.
items	Items	22	Перечень элементов списка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Слайдер и полоса прокрутки:			
value	Value	21	Положение слайдера.
cfg	Config	22	Конфигурация слайдера в формате: **<VertOrient>:<Min>:<Max>:<SinglStep>:<PageStep>**. Где: • *VertOrient* - признак вертикальной ориентации, по умолчанию ориентация горизонтальная; • *Min* - минимальное значение; • *Max* - максимальное значение; • *SinglStep* - размер одиночного шага; • *PageStep* - размер страничного шага.

3.8.3. Элемент текста (Text)

Данный примитив предназначен для вывода простого текста, используемого в роли меток и различных подписей. С целью простого создания частых декоративных оформлений примитив должен поддерживать обвод текста рамкой. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.3.

Таблица 3.8.3. Набор дополнительных свойств/атрибутов в примитиве Text

Id	Имя	№	Значение
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
color	Color	26	Цвет текста.
orient	Orientation angle	27	Ориентация текста, поворот на угол.
wordWrap	Word wrap	28	Автоматический перенос текста по словам.
alignment	Alignment	29	Выравнивание текста (Вверху слева; Вверху справа; Вверху по центру; Вверху по ширине; Внизу слева; Внизу справа; Внизу по центру; Внизу по ширине; По центру слева; По центру справа; По середине; По центру по ширине).
text	Text	30	Значение текстового поля.
numbArg	Arguments number	40	Количество аргументов.
Атрибуты аргументов			
arg{x}val	Argument {x}:value	50+10*x	Значение аргумента
arg{x}tp	Argument {x}:type	50+10*x+1	Тип аргумента: "Integer", "Real", "String"
arg{x}cfg	Argument {x}:config	50+10*x+2	Конфигурация аргумента: • *строка* : **[len]** - ширина строки; • *вещественное*: **[wdth];[form];[prec]** - ширина значения, форма значения ('g','f'); • *целое*: **[len]** - ширина значения.

3.8.4. Элемент отображения медиа-материалов (Media)

Данный примитив предназначен для проигрывания различных медиа-материалов, начиная от простых изображений и заканчивая полноценными аудио и видео потоками. Учитывая многообразность способов и библиотек проигрывания полноценных аудио и видео потоков, а также достаточно серьёзную трудоёмкость по имплиминтации всех этих механизмов в данный виджет, решено было на первоначальном этапе реализовать только работу с изображениями и простыми анимационными форматами изображений и видео. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.4.

Таблица 3.8.4. Набор дополнительных свойств/атрибутов в примитиве Media

Id	Имя	№	Значение
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).
src	Source	25	Источник медиа-данных.
fit	Fit to widget size	26	Признак "Согласовать содержимое с размером виджета".
type	Type	27	Тип медиа (Image;Movie).
areas	Map areas	28	Количество активных областей.
Атрибуты видеоролика (Movie)			
speed	Play speed	29	Скорость проигрывания, в процентах от оригинальной скорости. Если значение меньше 1%, то проигрывание прекращается.
Активные области			
area{x}shp	Area {x}:shape	40+3*x	Вид области (Rect;Poly;Circle).
area{x}coord	Area {x}:coordinates	40+3*x+1	Координаты областей. Через запятую идут координаты: "x1,y1,x2,y2,xN,yN"
area{x}title	Area {x}:title	40+3*x+2	Заголовок области.

3.8.5. Элемент построения диаграмм/трендов (Diagram)

Данный примитив предназначен для построения различных диаграмм, включая и графики/тренды отображения текущего процесса и архивных данных. На данный момент реализованы следующие типы диаграмм:

- "График" - позволяет строить одномерные графики из значений параметров подсистемы "Сбор данных" по времени, а также прямое использование архивных данных для построения графиков. Поддерживается режим отслеживания как текущих значений, так и значений по архиву. Поддерживается также возможность построения графиков параметров, не имеющих архива значений.
- "Спектр" - строит частотный спектр из значений параметров подсистемы "Сбор данных". Окно данных частотного спектра формируется, исходя из размера виджета по горизонтали в пикселах и доступных данных параметров, наложенных на сетку горизонтального размера. В связи с этим минимальная частота определяется значением атрибута tSize (1/tSize), а максимальная частота выделяемых частот определяется половинной шириной графика в пикселах умноженной на минимальную частоту (width/(2*tSize)). Поддерживается возможность формирования спектра в режиме слежения.

Процесс доступа к архивным данным оптимизирован путём ведения промежуточного буфера для отображения, а также упаковки трафика данных при запросе. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.5.

Таблица 3.8.5. Набор дополнительных свойств/атрибутов в примитиве Diagram

Id	Имя	№	Значение
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).
trcPer	Tracing period (s)	25	Режим и периодичность слежения.
type	Type	26	Тип диаграммы: "Trend", "Спектр".
Атрибуты тренда/графика (Trend)			
tSek	Time:sek	27	Текущее время, секунд.
tUSek	Time:usek	28	Текущее время, микросекунды.
tSize	Size, sek	29	Размер тренда, секунды.
curSek	Cursor:sek	30	Положение курсора, секунды.
curUSek	Cursor:usek	31	Положение курсора, микросекунды.
curColor	Cursor:color	32	Цвет курсора.
sclColor	Scale:color	33	Цвет шкалы/решетки.
sclHor	Scale:horizontal	34	Режим горизонтальной шкалы/решетки: "No draw", "Grid;Markers" и "Grid and markers".
sclVer	Scale:vertical	35	Режим вертикальной шкалы/решетки: "No draw", "Grid", "Markers", "Grid and markers", "Grid (log)", "Marker (log)", "Grid and markers (log)".
sclVerScl	Scale:vertical scale (%)	40	Вертикальный масштаб графика в процентах.
sclVerSclOff	Scale:vertical scale offset (%)	41	Смещение вертикального маштаба в процентах.
sclMarkColor	Scale:Markers:color	36	Цвет маркеров шкалы/решетки.

Id	Имя	№	Значение
sclMarkFont	Scale:Markers:font	37	Шрифт маркеров шкалы/решетки в виде {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
valArch	Value archivator	38	Архиватор архивов параметров.
parNum	Parameters number	39	Количество параметров, отображаемых на одном тренде.
Индивидуальные атрибуты параметров тренда/графика			
prm{X}addr	Parametr {X} :address	50+10*{X}	Полный адрес к параметру {X} или архиву значений.
prm{X}bordL	Parametr {X} :view border:lower	50+10*{X} +1	Нижняя граница значений параметра {X}.
prm{X}bordU	Parametr {X} :view border:upper	50+10*{X} +2	Верхняя граница значений параметра {X}.
prm{X}color	Parametr {X} :color	50+10*{X} +3	Цвет отображения тренда параметра {X}.
prm{X}val	Parametr {X} :value	50+10*{X} +4	Значение параметра {X} под курсором.

3.8.6. Элемент построения протоколов, на основе архивов сообщений (Protocol)

Данный примитив предназначен для визуализации данных архива сообщений путём формирования протоколов с различными способами визуализации, начиная от статического сканирующего просмотра и заканчивая динамическим отслеживанием протокола сообщения. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.6.

Таблица 3.8.6. Набор дополнительных свойств/атрибутов в примитиве Protocol

Id	Имя	№	Значение
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
font	Font	22	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
headVis	Header visible	23	Заголовок таблицы видим или нет.
time	Time, sek	24	Текущее время, секунд.
tSize	Size, sek	25	Размер запроса, секунды.
trcPer	Tracing period (s)	26	Режим и периодичность слежения.
arch	Archival	27	Архиватор архива сообщений.
tmpl	Template	28	Шаблон запроса в архиве.
lev	Level	29	Уровень сообщений.
viewOrd	View order	30	Порядок отображения ("По времени", "По уровню", "По категории", "По сообщению", "По времени (обратно)", "По уровню (обратно)", "По категории (обратно)", "По сообщению (обратно)").
col	View columns	31	Отображаемые колонки.
itProp	Item's properties	32	Количество свойств элементов.
Индивидуальные атрибуты свойств элементов			
it{X}lev	Item {X}:level	40+5*{X}	Критерий: уровень элемента {X}. Более или равно указанному.
it{X}tmpl	Item {X}:template	41+5*{X}	Критерий: шаблон категории элемента {X}. Включая специальные символы '*' и '?'.
it{X}fnt	Item {X}:font	42+5*{X}	Шрифт элемента {X}.
it{X}color	Item {X}:color	43+5*{X}	Цвет элемента {X}.

3.8.7. Элемент формирования отчётной документации (Document)

Примитив предназначен для формирования отчётной, оперативной и иной документации на основе шаблонов документов. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.7.

Таблица 3.8.7. Набор дополнительных свойств/атрибутов в примитиве Document

Id	Имя	№	Значение
style	CSS	20	Стиль документа (CSS).
tmpl	Template	21	XHTML исходный шаблон документа.
doc	Document	22	Псевдо-виртуальный атрибут текущего (выбранного) документа.
font	Font	26	Базовый шрифт текста документа в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
bTime	Time:begin	24	Время начала документа, секунд.
time	Time:current	23	Время генерации документа, секунд.
n	Archive size	25	Количество документов или глубина архива.
Атрибуты включеного режима архивирования			
aCur	Cursor:archive	-	Позиция текущего документа в архиве. Запись значения <0 производит архивацию текущего документа.
vCur	Cursor:view	-	Текущий визуализируемый документ архива. Запись значения -1 - выбор следующего документа, -2 - выбор предыдущего документа.
Атрибуты архива			
doc{X}	Document {X}	-	Архивный документ X (0...(n-1))

Возможности примитива "Документ":

- Гибкое формирования структуры документа на основе языка гипертекстовой разметки. Это предоставит поддержку широких возможностей форматирования документов с последующей реализацией обёртки графического интерфейса формирования документа.
- Формирования документов по команде или по плану в архив с последующим просмотром архива.
- Формирование документа в режиме реального времени полностью динамически и на основе архивов за указанное время.
- Использование атрибутов виджета для передачи значений и адресов на архивы в документ. Позволяет использовать виджет документа как шаблон для формирования отчётов с другими входными данными.

В основе любого документа лежит XHTML-шаблон. XHTML-шаблон это тег "body", WEB-страницы, содержащий статику документа в стандарте XHTML 1.0 и элементы исполняемых инструкций на одном из языков пользовательского программирования OpenSCADA в виде **<?dp <procedure> ?>**. Результирующий документ формируется путём исполнения процедур и вставки их результата в документ.

Источником значений исполняемых инструкций являются атрибуты виджета этого примитива, а также все механизмы языков пользовательского программирования OpenSCADA. Атрибуты могут добавляться пользователем и линковаться на реальные атрибуты параметров или-же являться автономными, значения которых будут формироваться в скрипте виджета. В случае со слинкованными атрибутами могут извлекаться значения из истории, архива.

На рис. 3.8.7.а изображена структурная схема виджета примитива "Документ". Согласно этой структуре "Документ" содержит: XHTML-шаблон, результирующие документы и скрипт обработки данных. Источником данных для скрипта и результирующих документов являются атрибуты виджета.

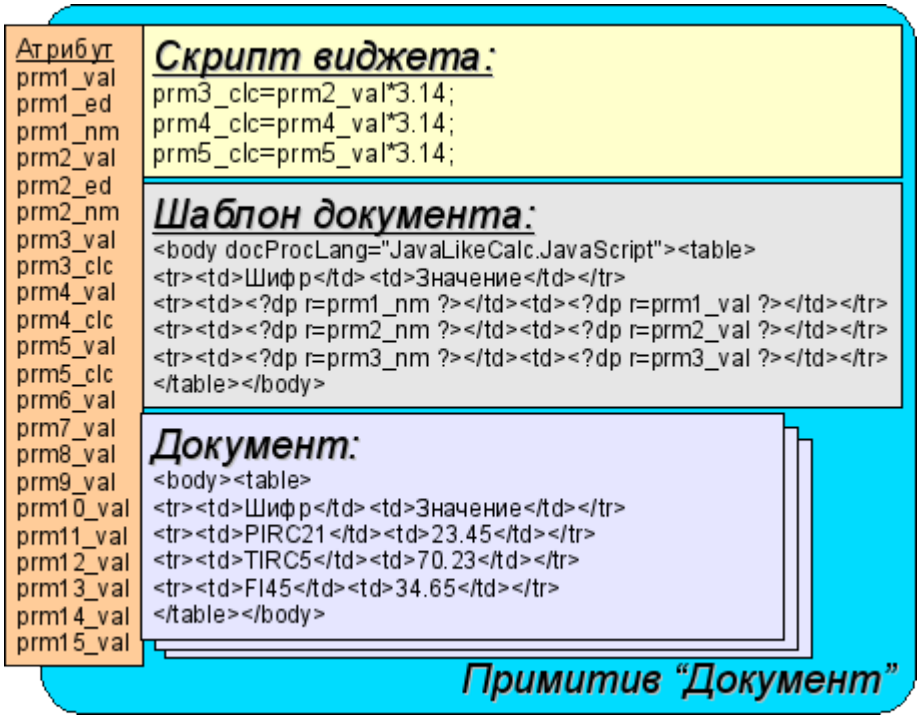


Рис.3.8.7.а Структурная схема примитива "Документ".

Предусматривается работа виджета в двух режимах: Динамический и Архивный. Отличие архивного режима заключается в наличии архива указанной глубины и атрибутов позволяющих управлять процессом архивирования и просмотра указанного документа в архиве.

Генерация документа всегда производится в момент установки атрибута времени <time> относительно установленного начального времени документа в атрибуте <bTime>. При выключенном архиве результирующий документ помещается непосредственно в атрибут <doc>. При включенном архиве результирующий документ помещается в ячейку под курсором, атрибут <aCur>, а так-же в <doc> если значение курсора архива <aCur> и курсора визуализируемого документа <vCur> совпадают. Атрибуты архивных курсоров предусматривают несколько командных значений:

- *aCur<0* - Перемещает курсор архиватора на следующую позицию, тем самым оставляя предыдущий документ в архиве и очищая документ под курсором.
- *vCur==-1* - Выбор следующего документа для отображений. Выбранный документ копируется в атрибут <doc>.
- *vCur==-2* - Выбор предыдущего документа для отображений. Выбранный документ копируется в атрибут <doc>.

Как было указано выше динамика шаблона документа определяется вставками исполняемых инструкций вида **<?dp <procedure> ?>**. В процедурах могут использоваться одноимённые атрибуты виджета и функции пользовательского интерфейса программирования OpenSCADA. Кроме атрибутов виджета зарезервированы специальные атрибуты (табл. 3.8.7.а).

Кроме специальных атрибутов в XHTML шаблоне зарезервированы теги и атрибуты тегов специального назначения (табл. 3.8.7.а).

Таблица 3.8.7.а. Специальные и зарезервированные элементы шаблона.

Имя	Назначение
Атрибуты	
rez	Атрибут результата исполнения процедуры, содержимое которого помещается в дерево документа.
lTime	Время последнего формирования. Если документ формируется впервые то <lTime> равен <bTime>.
rTime	Содержит время для перебираемых значений в секундах. Определяется внутри тегов с атрибутом "docRept".

Имя	Назначение
rTimeU	Содержит время для перебираемых значений в микросекундах. Определяется внутри тегов с атрибутом "docRept".
rPer	Содержит периодичность перебора значений (атрибут "docRept").
mTime, mTimeU, mLev, mCat, mVal	Определяются внутри тегов с атрибутом "docAMess" при разборе сообщений архива сообщений: mTime - время сообщения; mTimeU - время сообщения, микросекунды; mLev - уровень сообщения; mCat - категория сообщения; mVal - значение сообщения.
Специальные теги	
Специальные атрибуты стандартных тегов	
body.docProcLang	Язык исполняемых процедур документа. По умолчанию это JavaLikeCalc.JavaScript.
*.docRept="1s"	Тег с указанным атрибутом при формировании размножается путём смещения времени в атрибуте "rTime" на значение указанное в данном атрибуте.
.docAMess="1:PLC"	Указывает на необходимость размножения тега с атрибутом сообщения из архива сообщений за указанный интервал времени и в соответствии с уровнем (1) и шаблоном запроса (PLC*). Для данного тега, в процессе размножения, определяются атрибуты: mTime, mTimeU, mLev, mCat и mVal
*.docRevers="1"	Указывает на инвертирование порядка размножения, последний сверху.
*.docAppend="1"	Признак необходимости добавления результата выполнения процедуры в тег процедуры. Иначе результат исполнеия заменяет содержимое тега.
body.docTime	Время формирования документа. Используется для установки атрибута <lTime> при следующем формировании документа. Не устанавливается пользователем!

3.8.8. Контейнер (Box)

Примитив контейнера используется для формирования составных виджетов и/или страниц пользовательского интерфейса. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.8.

Таблица 3.8.8. Набор дополнительных свойств/атрибутов в примитиве Box

Id	Имя	№	Значение
pgOpenSrc	Page:open source	3	Полный адрес страницы, которая включена внутрь данного контейнера.
pgGrp	Page:group	4	Группа контейнера страниц.
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).

3.9. Использование БД для хранения библиотек виджетов и проектов

Хранение данных виджетов и библиотек виджетов реализовано в БД, доступных системе OpenSCADA. БД организована по принадлежности данных к библиотеке. Т.е. отдельная библиотека хранится в отдельной группе таблиц одной или разных БД. Перечень библиотек виджетов хранится в индексной таблице библиотек с именем "VCALibs" и структурой “Libs”. Экземпляр этой таблицы создаётся в каждой БД, где хранятся данные этого модуля с перечнем библиотек, содержащихся в конкретно взятой БД. В состав таблиц, принадлежащих библиотеке виджетов, входят следующие:

- {DB_TBL} — Таблица с виджетами, принадлежащими библиотеке (структура "LibWigets").
- {DB_TBL}_io — Таблица с рабочими свойствами виджетов этой библиотеки и вложенными виджетами контейнерных виджетов (структура "LibWidgetIO").
- {DB_TBL}_uio — Таблица с пользовательскими свойствами виджетов этой библиотеки и вложенными виджетами контейнерных виджетов (структура "LibWidgetUserIO", раздела БД).
- {DB_TBL}_incl — Таблица с перечнем вложенных виджетов в виджеты-контейнеры данной библиотеки (структура "LibWidgetIncl").
- {DB_TBL}_mime — Таблица с ресурсами библиотеки и её виджетов (структура "LibWidgetMime").
- {DB_TBL}_ses — Таблица для хранения данных режима исполнения проектов, т.е. сеансов (структура "PrjSesIO").

Проекции (структуры) основных таблиц таковы :

- Libs(ID, NAME, DSCR, DB_TBL, ICO) - Библиотеки виджетов <*ID*>.
  ID - идентификатор;
  NAME - имя;
  DSCR - описание;
  DB_TBL - БД с виджетами;
  ICO - закодированное (Base64) изображение иконки библиотеки.
- LibWigets(ID, ICO, PARENT, PROC, PROC_PER, USER, GRP, PERMIT, ATTRS, DBV) - Виджеты <*ID*> библиотеки.
  ID - идентификатор;
  ICO - закодированное (Base64) изображение иконки виджета;
  PARENT - адрес виджета основы в виде */wlb_originals/wdg_Box* ;
  PROC - внутренний сценарий и язык сценария виджета;
  PROC_PER - периодичность вычисления сценария виджета;
  ATTRS - перечень атрибутов виджета, модифицированных пользователем;
  DBV - версия БД.
- LibWidgetIO(IDW, ID, IDC, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) - Рабочие атрибуты <*ID*> виджета <*IDW*>.
  IDW - идентификатор виджета;
  IDC - идентификатор дочернего виджета;
  ID - идентификатор IO;
  IO_VAL - значение атрибута;
  SELF_FLG - внутренние флаги IO;
  CFG_TMPL - шаблон элемента конфигурации, основанного на данном атрибуте;
  CFG_VAL - значение элемента конфигурации (ссылка, константа ...).
- LibWidgetUserIO(IDW, ID, IDC, NAME, IO_TP, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) - Пользовательские атрибуты <*ID*> виджета <*IDW*>.
  IDW - идентификатор виджета;
  IDC - идентификатор дочернего виджета;
  ID - идентификатор IO;
  NAME - имя IO;
  IO_TP - тип и главные флаги IO;
  IO_VAL - значение IO;
  SELF_FLG - внутренние флаги IO;

CFG_TMPL - шаблон элемента конфигурации, основанного на данном атрибуте;
CFG_VAL - значение элемента конфигурации (ссылка, константа ...).

- LibWidgetIncl(IDW, ID, PARENT, ATTRS, USER, GRP, PERMIT, DBV) - Включенные в контейнер <*IDW*> виджеты <*ID*>.

IDW - идентификатор виджета;
ID - идентификатор экземпляра вложенного виджета;
PARENT - адрес виджета основы в виде */wlb_originals/wdg_Box* ;
ATTRS - перечень атрибутов виджета, модифицированных пользователем;
DBV - версия БД.

- LibWidgetMime(ID, MIME, DATA) - Audio, video, media и другие ресурсы виджетов библиотеки.

ID - Идентификатор ресурса.
MIME - Mime тип данных ресурса (в формате - <mimeType;Size>).
DATA - Данные ресурса кодированные Base64.

- Project(ID, NAME, DSCR, DB_TBL, ICO, USER, GRP, PERMIT, PER, FLGS) - Проекты интерфейсов визуализации <*ID*>.

ID - идентификатор проекта;
NAME - имя проекта;
DSCR - описание проекта;
DB_TBL - БД со страницами проекта.
ICO - закодированное (Base64) изображение иконки проекта;
USER - владелец проекта;
GRP - группа пользователей проекта;
PERMIT - права доступа к проекту;
PER - периодичность вычисления проекта;
FLGS - флаги проекта.

- ProjPage(OWNER, ID, ICO, PARENT, PROC, PROC_PER, USER, GRP, PERMIT, FLGS, ATTRS, DBV) - Страницы <*ID*> содержащиеся в проекте/странице <*OWNER*>.

OWNER - проект/страница - владелец данной страницы (в виде - "/AGLKS/so/1/gcadr")
ID - идентификатор страницы;
ICO - закодированное (Base64) изображение иконки страницы;
PARENT - адрес виджета основы страницы в виде */wlb_originals/wdg_Box* ;
PROC - внутренний сценарий и язык сценария страницы;
PROC_PER - периодичность вычисления сценария виджета;
FLGS - флаги страницы;
ATTRS - перечень атрибутов виджета, модифицированных пользователем;
DBV - версия БД.

- ProjSess(IDW, ID, IO_VAL) - Таблица проекта <*IDW*> для хранения данных сеансов, исполняющих проект.

IDW - полный путь элемента проекта;
ID - атрибут элемента;
IO_VAL - значение атрибута.

- ProjPageIO(IDW, ID, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) - Рабочие атрибуты страниц. Структура фактически совпадает с таблицей LibWidgetIO.
- ProjPageUserIO(IDW, ID, NAME, IO_TP, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) - Пользовательские атрибуты страниц. Структура фактически совпадает с таблицей LibWidgetUserIO.
- ProjPageWIncl(IDW, ID, PARENT, ATTRS, USER, GRP, PERMIT) - Включенные на страницы виджеты. Структура фактически совпадает с таблицей LibWidgetIncl.
- PrjSesIO(IDW, ID, IO_VAL) - Атрибуты <*ID*> элемента сеанса проекта <*IDW*>.

IDW - идентификатор элемента сеанса проекта;
ID - идентификатор IO;
IO_VAL - значение атрибута.

3.10 API пользовательского программирования и сервисные интерфейсы OpenSCADA

3.10.1. API пользовательского программирования

API пользовательского программирования движка визуализации представлено непосредственно объектами OpenSCADA, формирующие пользовательский интерфейс, а именно "Сеансом" и "Виджетами/страницами". Для пользователя эти объекты предоставляют набор функций управления:

Объект "Сеанс" (this.ownerSess())

- *string user( )* - текущий пользователь сеанса.
- *string alrmSndPlay( )* - путь виджета для которого на данный момент воспроизводится сообщение о нарушении.
- *int alrmQuittance(int quit_tmpl, string wpath = "")* - квитирование нарушений <*wpath*> с шаблоном <*quit_tmpl*>. Если <*wpath*> пустая строка то производится глобальное квитирование.

Объект "Виджет" (this)

- *TCntrNodeObj ownerSess( )* - получить объект сеанса данного виджета.
- *TCntrNodeObj ownerPage( )* - получить объект родительской страницы данного виджета.
- *TCntrNodeObj ownerWdg(bool base = false)* - получить родительский виджет данного виджета. При указании <*base*> будет возвращены и объекты страниц.
- *TCntrNodeObj wdgAdd(string wid, string wname, string parent)* - добавление виджета <*wid*> с именем <*wname*> на основе библиотечного виджета <*parent*>.

```
//Добавить новый виджет на основе виджета текстового примитива
nw = this.wdgAdd("nw", "Новый виджет", "/wlb_originals/wdg_Text");
nw.attrSet("geomX", 50).attrSet("geomY", 50);
```

- *bool wdgDel(string wid)* - удаление виджета <*wid*>.
- *bool attrPresent(string attr)* - проверка атрибута виджета <*attr*> на факт присутствия.
- *ElTp attr(string attr)* - получение значения атрибута виджета <*attr*>. Для отсутствующих атрибутов возвращается пустая строка.
- *TCntrNodeObj attrSet(string attr, ElTp vl)* - установка атрибута виджета <*attr*> в значение <*vl*>. Возвращается текущий объект для конкатенации функций установки.
- *string link(string attr, bool prm = false)* - получение ссылки для атрибута виджета <*attr*>. При установке <*prm*> запрашивается ссылка для группы атрибутов (параметр), представленных указанным атрибутом.
- *string linkSet(string attr, string vl, bool prm)* - установка ссылки для атрибута виджета <*attr*>. При установке <*prm*> осуществляется установка ссылка для группы атрибутов (параметр), представленных указанным атрибутом.

```
//Установить ссылку восьмого тренда параметром
this.linkSet("el8.name", "prm:/LogicLev/experiment/Pi", true);
```

Устаревшее API представляется группой функций непосредственно в модуле движка СВУ. Вызов этих функций из скриптов виджетов может осуществляться прямо по идентификатору функции, поскольку их область имён указывается для контекста скриптов виджетов:

Список виджетов (WdgList)

Описание: Возвращает список виджетов в контейнере виджетов или список дочерних виджетов. Если установлено <pg>, то возвращается список страниц для проектов и сеансов.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
list	Список	Строка	Возврат	
addr	Адрес	Строка	Вход	
pg	Страницы	Bool	Вход	0

Присутствие узла (NodePresent)

Описание: Проверка на присутствие узла, включая виджеты, атрибуты и другие.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Bool	Возврат	
addr	Адрес	Строка	Вход	

Список атрибутов (AttrList)

Описание: Возвращает список атрибутов виджета. Если установлен <noUser> тогда возвращаются только не пользовательские атрибуты.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
list	Список	Строка	Возврат	
addr	Адрес	Строка	Вход	
noUser	Без пользовательских	Bool	Вход	1

Запрос атрибута (AttrGet)

Описание: Запрос значения атрибута виджета. Запрос может осуществляться как указанием полного адреса атрибута в <addr>, так и отдельно адреса виджета в <addr>, а идентификатора атрибута в <attr>.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
val	Значение	Строка	Возврат	
addr	Адрес	Строка	Вход	
attr	Атрибут	Bool	Вход	

Установка атрибута (AttrSet)

Описание: Установка значения атрибута виджета. Установка может осуществляться как указанием полного адреса атрибута в <addr>, так и отдельно адреса виджета в <addr>, а идентификатора атрибута в <attr>.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
addr	Адрес	Строка	Вход	
val	Значение	Строка	Вход	
attr	Атрибут	Bool	Вход	

Пользователь сеанса (SesUser)

Описание: Возвращает пользователя сеанса по пути к виджету сеанса.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
user	Пользователь	Строка	Возврат	
addr	Адрес	Строка	Вход	

3.10.2. Сервисные интерфейсы OpenSCADA

Сервисные интерфейсы это интерфейсы доступа к системе OpenSCADA посредством интерфейса управления OpenSCADA из внешних систем. Данный механизм положен в основу всех механизмов обмена внутри OpenSCADA, реализованных посредством слабых связей и стандартного протокола обмена OpenSCADA.

Доступ к значениям атрибутов элементов визуализации (виджеты)

С целью предоставления унифицированного, группового и сравнительно быстрого доступа к значениям атрибутов визуальных элементов предусмотрена сервисная функция визуального элемента "/serv/attr" и команды получения/установки значений атрибутов: <get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/> и <set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 4.13.1.

Таблица 3.10.2.а. Атрибуты команд получения/установки атрибутов визуальных элементов

Id	Имя	Значение
Команда запроса визуальных атрибутов виджета: <get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>		
tm	Время/счётчик изменений	Установка времени/счётчика изменений для запроса только изменившихся атрибутов.
<el id="{attr}" p="{a_id}">{val}</el>	Формирование дочерних элементов с результатами атрибутов	В дочернем элементе указываются: строковых идентификатор {attr} атрибута, индекс {a_id} атрибута и его значение {val}.
Команда установки визуальных атрибутов виджета: <set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>		
<el id="{attr}">{val}</el>	Установка атрибутов	В дочерних элементах указывается идентификатор атрибута {attr} и его значение {val}.

Групповой доступ к значениям атрибутов элементов визуализации (виджетам)

С целью оптимизации трафика сетевого взаимодействия путём исключения мелких запросов, а использования одного, но большого запроса создан групповой запрос значений атрибутов визуальных элементов. Группировка данного запроса подразумевает запрос атрибутов всей ветви виджета, включая и вложенные элементы. Для данного запроса предусмотрена сервисная команда "/serv/attrBr". Запрос данной сервисной команды эквивалентен сервисной команде "/serv/attr" и выглядит следующим образом: *<get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattrBr"/>*

tm - Время/счётчик изменений. Установка времени/счётчика изменений для запроса только изменившихся атрибутов.

Результат:

<el id="{attr}" p="{a_id}">{val}</el> - Элементы с результатами атрибутов. В элементе указываются: строковый идентификатор {attr} атрибута, индекс {a_id} атрибута и его значение {val}.

<w id="{wid}" lnkPath="{lnk_path}">{childs+attrs}</w> - Элементы с дочерними виджетами и их атрибутами. В элементе указываются идентификатор дочернего виджета {wid} и путь виджета на который ссылается данный виджет если он является ссылкой {lnk_path}.

Доступ к страницам сеанса

С целью унификации и оптимизации доступа к страницам предусмотрена сервисная функция сеанса "/serv/pg" и команды запроса перечня открытых страниц (<openlist path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>); открытия страницы (<open

path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>); и закрытия страницы <close path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>).

Результатом запроса перечня открытых страниц являются дочерние элементы <el>{OpPage}</el> содержащие полный путь открытой страницы. Кроме перечня открытых страниц запрос возвращает значение текущего счётчика вычисления сеанса в атрибуте <tm>. Если данный атрибут устанавливается при запросе, то для каждой открытой страницы возвращается список изменённых с момента указанного значения счётчика виджетов открытой страницы.

Управление сигнализацией

Для предоставления механизма глобального контроля за сигнализацией сеанса предусмотрена сервисная функция сеанса "/serv/alarm" и команды запроса статуса сигналов (<get path="/UI/VCAEngine/ses_{Session}/%2fserv%2falarm"/>); и квитирование сигналов (<quittance path="/UI/VCAEngine/ses_{Session}/%2fserv%2falarm"/>).

Запрос статуса сигналов возвращает обобщённое состояние сигналов, а так-же дополнительную информацию для звуковой сигнализации. Дополнительная информация звуковой сигнализации предоставляет текущий ресурс, звуковой файл, для воспроизведения и обеспечивает отслеживание последовательности сигнализации и квитирования отдельных файлов звуковых сообщений.

Запрос на квитирование выполняет квитирование указанного виджета, атрибут <*wdg*>, в соответствии с шаблоном, атрибут <*tmpl*>.

Манипуляция сеансами проектов

Для предоставления унифицированного механизма манипуляции сеансами визуализаторам СВУ в модуле движка СВУ (VCAEngin) предусмотрена сервисная функция "/serv/sess" и команды запроса перечня открытых сеансов, подключения/создания нового сеанса и отключения/удаления сеанса: <list path="/UI/VCAEngine/%2fserv%2fsess"/>, <connect path="/UI/VCAEngine/%2fserv%2fsess"/> и <disconnect path="/UI/VCAEngine/%2fserv%2fsess"/> соответственно. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 4.13.5.

Таблица 3.10.2.b. Атрибуты команд механизма манипуляции сеансами

Id	**Имя**	**Значение**
Команда запроса перечня открытых сеансов для проекта: <list path="/UI/VCAEngine/%2fserv %2fsess"/>		
prj	Указание проекта	Указывает проект, для которого возвращать перечень открытых сеансов.
<el>{Session}</el>	Контроль перечня сеансов	В дочерних элементах указываются сеансы, открытые для запрошенного проекта.
Команда подключения/открытия сеанса: <connect path="/UI/VCAEngine/%2fserv%2fsess"/>		
sess	Установка и контроль имени сеанса	Если атрибут определён, то производится подключение к существующему сеансу, иначе создание нового сеанса. В случае открытия нового сеанса в данный атрибут помещается его имя.
prj	Установка имени проекта	Используется для открытия нового сеанса для указанного проекта и если атрибут {sess} не указан.
Команда отключения/закрытия сеанса: <disconnect path="/UI/VCAEngine/%2fserv%2fsess"/>		
sess	Установка имени сеанса	Указывает имя сеанса от которого выполняется отключение или закрытие. Сеансы, не являющиеся фоновыми и к которым ни один из визуализаторов не подключен, автоматически закрываются.

Групповой запрос дерева библиотек виджетов

С целью оптимизации производительности локального и особенно сетевого взаимодействия предусмотрена сервисная функция "/serv/wlbBr" и команда запроса дерева библиотек виджетов: <get

path="/UI/VCAEngine/%2fserv%2fwlbBr"/>. Результатом запроса является дерево с элементами библиотек виджетов, теги <*wlb*>. Внутри тегов библиотек виджетов содержаться тег иконки <*ico*> и теги виджетов библиотеки <*w*>. Теги виджетов, в свою очередь, содержат тег иконки и теги дочерних виджетов <*cw*>.

4. Конфигурация модуля посредством интерфейса управления OpenSCADA

Посредством интерфейса управления OpenSCADA компоненты, которые его используют, можно конфигурировать из любого конфигуратора системы OpenSCADA. Данным модулем предоставляется интерфейс для доступа ко всем объектам данных СВУ. Главная вкладка конфигурационной страницы модуля предоставляет доступ к библиотекам виджетов и проектам (рис. 4.1). Вкладка "Сеансы" предоставляет доступ и открытым сеансам проектов (рис. 4.2). Для настройки движка синтеза речи предусмотрена соответствующая страница (рис. 4.3).

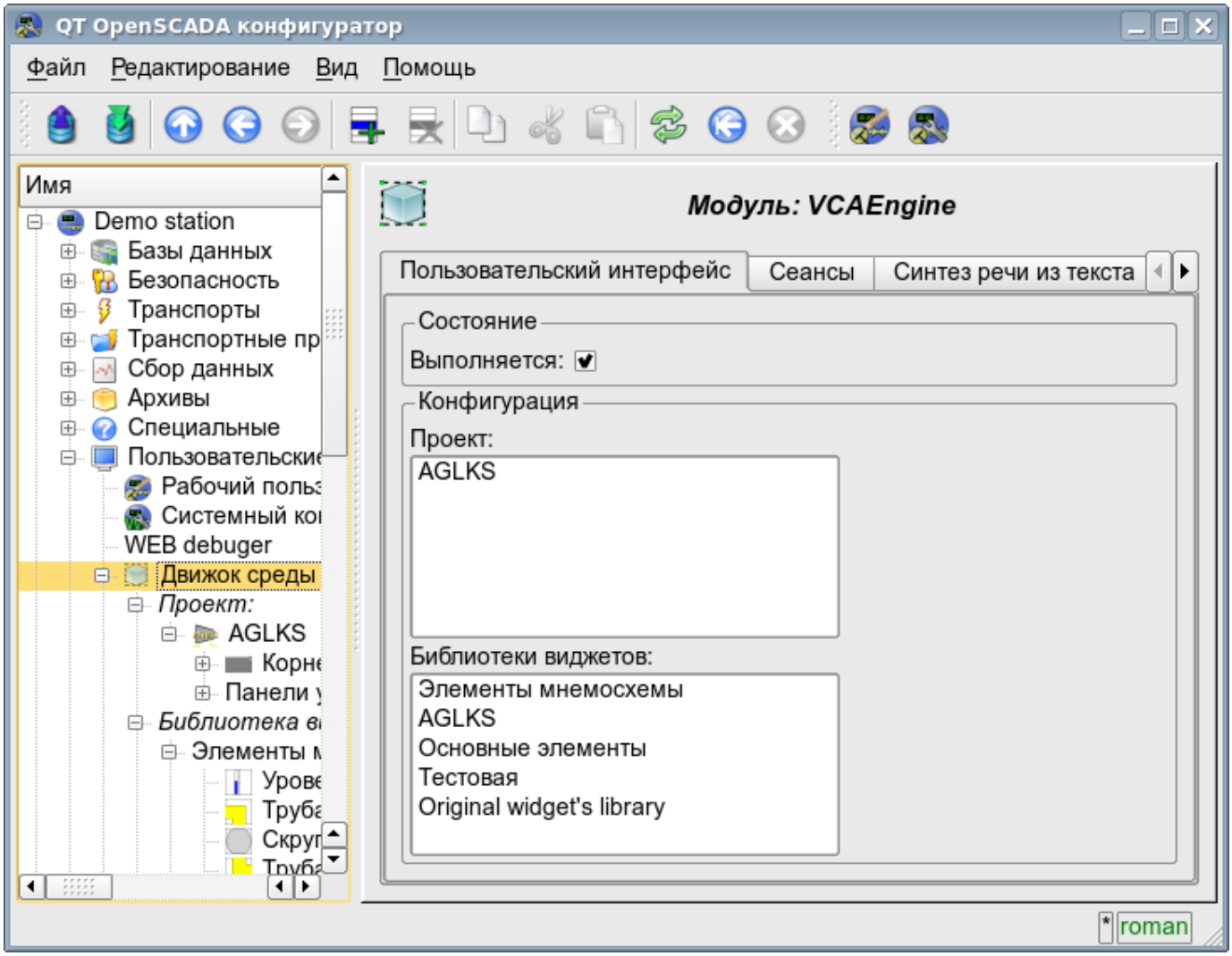


Рис.4.1 Главная конфигурационная страница модуля.

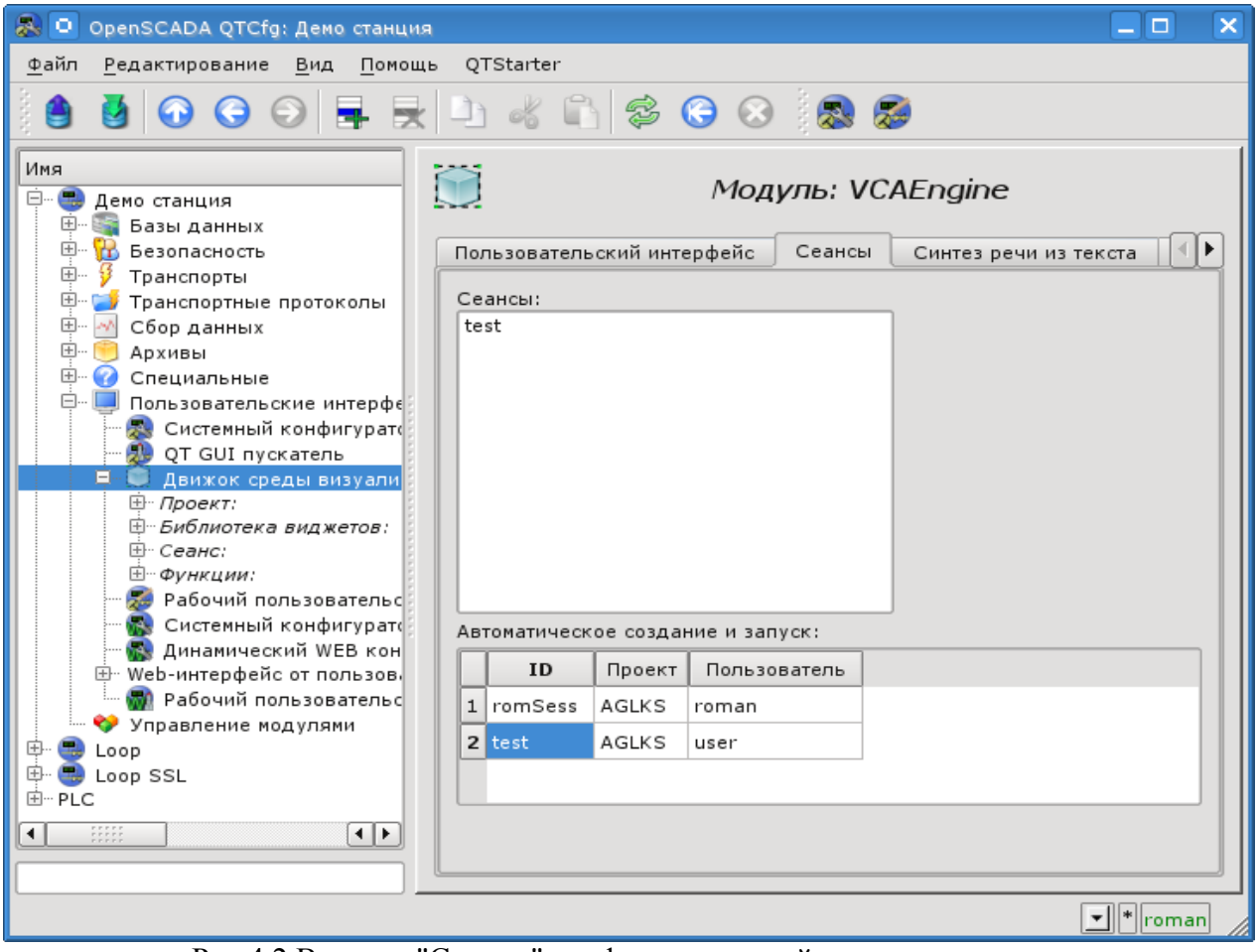


Рис.4.2 Вкладка "Сеансы" конфигурационной страницы модуля.

Кроме списка открытых сеансов вкладка на рисунке 4.2 содержит таблицу с перечнем сеансов, которые должны создаваться и запускаться в момент загрузки системы OpenSCADA. Создание сеансов посредством этого инструмента может быть полезным для Web-интерфейса. В таком случае при подключении Web-пользователя все данные уже готовы и обеспечивается непрерывность формирования архивных документов.

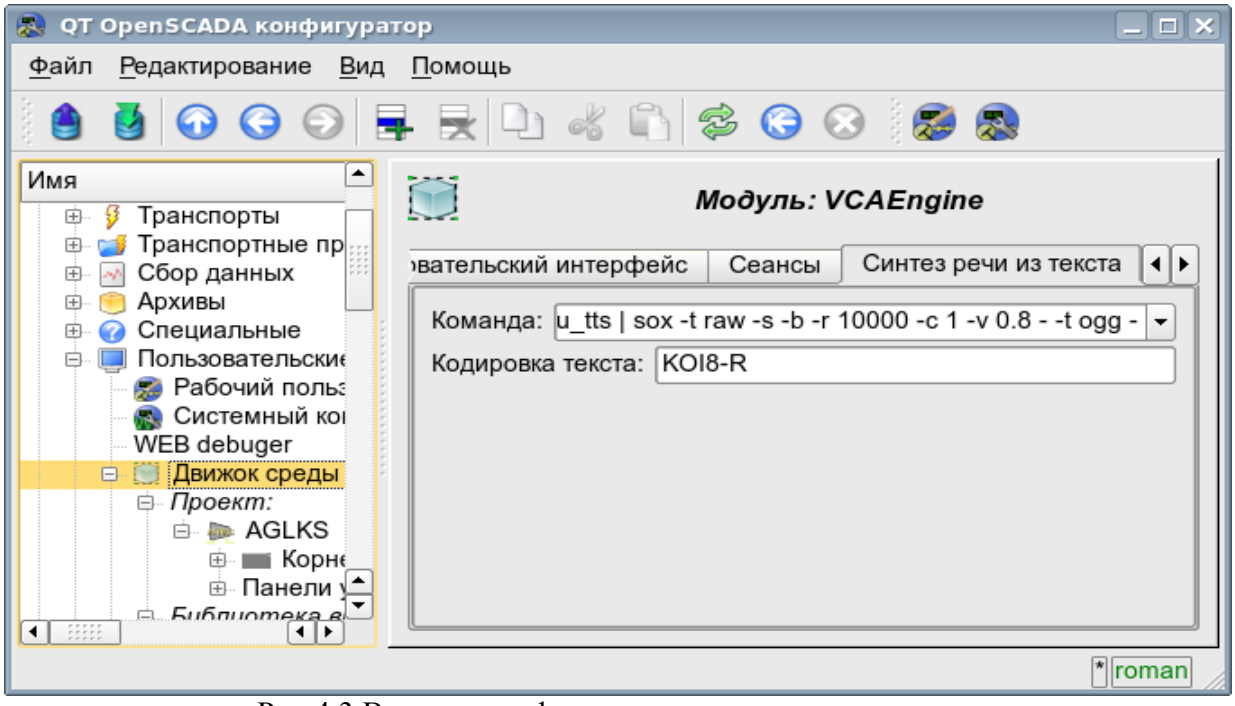


Рис.4.3 Вкладка конфигурации движка синтеза речи.

Конфигурация контейнеров виджетов, в лице библиотек виджетов и проектов выполняется посредством страниц на рис. 4.4 (для проекта) и рис.4.5 (для библиотеки виджетов). Библиотека виджетов содержит виджеты, а проект - страницы. Оба типа контейнера содержат вкладку конфигурации Mime-данных, используемых виджетами (рис.4.6).

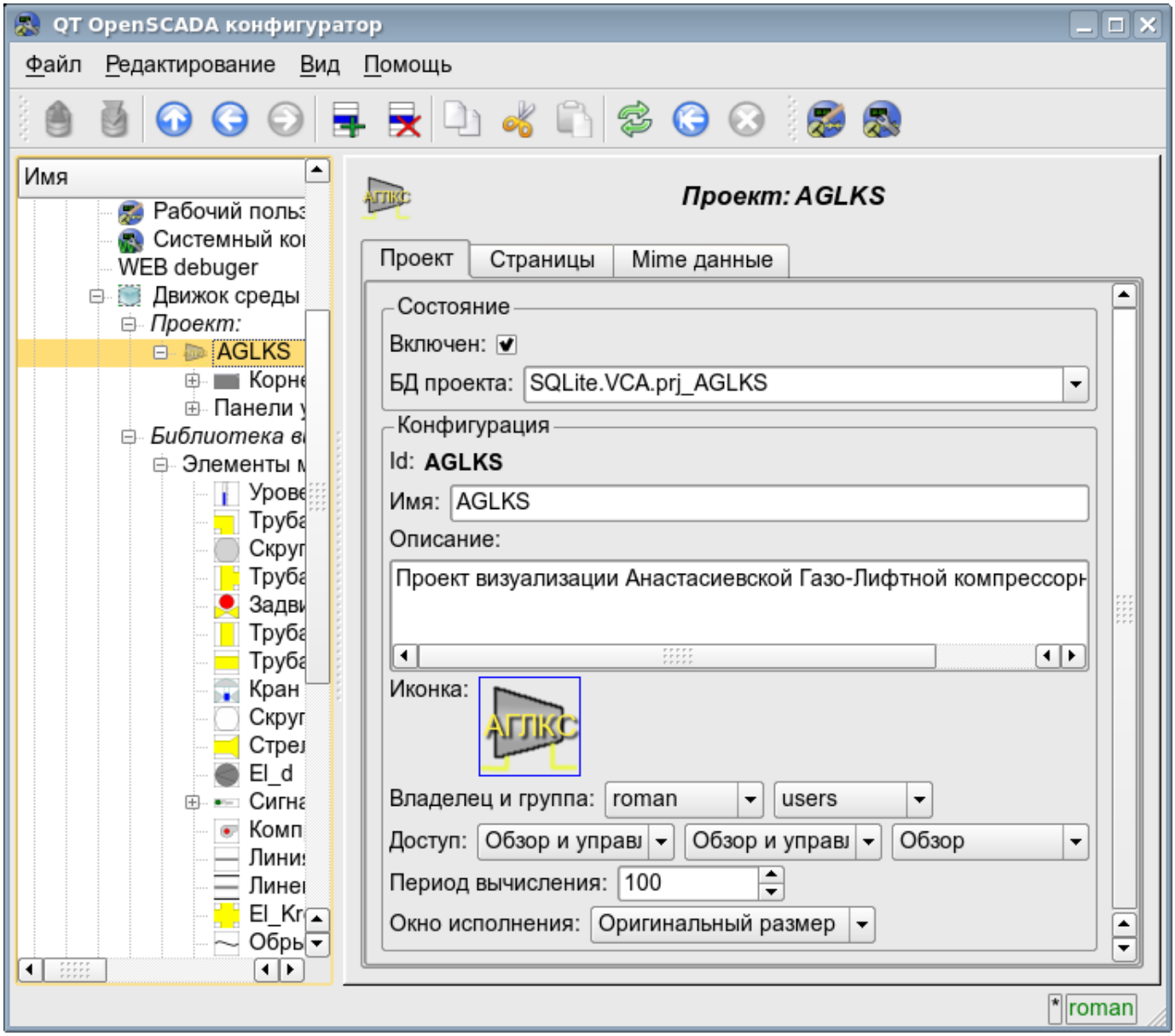


Рис.4.4 Страница конфигурации проектов.

С помощью этой страницы можно установить:

- Состояние контейнера, а именно: «Включен», имя БД, содержащей конфигурацию, владельца и группу контейнера.
- Идентификатор, имя, описание и иконку контейнера.
- Права доступа к контейнеру.
- Период вычисления сеансов основанных на данном проекте.
- Способ открытия главного окна исполнения (Оригинальный размер, максимизация и на весь экран).

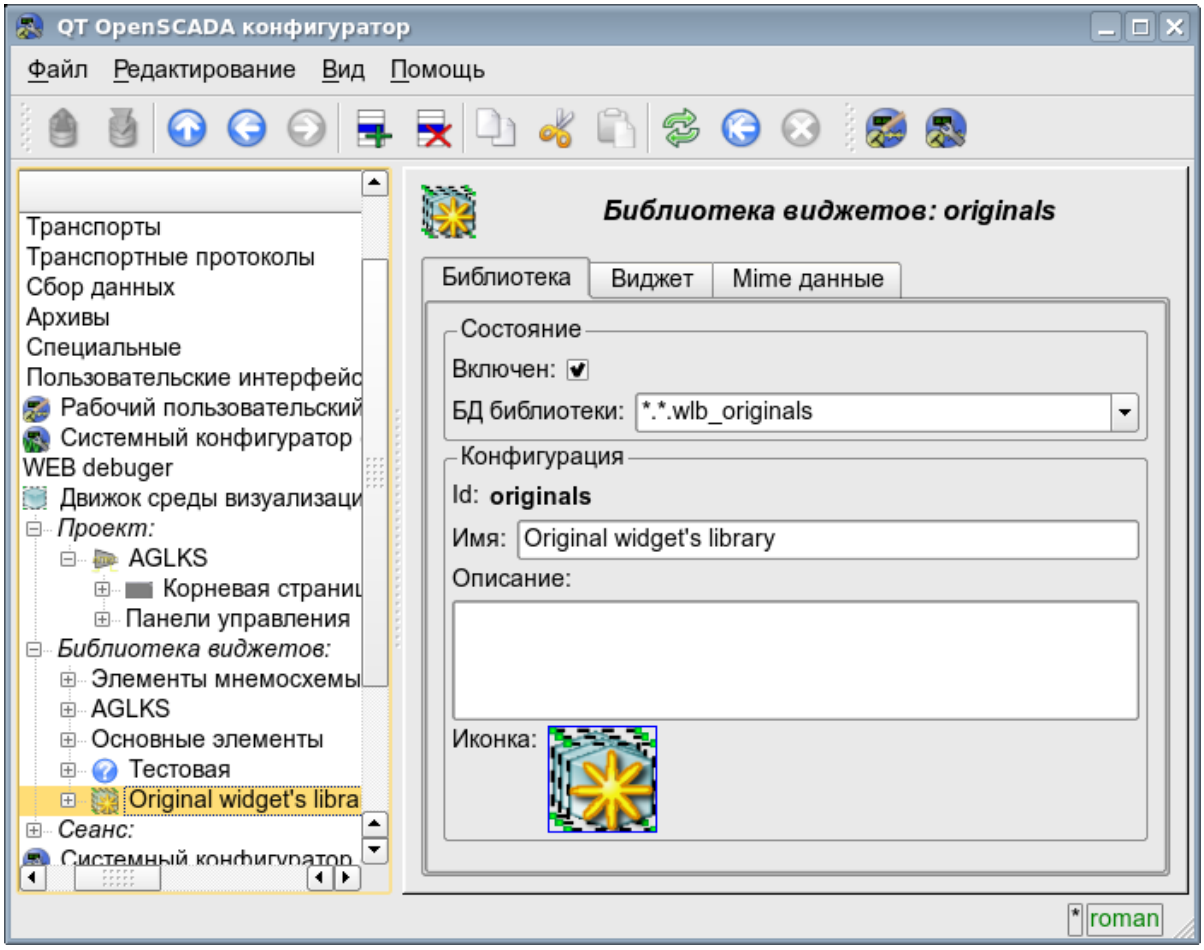


Рис.4.5 Страница конфигурации библиотек виджетов.

С помощью этой страницы можно установить:

- Состояние контейнера, а именно: «Включен», имя БД, содержащей конфигурацию.
- Идентификатор, имя, описание и иконку контейнера.

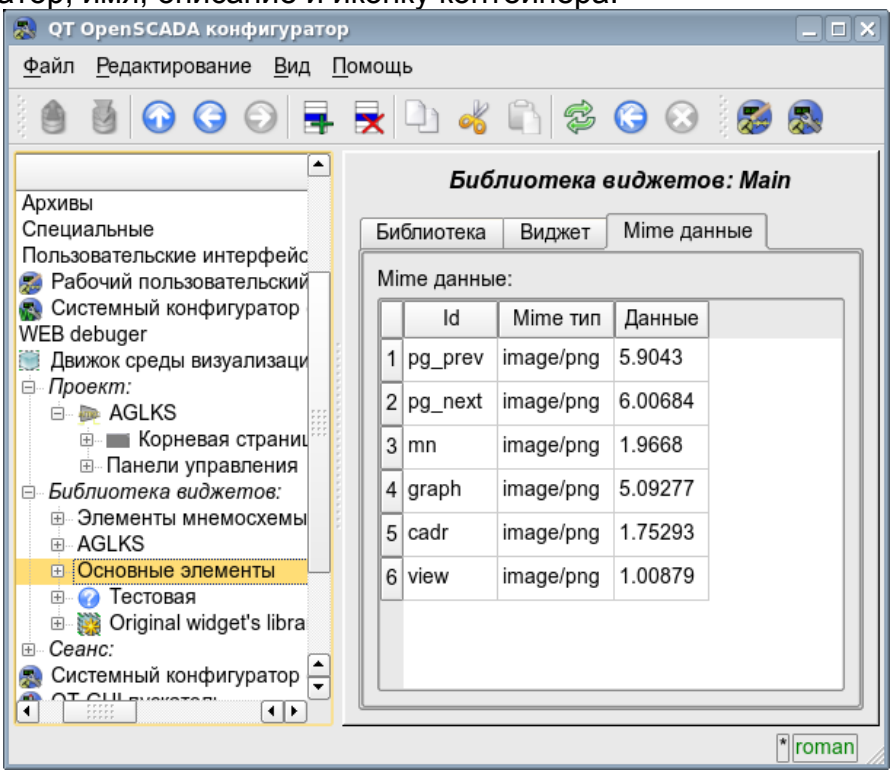


Рис.4.6 Вкладка конфигурации mime-данных контейнера.

Конфигурация сеанса проекта значительно отличается от конфигурации проекта (рис. 4.7), однако также содержит страницы проекта.

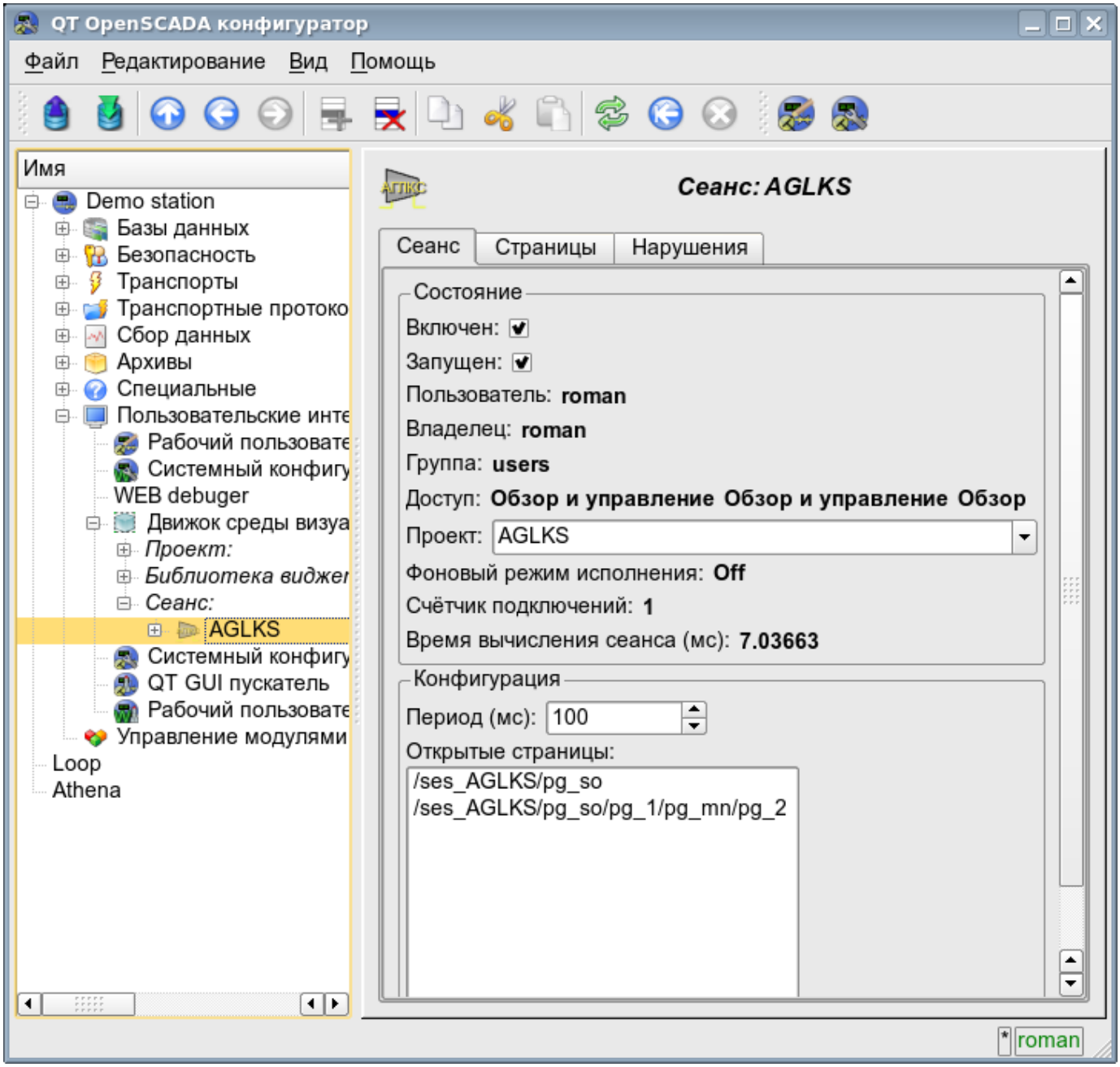


Рис.4.7 Страница конфигурации сеансов проектов.

С помощью этой страницы можно установить:

- Состояние сеанса, а именно: "Включен", "Запущен", пользователь, владелец, группа пользователей, доступ, исходный проект, режим исполнения в фоне, счётчик клиентских подключений и время исполнения сеанса.
- Период обсчёта сеанса.
- Перечень открытых страниц.

Страницы конфигурации визуальных элементов, расположенных в разных контейнерах, могут сильно отличаться, однако, это отличие заключается в наличии или отсутствии отдельных вкладок. Главная вкладка визуальных элементов фактически везде одинакова, отличаясь на одно конфигурационное поле (рис. 4.8). У страниц присутствуют вкладки дочерних страниц и вложенных виджетов. У контейнерных виджетов содержится вкладка вложенных виджетов. Все визуальные элементы содержат вкладку атрибутов (рис. 4.9), кроме логических контейнеров проектов. Элементы, на уровне которых можно формировать пользовательскую процедуру и определять связи, содержат вкладки "Обработка" (рис. 4.10) и "Связи" (рис.4.11).

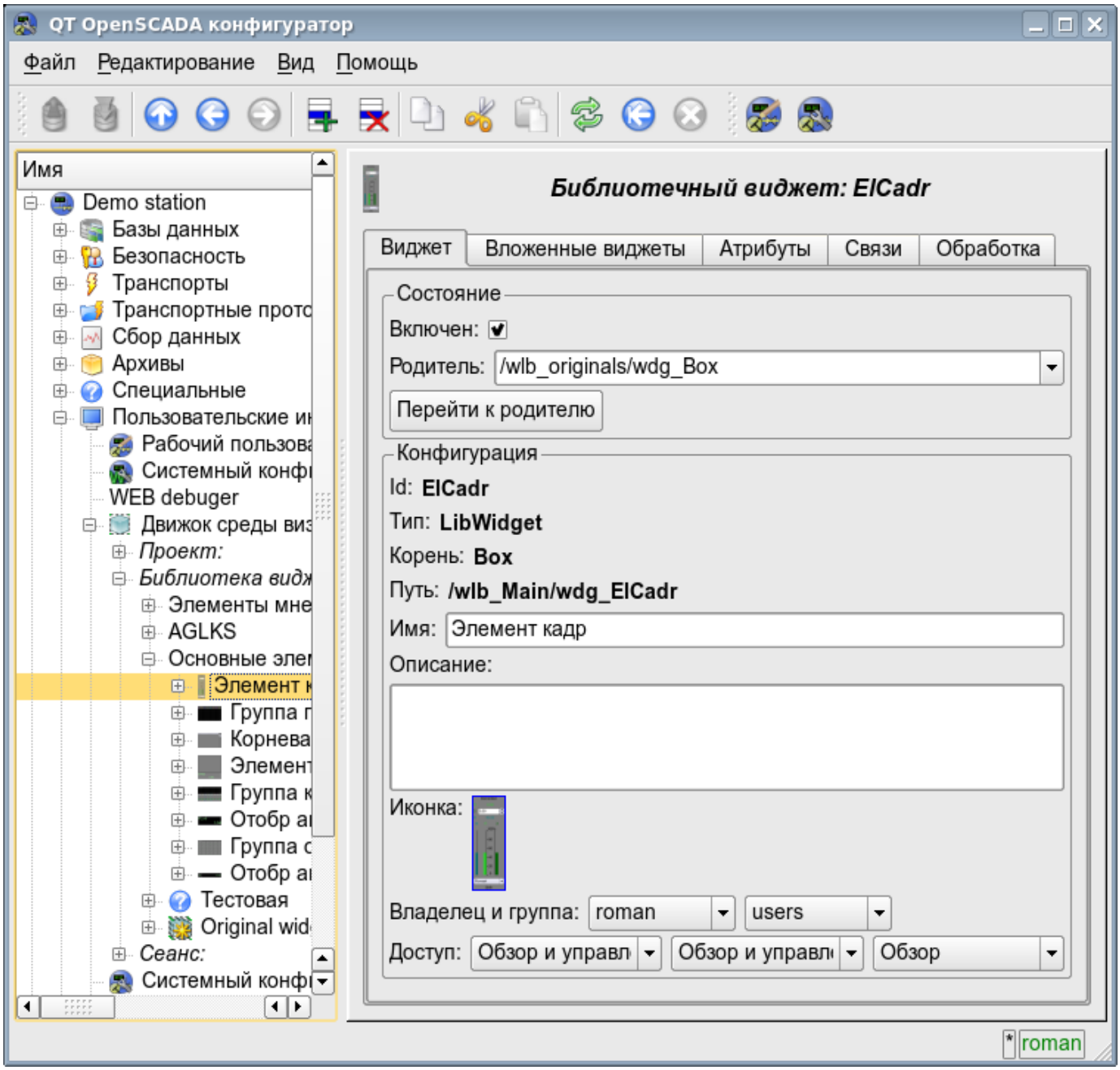


Рис.4.8 Главная вкладка конфигурации визуальных элементов.

С помощью этой страницы можно установить:

- Состояние элемента, а именно: «Включен», родительский элемент и переход к нему. Для страницы в состоянии указывается тип страницы.
- Идентификатор, тип, корень, путь, имя, описание и иконку элемента.
- Владелец, группа пользователей и права доступа к элементу.

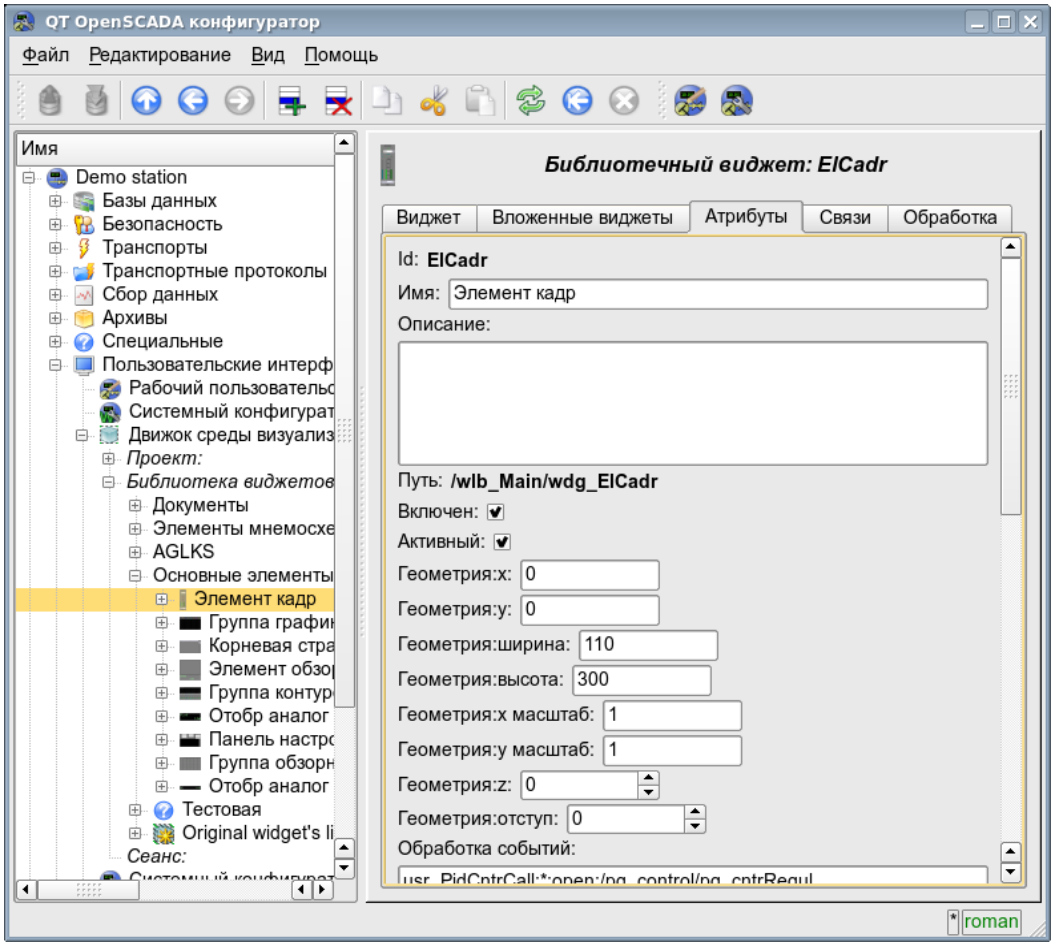


Рис.4.9 Вкладка атрибутов визуальных элементов.

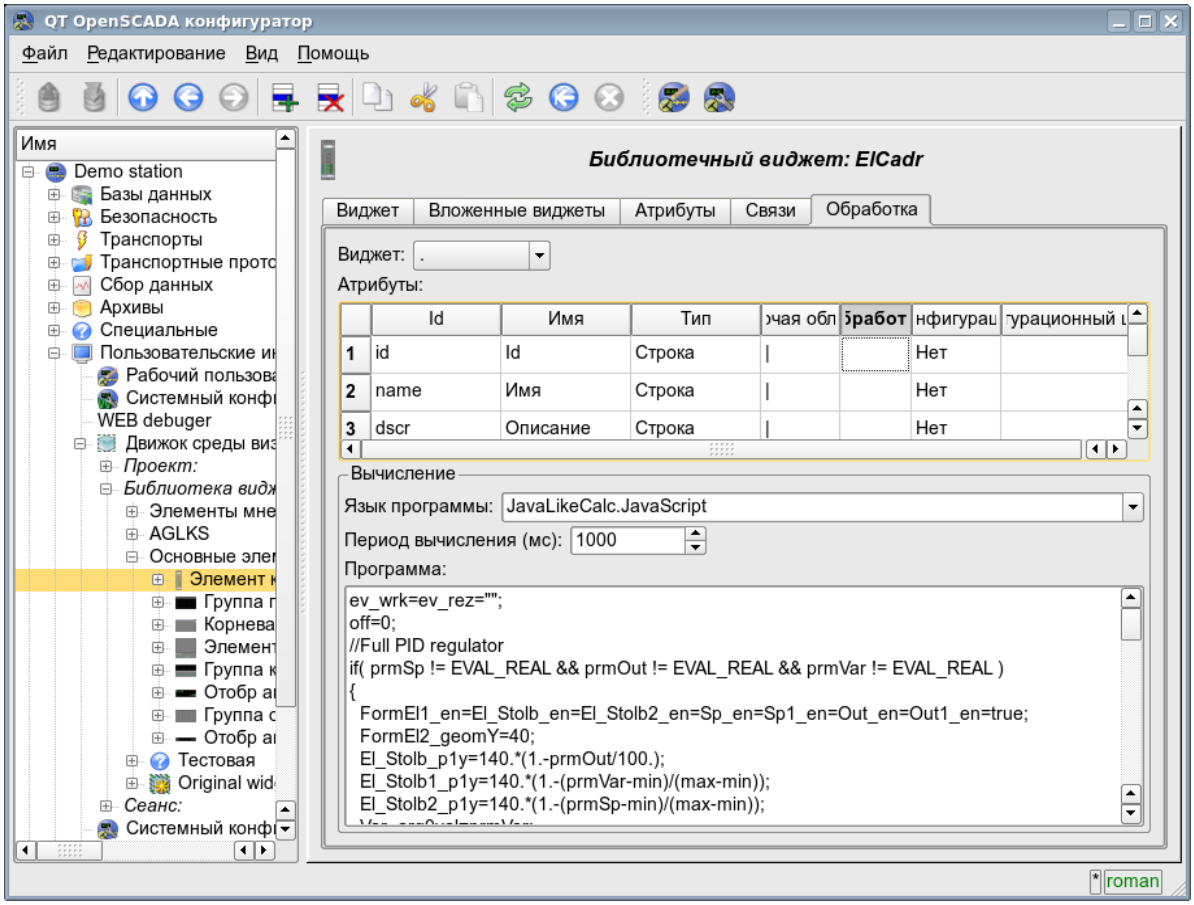


Рис.4.10 Вкладка обработки визуальных элементов.

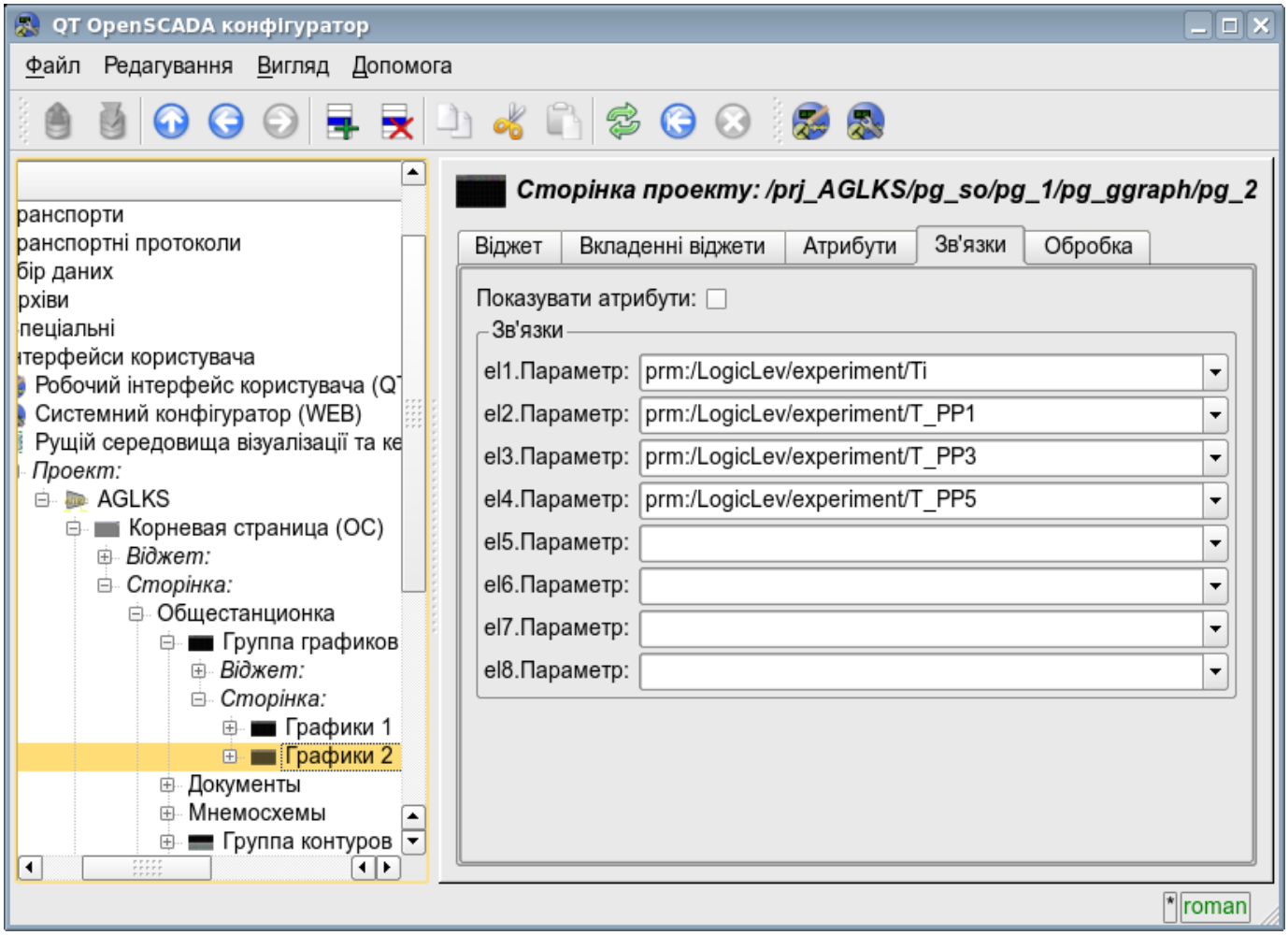


Рис.4.11 Вкладка связей визуальных элементов.

Модуль подсистемы “Пользовательские интерфейсы” <Vision>

Модуль:	Vision
Имя:	Рабочий пользовательский интерфейс (QT)
Тип:	Пользовательские интерфейсы
Источник:	ui_Vision.so
Версия:	1.0.0
Автор:	Роман Савоченко
Разработчик:	Роман Савоченко, Максим Лысенко, Ксения Яшина
Описание:	Визуальный рабочий пользовательский интерфейс.
Лицензия:	GPL

Модуль Vision предоставляет механизм конечной визуализации среды визуализации и управления (СВУ) в систему OpenSCADA. Модуль основан на многоплатформенной библиотеке графического пользовательского интерфейса (GUI) фирмы TrollTech – QT(http://www.trolltech.com/qt/). В своей работе модуль использует данные движка СВУ (модуль VCAEngine).

Среда визуализации и управления (СВУ) является неотъемлемой составляющей SCADA системы. Она применяется на клиентских станциях с целью доступного предоставления информации об объекте управления и выдачи управляющих воздействий на объект. В различных практических случаях и условиях могут применяться СВУ, построенные на различных принципах визуализации. Например, это могут быть библиотеки виджетов QT, GTK+, wxWidgets или гипертекстовые механизмы на основе технологий HTML, XHTML, XML, CSS и JavaScript, или же сторонние приложения визуализации, реализованные на различных языках программирования: Java, Python и т.д. Любой из этих принципов имеет свои преимущества и недостатки, комбинация которых может стать непреодолимым препятствием в возможности использования СВУ в том или ином практическом случае. Например, технологии вроде библиотеки QT позволяют создавать высокореактивные СВУ, что несомненно важно для станций оператора управления технологическим процессом (ТП). Однако необходимость инсталляции данного клиентского ПО в отдельных ситуациях может сделать использование его невозможным. С другой стороны Web-технологии не требуют инсталляции на клиентские системы и являются предельно многоплатформенными (достаточно создать ссылку на Web-сервер в любом Web-браузере), что наиболее важно для различных инженерных и административных станций. Но реактивность и надёжность таких интерфейсов ниже, что практически исключает их использования на станциях оператора ТП.

Система OpenSCADA имеет гибкую архитектуру, которая позволяет создавать внешние интерфейсы, в том числе и пользовательские, на любой основе и на любой вкус. Например, среда конфигурации системы OpenSCADA доступна как на QT библиотеке, так и на Web-основе.

В тоже время независимое создание реализаций СВУ на различной основе может повлечь за собой невозможность использования данных конфигурации одной СВУ в другой, что неудобно и ограничено с пользовательской стороны, а также накладно в плане реализации и последующей поддержки. С целью избежания этих проблем, а также создания в кратчайшие сроки полного спектра различных типов СВУ основан проект создания концепции СВУ. Результатом этого проекта и стал данный модуль непосредственной визуализации (на основе библиотеки QT), модуль непосредственной визуализации WebVision и движок СВУ VCAEngine.

1. Назначение

Данный модуль непосредственной визуализации СВУ предназначен для формирования и исполнения интерфейсов СВУ в среде графической библиотеки QT.

Финальная версия этого модуля СВУ, построенная на основе данного модуля, обеспечит:

- три уровня сложности в формировании интерфейса визуализации, позволяющие органично осваивать и применять инструментарий по методике от простого к сложному:
  - формирование из шаблонных кадров путём назначения динамики (без графической конфигурации);
  - графическое формирование новых кадров путём использования готовых элементов визуализации из библиотеки (мнемосхемы);
  - формирование новых кадров, шаблонных кадров и элементов отображение в библиотеки.
- построение интерфейсов визуализации различной сложности, начиная от простых плоских интерфейсов мониторинга и заканчивая полноценными иерархическими интерфейсами, используемыми в SCADA системах;
- предоставление различных способов формирования и конфигурации пользовательского интерфейса, основанных на различных интерфейсах графического представления (QT, Web, Java ...) или же посредством стандартного интерфейса управления системой OpenSCADA;
- смену динамики в процессе исполнения;
- построение новых шаблонных кадров на уровне пользователя и формирование специализированных под область применения библиотек кадров (например, включение кадров параметров, графиков и других элементов с увязкой их друг с другом), в соответствии с теорией повторного использования и накопления;
- построение новых пользовательских элементов визуализации и формирование специализированных под область применения библиотек кадров в соответствии с теорией повторного использования и накопления;
- описание логики новых шаблонных кадров и пользовательских элементов визуализации как простыми связями, так и лаконичным, полноценным языком пользовательского программирования;
- возможность включения в пользовательские элементы визуализации функций (или кадров вычисления функций) объектной модели OpenSCADA, практически связывая представление с алгоритмом вычисления (например, визуализируя библиотеку моделей аппаратов ТП для последующего визуального построения моделей ТП);
- разделение данных пользовательских интерфейсов и интерфейсов представления этих данных, позволяющее строить интерфейс пользователя в одной среде, а исполнять во многих других (QT, Web, Java ...);
- возможность подключение к исполняющемуся интерфейсу для наблюдения и коррекции действий (например, при обучении операторов и контроля в реальном времени за его действиями);
- визуальное построение различных схем с наложением логических связей и последующем централизованным исполнением в фоне (визуальное построение и исполнение математических моделей, логических схем, релейных схем и иных процедур);
- предоставление функций объектного API в систему OpenSCADA, может использоваться для управления свойствами интерфейса визуализации из пользовательских процедур;
- построение серверов кадров, элементов визуализации и проектов интерфейсов визуализации с возможностью обслуживания множественных клиентских соединений;
- простая организация клиентских станций на различной основе (QT, Web, Java ...) с подключением к центральному серверу;
- полноценный механизм разделения полномочий между пользователями, позволяющий создавать и исполнять проекты с различными правами доступа к его компонентам;
- гибкое формирование правил сигнализаций и уведомления, с учётом и поддержкой различных способов уведомления;
- поддержка пользовательского формирования палитры и шрифтовых предпочтений для

интерфейса визуализации;

- поддержка пользовательского формирования карт событий под различное оборудование управления и пользовательские предпочтения;
- поддержка профилей пользователей, позволяющая определять различные свойства интерфейса визуализации (цветовая гамма, шрифтовые особенности, предпочтительные карты событий);
- гибкое хранение и распространение библиотек виджетов, кадров и проектов интерфейсов визуализации в БД, поддерживаемых системой OpenSCADA; практически пользователю нужно только зарегистрировать полученную БД с данными.

2. Инструмент графического формирования интерфейса СВУ

Разработка интерфейса СВУ выполняется в одном окне, реализующем многодокументный интерфейс (MDI) (рис. 1). Данный подход позволяет одновременно редактировать несколько кадров различных размеров. Использованы следующие механизмы управления разработкой: панели инструментов, пункты меню и контекстное меню. Большинство действий дублируются разными механизмами, что позволяет быстро найти инструмент предпочитаемым способом. Навигационные интерфейсы реализованы присоединяемыми окнами. Конфигурация панелей инструментов и присоединяемых окон сохраняется при выходе и восстанавливается при старте, что позволяет настраивать интерфейс под себя

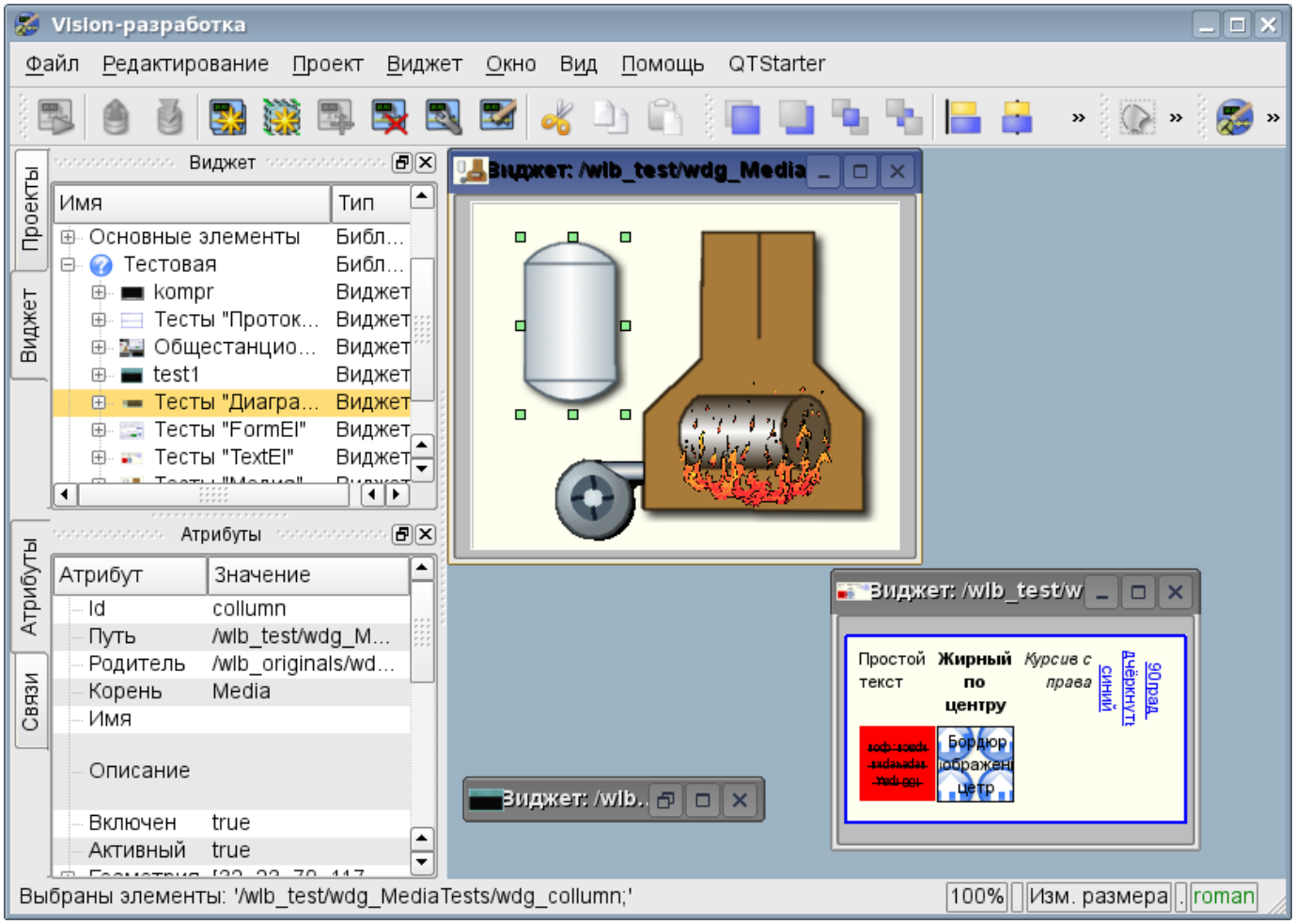


Рис.1. Окно разработки интерфейсов СВУ.

Доступ к основным компонентам СВУ производится посредством пришвартованных окон, на рис.1 эти окна изображены слева. В этих окнах содержатся:

- Дерево библиотек виджетов. С помощью навигатора можно быстро найти нужный виджет или библиотеку и проделать над ними необходимые операции. Реализованы операции добавления, удаления, копирования, настройки виджетов и библиотек, а также очистки и визуального редактирования виджетов. Для гибкого управления поддерживается контекстное меню в составе пунктов:
  - "Новая библиотека" - создание новой библиотеки.
  - "Добавить визуальный элемент" - добавление визуального элемента в библиотеку.
  - "Удалить визуальный элемент" - удаление визуального элемента из библиотеки.
  - "Очистить визуальный элемент" - очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию.
  - "Свойства визуального элемента" - конфигурация визуального элемента.
  - "Редактировать визуальный элемент" - визуальное редактирование элемента.
  - "Вырезать визуальный элемент" - вырезание/перемещение визуального элемента в

момент вставки.
- "Копировать визуальный элемент" - копирование визуального элемента в момент вставки.
- "Вставить визуальный элемент" - вставка визуального элемента.
- "Загрузить из БД" - загрузка данных визуального элемента из БД.
- "Сохранить в БД" - сохранение данных визуального элемента в БД.
- "Обновить библиотеки" - выполняет перечитывание конфигурации и состава библиотек из модели данных.

- Дерево страниц проектов. Предусматривает механизм "Перетащи и брось" для формирования пользовательских кадров на основе элементов библиотек. Для гибкого управления поддерживает контекстное меню в составе пунктов:
  - "Запустить исполнение проекта" - запуск исполнения выбранного проекта.
  - "Новый проект" - создание нового проекта.
  - "Добавить визуальный элемент" - добавление визуального элемента в проект/страницу.
  - "Удалить визуальный элемент" - удаление визуального элемента из проекта/страницы.
  - "Очистить визуальный элемент" - очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию.
  - "Свойства визуального элемента" - конфигурация визуального элемента.
  - "Редактировать визуальный элемент" - визуальное редактирование элемента.
  - "Вырезать визуальный элемент" - вырезание/перемещение визуального элемента в момент вставки.
  - "Копировать визуальный элемент" - копирование визуального элемента в момент вставки.
  - "Вставить визуальный элемент" - вставка визуального элемента.
  - "Загрузить из БД" - загрузка данных визуального элемента из БД.
  - "Сохранить в БД" - сохранение данных визуального элемента в БД.
  - "Обновить проекты" - выполняет перечитывание конфигурации и состава проектов из модели данных.
- атрибуты виджетов;
- внешние связи виджетов.

В основном пространстве рабочего окна размещаются окна страниц проектов, кадров библиотек виджетов, пользовательских элементов и элементов примитивов на момент их визуального редактирования.

Сверху рабочего окна содержится меню. В меню размещены все инструменты, необходимые для разработки интерфейсов СВУ. Меню имеет следующую структуру:
- "Файл" - Общие операции.
  - "Загрузить из БД" - загрузка данных визуального элемента из БД.
  - "Сохранить в БД" - сохранение данных визуального элемента в БД.
  - "Закрыть" - закрыть окно редактора.
  - "Выход" - Выход из системы OpenSCADA.
- "Редактирование" - Операции редактирования визуальных элементов.
  - "Вырезать визуальный элемент" - вырезание/перемещение визуального элемента в момент вставки.
  - "Копировать визуальный элемент" - копирование визуального элемента в момент вставки.
  - "Вставить визуальный элемент" - вставка визуального элемента.
- "Проект" - Операции над проектами.
  - "Запустить исполнение проекта" - запуск исполнения выбранного проекта.
  - "Новый проект" - создание нового проекта.
  - "Добавить визуальный элемент" - добавление визуального элемента в проект.
  - "Удалить визуальный элемент" - удаление визуального элемента из проекта.
  - "Очистить визуальный элемент" - очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию.

        - "Свойства визуального элемента" - конфигурация визуального элемента.
        - "Редактировать визуальный элемент" - визуальное редактирование элемента.
    - "Виджет" - Операции над виджетами и библиотеками виджетов.
        - "Новая библиотека" - создание новой библиотеки.
        - "Добавить визуальный элемент" - добавление визуального элемента в библиотеку.
        - "Удалить визуальный элемент" - удаление визуального элемента из библиотеки.
        - "Очистить визуальный элемент" - очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию.
        - "Свойства визуального элемента" - конфигурация визуального элемента.
        - "Редактировать визуальный элемент" - визуальное редактирование элемента.
        - "Вид" - управление расположением визуальных элементов на кадрах.
            - "Виджет на верх" - поднятие виджета на самый верх.
            - "Виджет на низ" - опускание виджета на самый низ.
            - "Поднять виджет" - поднять виджет выше.
            - "Опустить виджет" - опустить виджет ниже.
            - "Выравнять слева" - выравнивание виджета слева.
            - "Выровнять в центре вертикально" - выравнивание виджета вертикально в центре.
            - "Выровнять справа" - выравнивание виджета справа.
            - "Выровнять сверху" - выравнивание виджета сверху.
            - "Выровнять в центре горизонтально" - выравнивание виджета горизонтально в центре.
            - "Выровнять снизу" - выравнивание виджета снизу.
        - "Библиотека: {Имя библиотеки}" - пункты меню для доступа к кадрам/виджетам содержащимся в библиотеке.
    - "Окно" - Управление окнами MDI-интерфейса.
        - "Закрыть" - закрыть активное окно.
        - "Закрыть все" - закрыть все окна.
        - "Уложить" - уложить все окна для видимости одновременно.
        - "Каскадировать" - расположить все окна каскадом.
        - "Следующий" - активировать следующее окно.
        - "Предыдущий" - активировать предыдущее окно.
        - "Виджет: {Имя виджета}" - пункты активации конкретного окна.
    - "Вид" - Управление видимостью рабочего окна и панелей на нём.
        - "Панель визуальных элементов" - панель управления визуальными элементами.
        - "Функции видимости виджетов" - панель управления видимостью и расположением виджетов на панелях.
        - "Панель элементарных фигур" - дополнительная панель редактирования примитива элементарных фигур ("ElFigure").
        - "Проекты" - пришвартованное окно управления деревом проектов.
        - "Виджеты" - пришвартованное окно управления деревом библиотек виджетов.
        - "Атрибуты" - пришвартованное окно диспетчера атрибутов.
        - "Связи" - пришвартованное окно диспетчера связей.
        - "Библиотека: {Имя библиотеки}" - управление видимостью панелей библиотек виджетов.
    - "Помощь" - Помощь по OpenSCADA и модулю Vision.
        - "Про" - информация про данный модуль.
        - "Про QT" - информация о библиотеке QT, используемой модулем.
        - "Что это" - запрос описания элементов интерфейса окна.

Сверху, под меню, или по сторонам располагаются панели инструментов. Панели могут быть скрыты или расположены, что управляется в пункте меню "Вид". Существуют следующие панели инструментов:

- "Панель визуальных элементов" - Панель управления визуальными элементами:
    - "Запуск исполнения проекта для выбранного элемента" - производит запуск проекта на исполнение и активацию выбранной страницы проекта.

- "Загрузить данные элемента из БД" - выполняет загрузку данных выбранного(ых) элементов из БД.
- "Сохранить данные элемента в БД" - выполняет сохранение данных выбранного(ых) элементов в БД.
- "Новый проект" - создание нового проекта.
- "Новая библиотека" - создание новой библиотеки.
- "Добавить визуальный элемент" - добавление визуального элемента в проект.
- "Удалить визуальный элемент" - удаление визуального элемента из проекта.
- "Свойства визуального элемента" - конфигурация визуального элемента.
- "Редактировать визуальный элемент" - визуальное редактирование элемента.
- "Вырезать визуальный элемент" - вырезание/перемещение визуального элемента в момент вставки.
- "Копировать визуальный элемент" - копирование визуального элемента в момент вставки.
- "Вставить визуальный элемент" - вставка визуального элемента.

- "Функции видимости виджетов" - Панель управления видимостью и расположением виджетов на панелях:
  - "Виджет на верх" - поднятие виджета на самый верх.
  - "Виджет на низ" - опускание виджета на самый низ.
  - "Поднять виджет" - поднять виджет выше.
  - "Опустить виджет" - опустить виджет ниже.
  - "Выравнять слева" - выравнивание виджета слева.
  - "Выровнять в центре вертикально" - выравнивание виджета вертикально в центре.
  - "Выровнять справа" - выравнивание виджета справа.
  - "Выровнять сверху" - выравнивание виджета сверху.
  - "Выровнять в центре горизонтально" - выравнивание виджета горизонтально в центре.
  - "Выровнять снизу" - выравнивание виджета снизу.
- "Панель элементарных фигур" - Дополнительная панель редактирования примитива элементарных фигур ("ElFig").
  - "Возврат курсора" - возврат к курсору для действий над фигурами на виджете.
  - "Добавить линию" - добавление линии к элементарной фигуре.
  - "Добавить дугу" - добавление дуги к элементарной фигуре.
  - "Добавить кривую безье" - добавление кривой безье к элементарной фигуре.
  - "Включить привязки" - включение привязок у элементарной фигуры.
- "Библиотека: {Имя библиотеки}" - Управление видимостью панелей библиотек виджетов. Содержимое панели зависит от содержимого библиотеки и включает кнопки вызова элементов библиотеки.
- "QTStarter панель инструментов" - Панель, созданная модулем запуска модулей библиотеки QT. Содержит кнопки для запуска UI модулей системы OpenSCADA, основанных на библиотеке QT. С помощью этой панели можно открыть несколько экземпляров окон данного модуля или других модулей.

Внизу окна разработки СВУ располагается строка статуса. С правой стороны строки статуса размещаются индикаторы визуального масштаба редактируемого кадра, режима изменения размера элементов, режима текущей станции движка СВУ и пользователя, от имени которого ведётся разработка интерфейса СВУ. По двойному клику на индикаторе пользователя можно пользователя сменить, введя новое имя и пароль пользователя. В главное поле строки статуса выводятся различные информационные сообщения и сообщения помощи.

Для редактирования свойств визуальных элементов предусмотрено два диалога. Первый диалог позволяет редактировать свойства контейнеров визуальных элементов (библиотек виджетов и проектов), рис, 2. Второй диалог для редактирования свойств самих визуальных элементов, рис. 3. Изменения, внесённые в диалогах, сразу же попадают в движок СВУ. Для сохранения этих изменения в БД или восстановления из БД необходимо воспользоваться соответствующими инструментами главного окна разработки.

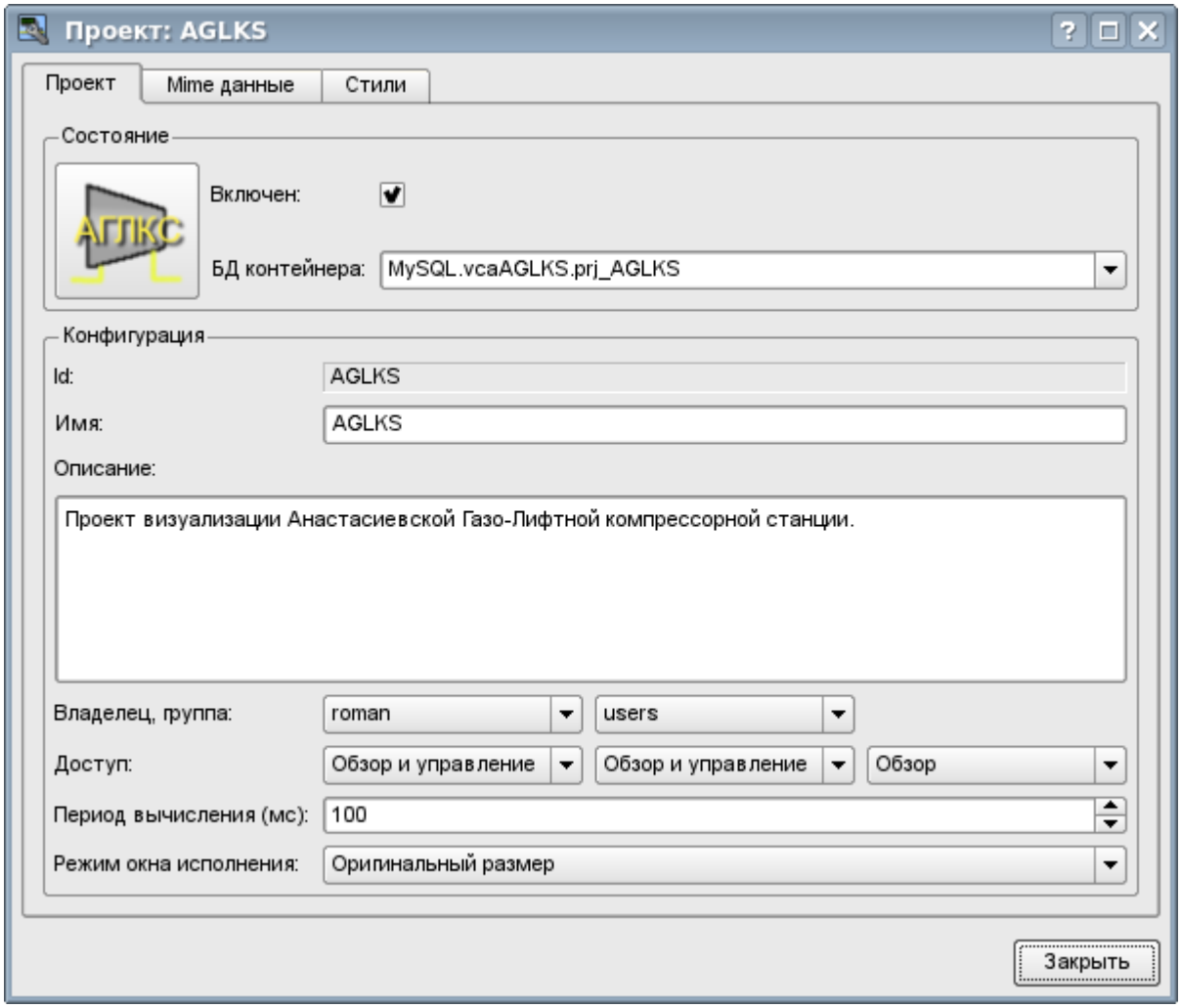


Рис.2. Диалог редактирования свойств контейнеров визуальных элементов.

С помощью главной вкладки этого диалога можно установить:

- Состояние контейнера элементов, а именно: "Включен", БД контейнера.
- Идентификатор, имя и описание контейнера.
- Для проекта: пользователя, группу пользователей и доступ пользователя, группы пользователей и всех остальных.
- Для проекта: период вычисления проекта и режим открытия окна при исполнении.

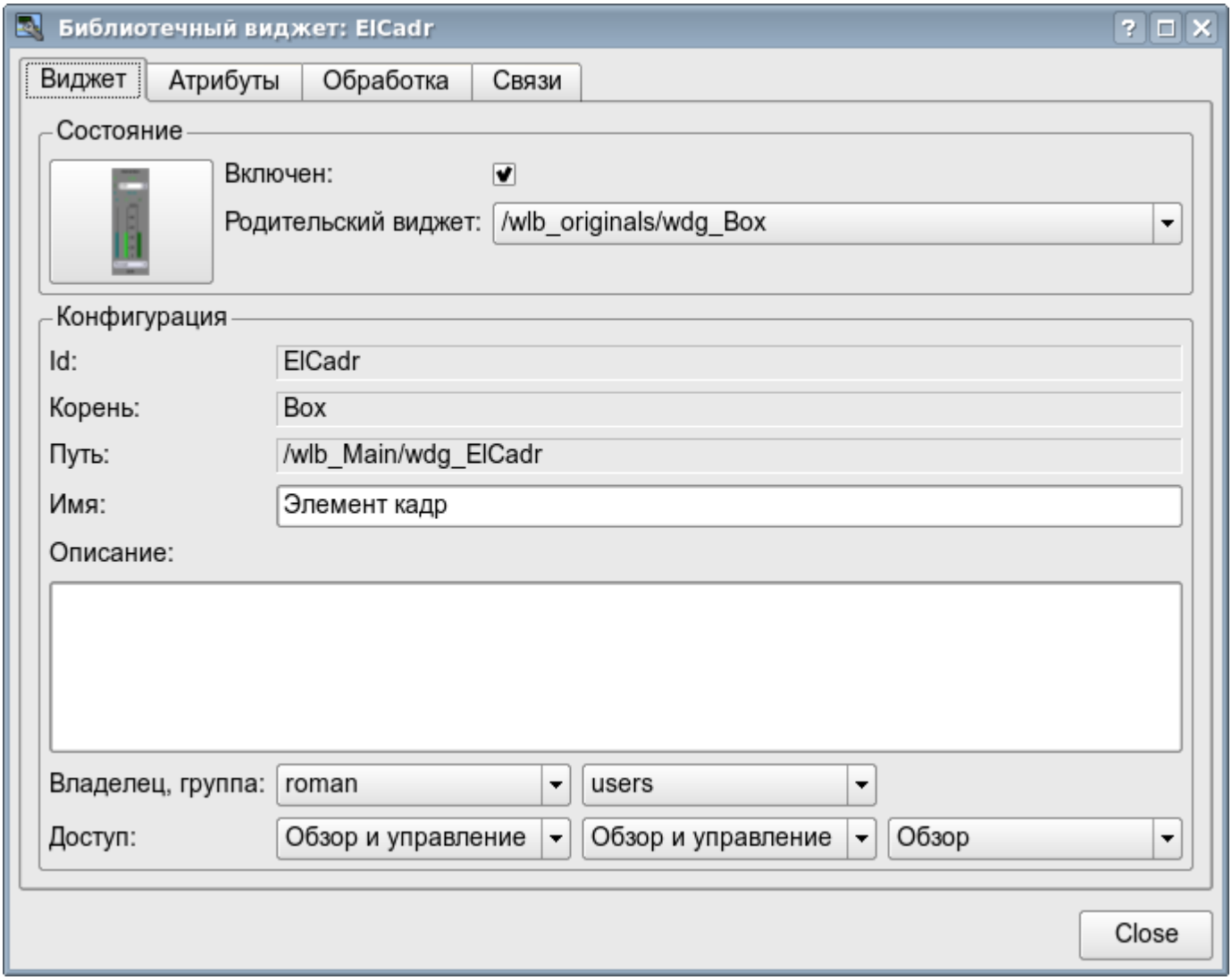


Рис.3. Диалог редактирования свойств визуальных элементов.

С помощью главной вкладки этого диалога можно установить:

- Состояние элемента, а именно: "Включен", родительский виджет.
- Идентификатор, корень, путь, имя и описание элемента.
- Пользователя, группу пользователей элемента и доступ пользователя, группы пользователей и всех остальных.

Диалог редактирования свойств контейнеров визуальных элементов содержит две вкладки: вкладку конфигурации основных параметров (рис.2) и вкладку конфигурации mime-данных контейнеров (рис.4).

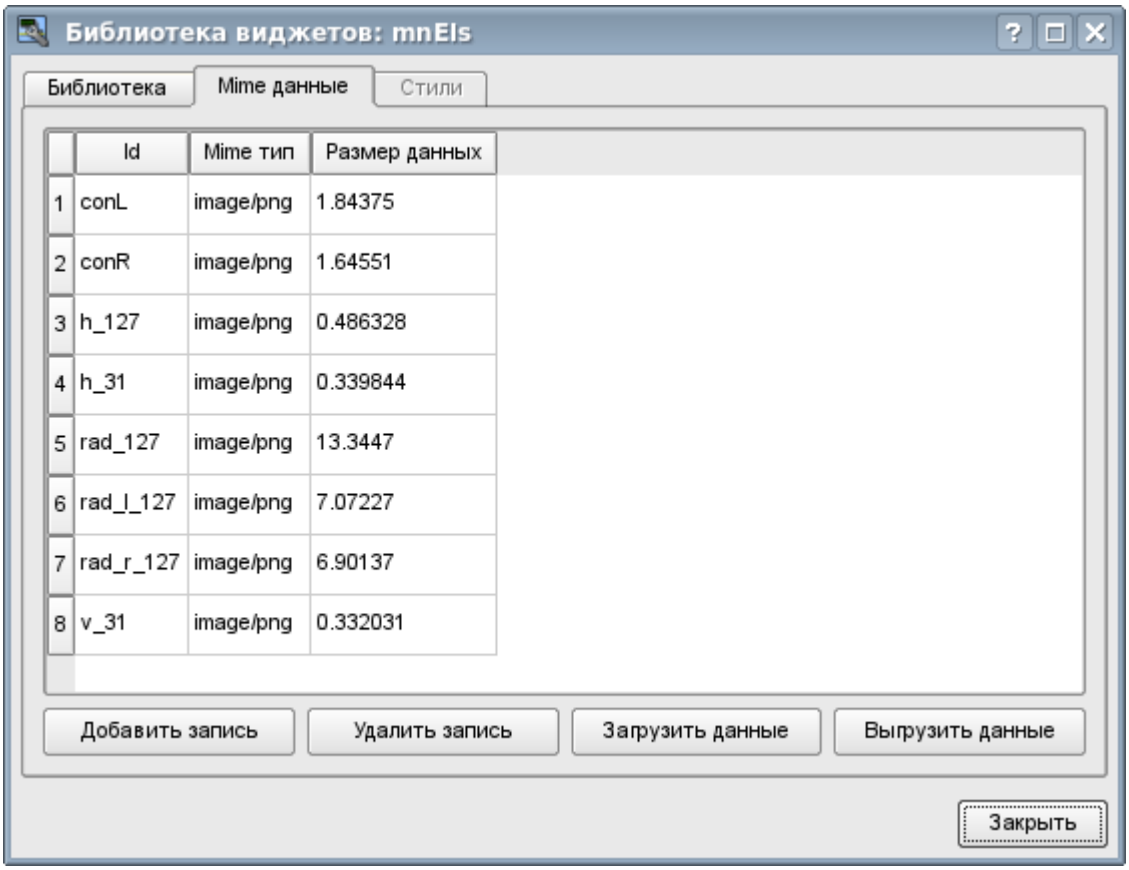


Рис.4. Вкладка редактирования mime-данных контейнера визуальных элементов.

Диалог редактирования свойств визуальных элементов содержит четыре вкладки: вкладку конфигурации основных параметров (рис.2), вкладку атрибутов элемента (рис.5), вкладку обработки элемента (рис.6) и вкладку связей элемента (рис.7). На разных уровнях иерархии визуальных элементов какие-то вкладки могут быть доступны, а какие-то нет.

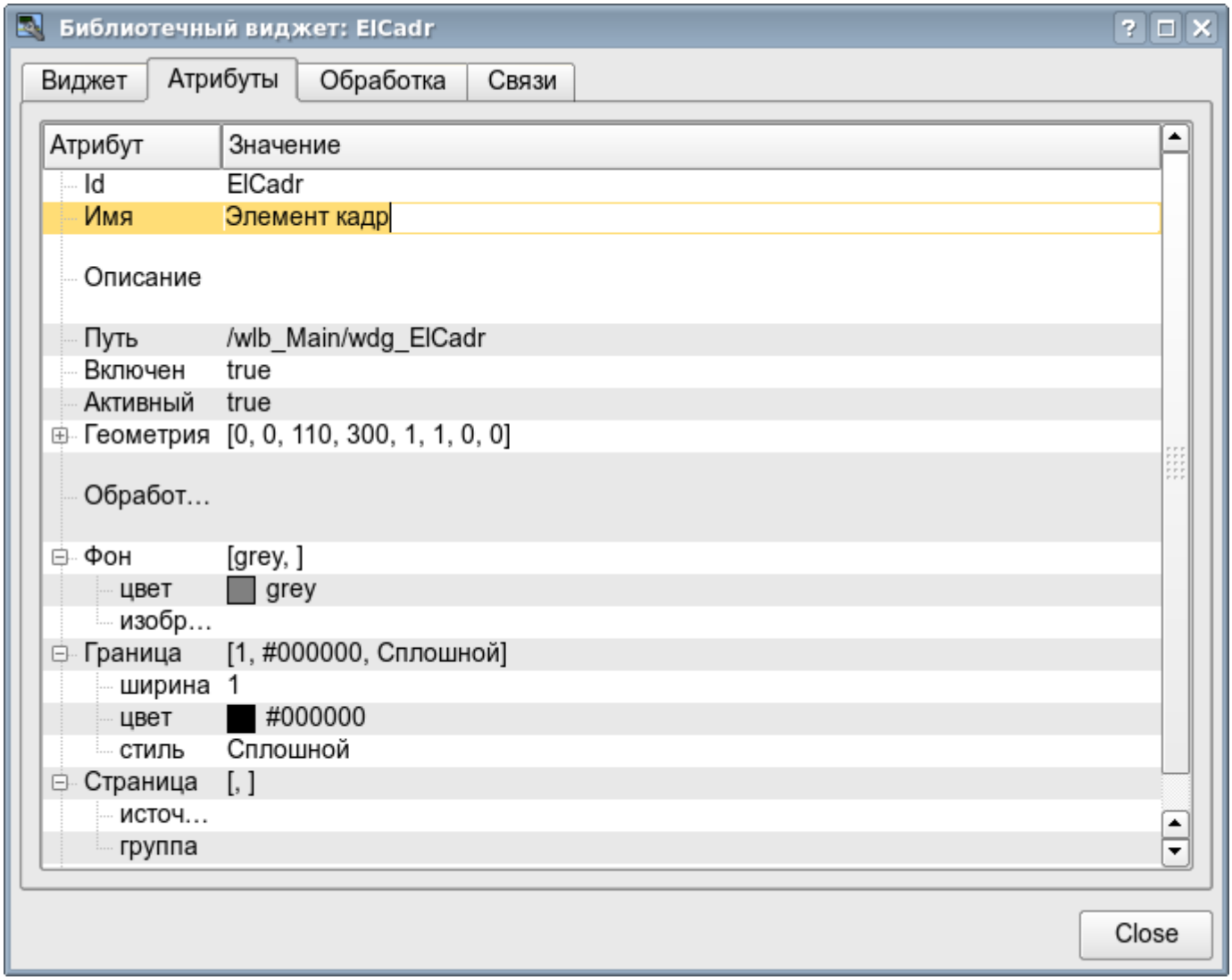


Рис.5. Вкладка атрибутов диалога редактирования свойств визуального элемента.

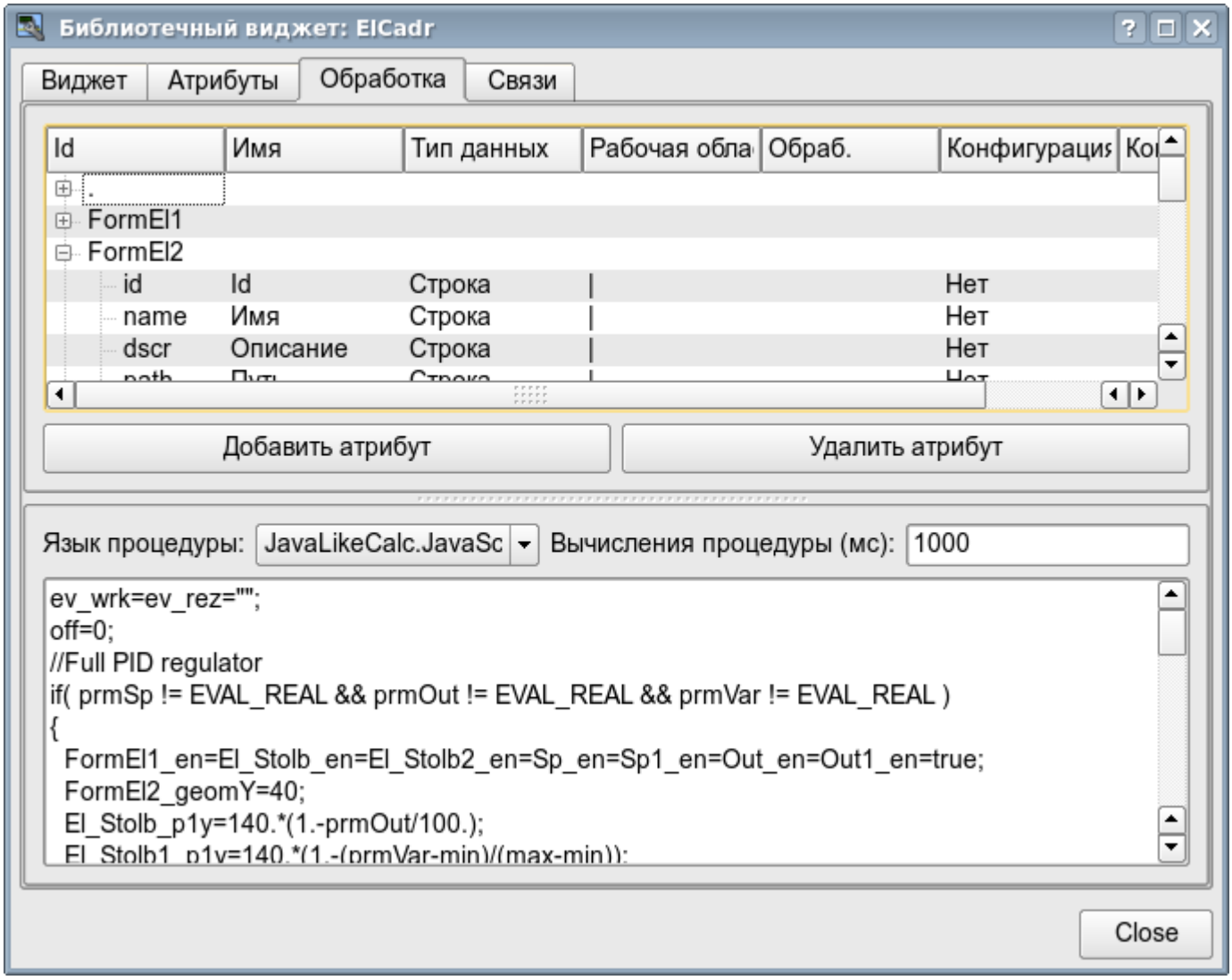


Рис.6. Вкладка обработки диалога редактирования свойств визуального элемента.

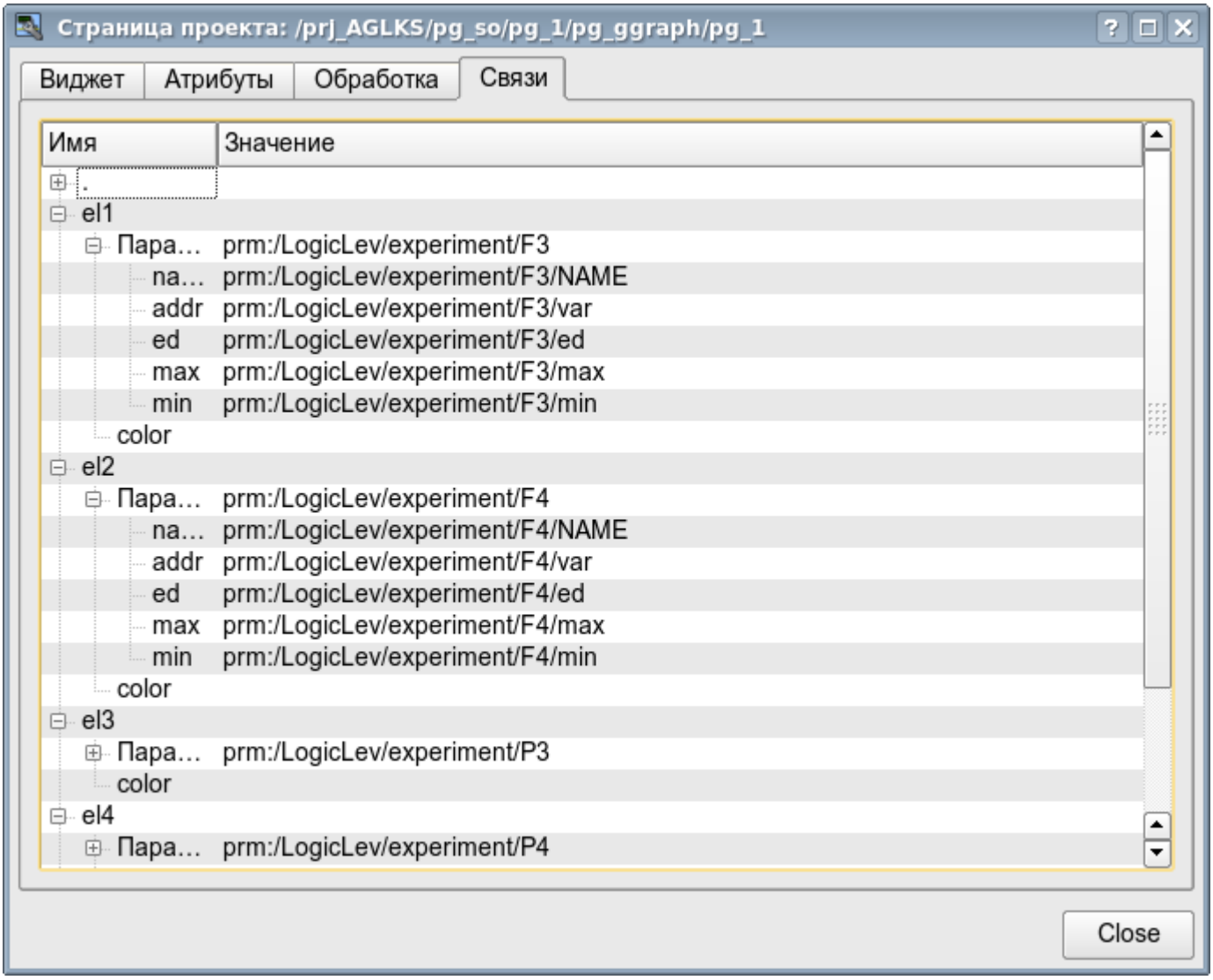


Рис.7. Вкладка связей диалога редактирования свойств визуального элемента.

2.1. Стили

Известно, что человек может иметь индивидуальные особенности в восприятии графической информации. Если эти особенности не учитывать то можно получить неприятие и отторжение пользователя к интерфейсу ВУ. Такое неприятие и отторжение может привести к фатальным ошибкам при управлении ТП, а также травмировать человека постоянной работой с таким интерфейсом. В SCADA системах приняты соглашения, которые регламентируют требования по созданию унифицированного интерфейса ВУ, нормально воспринимаемого большинством людей. При этом практически отсутствует учёт особенностей людей с некоторыми отклонениями.

С целью учесть это обстоятельство и предоставить возможность централизованно и просто изменять визуальные свойства интерфейса проектом предусматривается реализация менеджера стилей интерфейса визуализации.

Пользователем может быть создано множество стилей, каждый из которых будет хранить цветовые, шрифтовые и другие свойства элементов кадра. Простая смена стиля позволит быстро преобразить интерфейс ВУ, а возможность назначения индивидуальной стиля для пользователя позволит учесть его индивидуальные особенности.

Для реализации этой возможности, при создании кадров необходимо для свойств цвета, шрифта и других установить параметр «Конфигурация» (таблицы во вкладке «Обработка») в значение «Из стиля» (рис. 6). А в параметре «Конфигурационный шаблон» указать идентификатор поля стиля. Далее это поле автоматически появится в менеджере стилей и его можно будет там менять. Менеджер стилей доступен на странице конфигурации проекта во вкладке «Стили» (рис. 8). На этой вкладке можно создавать новые стили, удалять старые, изменять поля стиля и удалять ненужные.

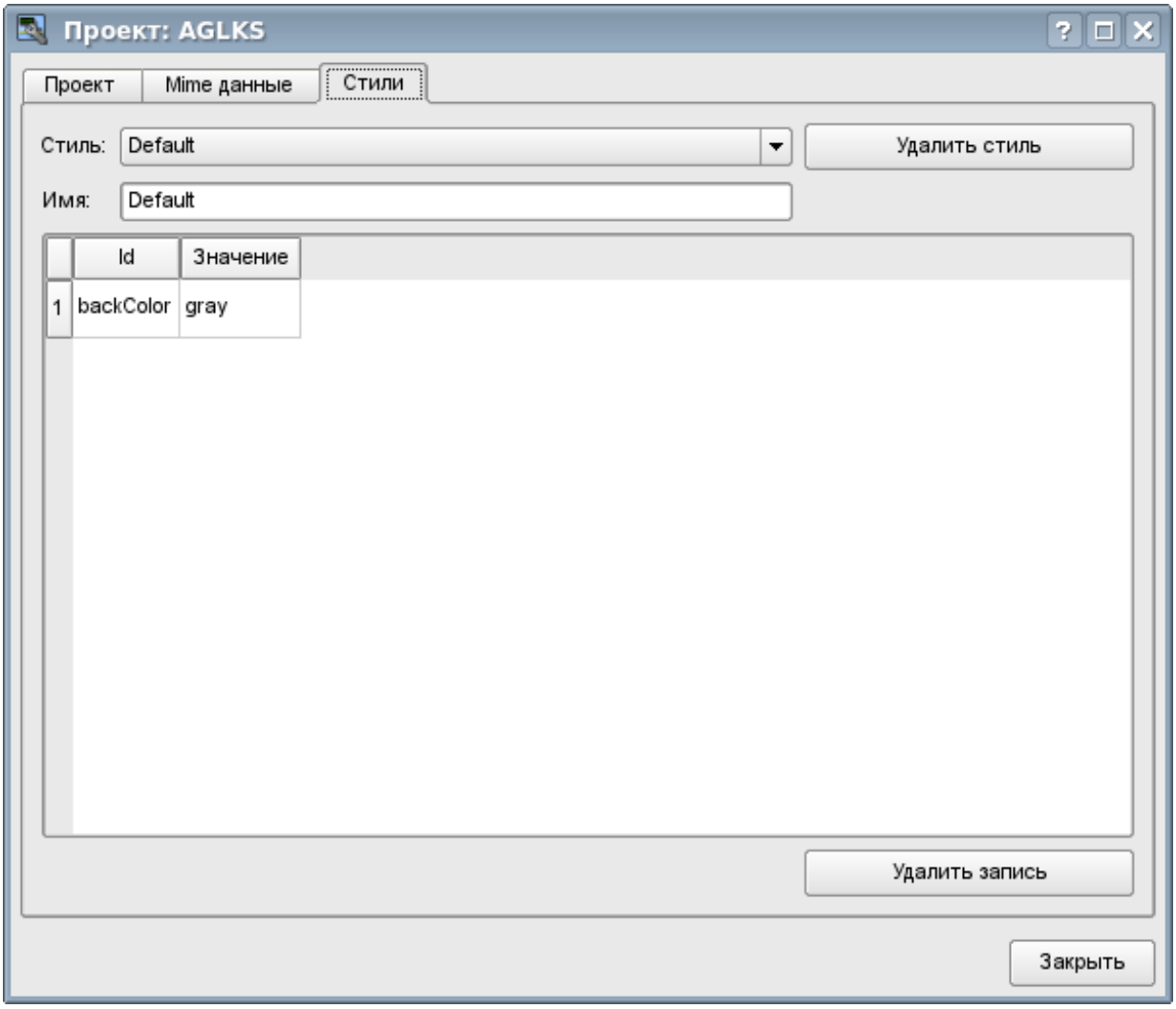


Рис. 8 Вкладка "Стили" страницы конфигурации проекта.

В целом стили доступны, начиная с уровня проектов. На уровне библиотек виджетов можно только определять поля стилей у виджетов. На уровне проекта при выборе стиля включается работа со стилями, что предполагает доступ к полям стилей вместо непосредственных значений атрибутов. Фактически это означает, что при чтении или записи атрибута виджета указанные операции будут осуществляться с соответствующим полем выбранного стиля.

При запуске проекта на исполнения будет использован установленный в проекте стиль. В последствии пользователь может выбрать стиль из перечня доступных. Выбранный пользователем стиль будет сохранён и использован при следующем запуске проекта.

2.2. Связывание с динамикой

Для предоставления актуальных данных в интерфейс визуализации должны использоваться данные подсистемы "Сбор данных (DAQ)". Природа этих данных следующая:

1. параметры, содержащие некоторое количество атрибутов;
2. атрибуты параметра могут предоставлять данные четырёх типов: Логический, Целый, Вещественный и Строковый;
3. атрибуты параметра могут иметь историю (архив);
4. атрибуты параметра могут быть для чтения, записи и с полным доступом.

Учитывая первый пункт, нужно обеспечить возможность группового назначения ссылки. Для этого используем концепцию логического уровня.

В соответствии с пунктом 2 связи обеспечивают прозрачное преобразование типов и не требуют специальной конфигурации.

Для удовлетворения возможности доступа к архивам в соответствии с пунктом 3 связи выполняют проверку типа атрибута и, в случае подключения к "Адресу", в значение помещается адрес связи.

В терминах СВУ динамические связи и конфигурация динамики являются одним процессом, для описания конфигурации которого предусматривается вкладка "Обработка" виджетов (рис.6). Вкладка содержит таблицу конфигурации свойств атрибутов виджета и текст процедуры вычисления виджета.

Кроме полей конфигурации атрибутов в таблице предусматривается колонка "Обработка" для избирательного использования атрибутов виджетов в вычислительной процедуре виджета и колонки "Конфигурация" и "Конфигурационный шаблон" для описания конфигурации связей.

Колонка "Конфигурация" позволяет указать тип связи для атрибута виджета:

- *Постоянная* - во вкладке связей виджета появляется поле указания постоянной, например особого цвета или заголовка для шаблонных кадров;
- *Входная связь* - связь с динамикой только для чтения;
- *Выходная связь* - связь с динамикой только для записи;
- *Полная связь* - полная связь с динамикой (чтение и запись).

Колонка "Конфигурационный шаблон" позволяет описать группы динамических атрибутов. Например, это могут быть разные типы параметров подсистемы "DAQ". Кроме того, при корректном формировании этого поля работает механизм автоматического назначения атрибутов при указании только параметра подсистемы "DAQ", что упрощает и ускоряет процесс конфигурации. Значение этой колонки имеет следующий формат: **<Параметр>|<Идентификатор>**, где:

- *<Параметр>* - группа атрибута;
- *<Идентификатор>* - идентификатор атрибута, именно это значение сопоставляется с атрибутами параметров DAQ при автоматическом связывании после указания групповой связи.

Установка связей может быть нескольких типов, который определяется префиксом:

- *val:* - Прямая загрузка значения через механизм связей. Например, связь: "val:100" загружает в атрибут виджета значение 100. Часто используется в случае отсутствия конечной точки связи с целью прямой установки значения.

- *prm:* - Связь на атрибут параметра или параметр в целом, для группы атрибутов, подсистемы "Сбор данных". Например, связь "prm:/LogicLev/experiment/Pi/var" осуществляет доступ атрибута виджета к атрибуту параметра подсистемы "Сбор данных".
- *wdg:* - Связь на атрибут другого виджета или виджет в целом, для группы атрибутов. Например, связь "wdg:/ses_AGLKS/pg_so/pg_1/pg_ggraph/pg_1/a_bordColor" осуществляет доступ атрибута одного виджета к атрибуту другого. На данный момент этот тип связи не предназначен для установки пользователем вручную, а устанавливается автоматически в режиме динамического связывания!

Обработка связей происходит с периодичностью вычисления виджета в порядке:

- Получение данных входных связей.
- Выполнение вычисления скрипта.
- Передача значений по выходным связям.

На рис. 7 представлена вкладка связей с возможностью группового и индивидуального назначением атрибутов.

При размещении виджета, содержащего конфигурацию связей, в контейнер виджетов все связи исходного виджета добавляются в список результирующих связей контейнера виджетов.

Из вышесказанного видно, что связи устанавливаются пользователем в процессе конфигурации интерфейса. Однако, для предоставления возможности создания кадров общего назначения с функцией предоставления детализированных данных разных источников одного типа необходим механизм динамической установки связей. Такой механизм предусматривается посредством зарезервированного ключевого идентификатора "**<page>**" группы атрибутов связей у кадров общего назначения и динамическое назначение связей с идентификатором "**<page>**" в процессе открытия кадра общего назначения сигналом от другого виджета.

Рассмотрим пример, когда имется кадр общего назначения "Панель контроля графиком" и множество "Графиков" на разных кадрах. "Панель контроля графиком" имеет связи с шаблонами:

- tSek -> "<page>|tSek"
- tSize -> "<page>|tSize"
- trcPer -> "<page>|trcPer"
- valArch -> "<page>|valArch"

При этом каждый виджет "График" имеет атрибуты tSek, tSize, trcPer и valArch. В случае вызова сигнала открытия "Панели контроля графиком" из любого виджета "График" происходит связывания атрибутов "Панели контроля графиком" в соответствии атрибуту, указанному в шаблоне, с атрибутом виджета "График". Как результат, все изменения на "Панели контроля графиком" будут отражаться на графике посредством связи.

В случае наличия у виджета "График" внешних связей на параметры подсистемы "Сбор данных", связи "Панели контроля графиком" будут устанавливаться на внешний источник. Кроме этого, если у "Панели контроля графиком" будут заявлены связи на отсутствующие непосредственно у виджета "График" атрибуты, то будет производится поиск на наличие таких атрибутов у внешнего источника, первого на который установлена прямая связь, выполняя тем самым дополнение недостающих связей.

Для наглядного изображения этого механизма приведена таблица 2.1.

Таблица 2.1. Механизм динамической линковки.

Атрибуты "Панели контроля графиком" (шаблон динамической связи)	**Атрибуты "Графика"**	**Атрибуты внешнего "Параметра"**	**Результирующая связь или значение связующегося атрибута**
tSek (<page>|tSek)	tSek	-	"График".tSek
tSize (<page>|tSize)	tSize	-	"График".tSize
trcPer (<page>|trcPer)	trcPer	-	"График".trcPer
valArch (<page>|valArch)	valArch	-	"График".valArch
var (<page>|var)	var	var	"Параметр".var

Атрибуты "Панели контроля графиком" (шаблон динамической связи)	Атрибуты "Графика"	Атрибуты внешнего "Параметра"	Результирующая связь или значение связующегося атрибута
ed (<page>|ed)	-	ed	"Параметр".ed
max (<page>|max)	-	-	EVAL
min (<page>|min)	-	-	EVAL

3. Исполнение интерфейсов СВУ

Исполнение интерфейса СВУ заключается в запуске нового сеанса проекта или подключении к существующему на уровне движка СВУ. Далее модуль непосредственной визуализации отражает и управляет данными сеанса. Главное окно режима исполнения данного модуля имеет вид, представленный на рис.9.

Реализовано обновление содержимого открытых страниц интерфейса визуализации с периодичностью исполнения сессии проекта. В процессе обновления выполняется:

- запрос списка открытых страниц с признаком модификации страницы у модели и проверка соответствия реально открытых страниц этому списку;
- запрос ветви данных модифицированных страниц;
- обновление содержимого модифицированных страниц и их виджетов в соответствии с полученными измененными данными.

По закрытию "RunTime" окна производится закрытие сессии проекта в движке СВУ.

Механизм запроса только изменённых данных основан на абсолютном счётчике исполнения сессии. При внесении реальных изменений в атрибуты виджетов выполняется запоминание значения этого счётчика, что и позволяет идентифицировать изменённые атрибуты. Такой подход позволяет повысить производительность и уменьшить нагрузку на сетевой обмен в случае доступа к движку СВУ через сеть.

Иерархически модулем предусматривается возможность размещения страниц проекта как на главном окне исполнения (рис.9), так и вкладывая внутрь виджетов контейнеров, а также путем открытия дополнительных окон поверх основного.

При разворачивании главного окна исполнения или переходе в полноэкранный режим выполняется масштабирование содержимого страницы интерфейса СВУ, заполняя всё пространство окна и, позволяя тем самым исполнять проекты, разработанные на одном разрешении экрана, на разных разрешениях.

Главное окно состоит из меню (сверху) строки статуса (снизу) и исполняемого содержимого сеанса между ними. Меню в режиме исполнения позиционируется как инструмент администратора OpenSCADA, содержащий общесистемные функции, и доступно только привилегированным пользователям, состоящим в группе "root". Меню имеет следующую структуру:

- "Файл" - Общие операции.
  - "Печать" - Печать:
    - "Страница" - страницу пользовательского интерфейса;
    - "Диаграмма" - диаграмму на пользовательском интерфейсе;
    - "Документ" - документ на пользовательском интерфейсе.
  - "Экспорт" - Экспорт:
    - "Страница" - страницу пользовательского интерфейса;
    - "Диаграмма" - диаграмму на пользовательском интерфейсе;
    - "Документ" - документ на пользовательском интерфейсе.
  - "Закрыть" - закрыть окно редактора.
  - "Выход" - Выход из системы OpenSCADA.
- "Нарушение" - Квитация нарушений:
  - "Уровень нарушения" - все нарушения;
  - "Световое предупреждение" - световое уведомление;
  - "Предупреждение гудком" - уведомление гудком;

- "Звуковое/речевое предупреждение" - уведомление звуком/речью.
- "Вид" - Параметры отображения сеанса проекта.
  - "Весь экран" - переключатель полноэкранного режима исполнения.
- "Помощь" - Помощь по OpenSCADA и модулю Vision.
  - "Про" - информация про данный модуль.
  - "Про QT" - информация о библиотеке QT, используемой модулем.

С правой стороны строки статуса размещаются индикаторы часов, текущей станции движка СВУ и пользователя, от имени которого исполняется интерфейс СВУ, а также панель с кнопками квитации нарушений, печати и экспорта. По двойному клику на индикаторе пользователя можно пользователя сменить, введя новое имя и пароль пользователя, а по клику на кнопки квитации - сквитировать нарушения полностью или только нужное уведомление. В главное поле строки статуса выводятся различные информационные сообщения и сообщения помощи.

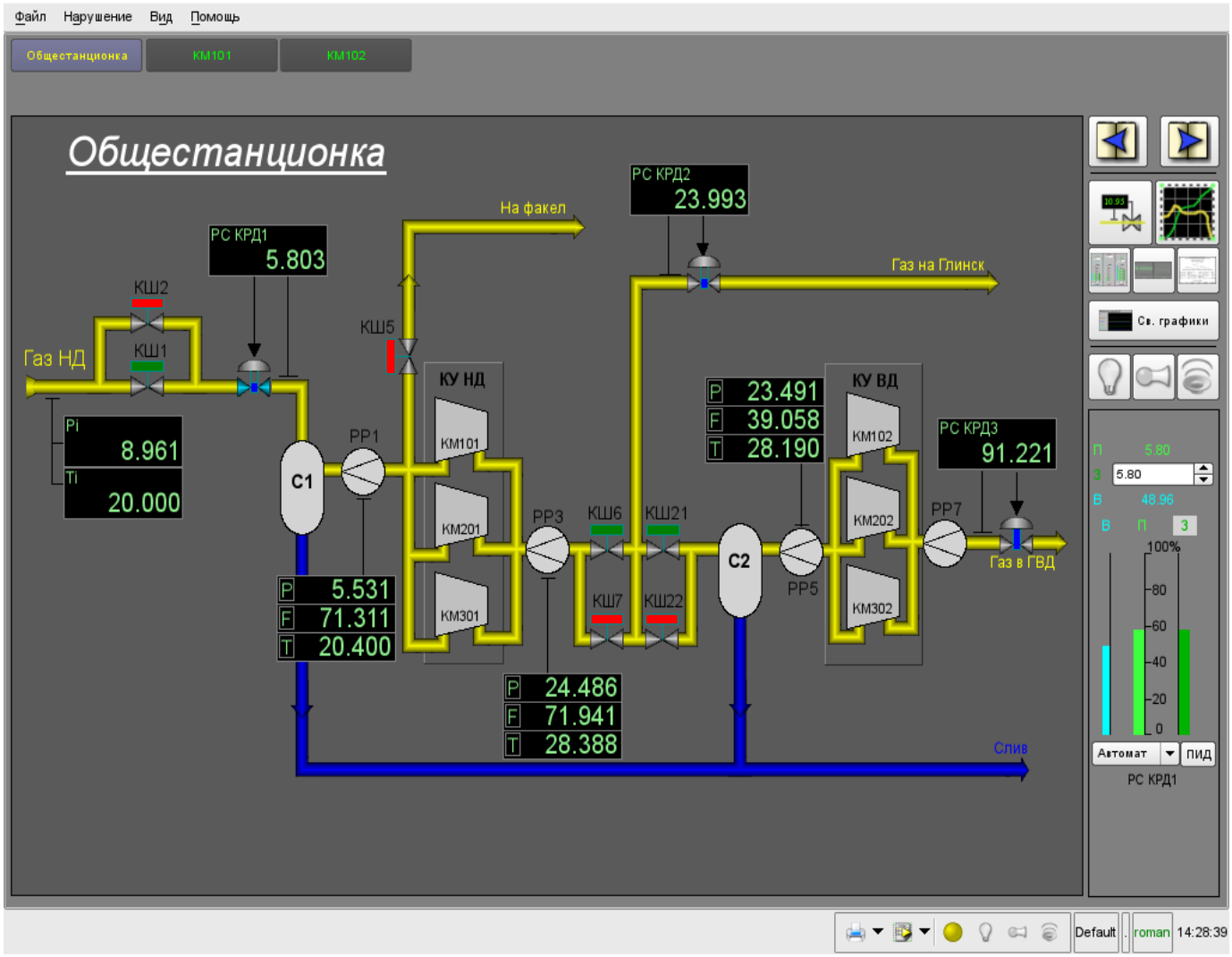


Рис.9. Главное окно режима исполнения.

4. Представление базовых элементов (примитивов)

В данной версии этого модуля реализованы не все образы примитивов заложенные этим проектом. В общем же проектом заложены примитивы:

Id	Наименование	Функция
ElFigure	Элементарные графические фигуры	Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Предусматривается поддержка следующих элементарных фигур: • Линия. • Дуга. • Кривая безье. • Заливка замкнутого пространства. Для всех фигур, содержащихся в виджете, устанавливаются единые свойства толщины, цвета и т.д., но это не исключает возможность указания вышеперечисленных атрибутов конкретно для каждой фигуры отдельно.
FormEl	Элементы формы.	Включает поддержку стандартных компонентов формы: • Редактирование строки. • Редактирование текста. • Флажок. • Кнопка. • Поле выбора из списка. • Список. • Слайдер. • Строка прокрутки.
Text	Текст	Элемент текста(метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием.
Media	Медиа	Элемент отображения растровых и векторных изображений различных форматов, проигрывания анимированных изображений, проигрывание аудио фрагментов и просмотр видео-фрагментов. Возможно в него стоит включить поддержку OpenGL!
Diagram	Диаграмма	Элемент диаграммы с поддержкой возможности отображения нескольких потоков трендов и различных режимов отображения от минималистического до полноэкранного, двухмерного, трёхмерного, кругового и т.д.
Protocol	Протокол	Элемент протокола, визуализатора системных сообщений, с поддержкой различных режимов работы при различных размерах и установках.
Document	Документ	Элемент формирования отчётов, журналов и другой документации на основе указанных данных.
Function	Функция API объектной модели OpenSCADA	Невизуальный на стороне исполнения виджет, позволяющий включать вычислительные функции объектной модели OpenSCADA в СВУ.
Box	Контейнер	Содержит механизм размещения других виджетов с целью формирования новых, более сложных виджетов и страниц конечной визуализации.

Более детально рассмотрим реализацию каждого примитива.

4.1. Примитив элементарная фигура (ElFigure)

Реализована поддержка элементарных фигур: линии, эллиптической дуги, кривой Безье и заливка замкнутых контуров цветом и/или изображением. Для элементарных фигур реализованы следующие операции:

- создание/удаление фигур;
- копирование фигур;
- перемещение и изменение размеров фигур с помощью мыши и клавиатуры;
- возможность связывать элементарные фигуры друг с другом, получая более сложные, для которых доступны все свойства исходных элементарных фигур;
- возможность одновременного перемещения нескольких фигур;
- заливка замкнутого контура цветом и/или изображением;
- генерация событий клавиш мыши в момент клика мышью на залитые контура;
- масштабирование;
- поворот.

На рис. 10 представлена часть экрана с кадром, содержащим элементарные фигуры.

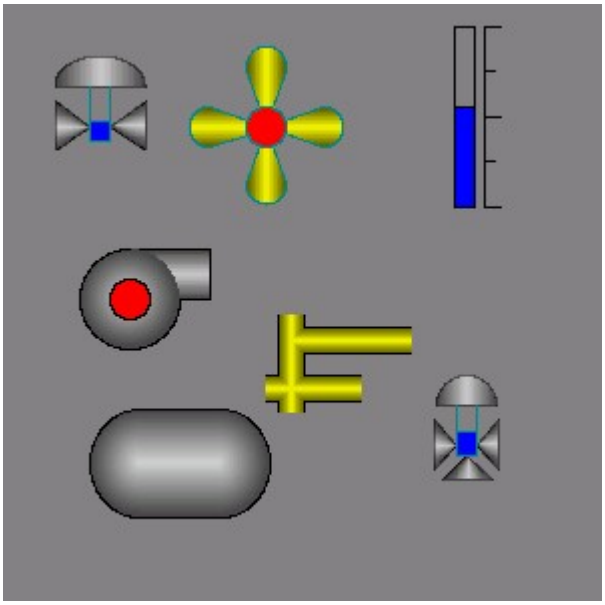


Рис.10 Реализация элементарных фигур в Vision.

Фигуры, лежащие в основе данного виджета, содержат точки(начальная и конечная), которые могут стыковаться с соответствующими точками других фигур, и точки, с помощью которых изменяется геометрия фигуры.

Добавить фигуру можно с помощью мыши:

1. Выбрать желаемую фигуру из контекстного меню.
2. Задать с помощью левой кнопки мыши начальную и конечную точки (для линии при удерживании клавиши SHIFT происходит ортогональная её отрисовка).

Удалить фигуру(ы) можно путём нажатия кнопки "Del", имея выделенную(ые) фигуру(ы).

Скопировать фигуру(ы) можно путём нажатия комбинации клавиш "Ctrl" + "C", имея выделенную(ые) фигуру(ы).

Передвинуть/изменить габариты фигуры можно с помощью мыши или клавиатуры:

1. Выделить фигуру, клацнув по ней левой кнопкой мыши.
2. Перетащить (с помощью мыши или управляющих клавиш) фигуру или одну из её контрольных точек в желаемое место и отпустить кнопку мыши(клавишу).

Существует возможность перемещать несколько выделенных фигур, выбранных при помощи удержания "Ctrl"(эта опция работает при отключенной кнопке Connetcions(Привязки)) либо при помощи выделения мышкой.

Связать фигуры друг с другом можно следующим образом:

1. Нажать кнопку Connections.
2. Выделить одну из фигур и переместить её начальную или конечную точку к желаемой начальной или конечной точке другой фигуры так, чтобы она попадала в появившуюся окружность. Связанные фигуры перемещаются также как и отдельные, общая точка перемещается для всех фигур, к которым она относится(приоритет отдается дуге, две дуги не могут быть соединены непосредственно друг с другом).

Залить замкнутый контур из фигур можно следующим образом:

1. Нажать кнопку Connections.
2. Создать замкнутый контур.
3. Два раза клацнуть мышкой внутри его.

Удалить заливку замкнутого контура можно из контекстного меню, клацнув правой кнопкой манипулятора "мышь" по заливке; разорвав контур заливки; двойным кликом левой кнопки манипулятора "мышь" по уже имеющемуся залитому контуру.

Поворот фигуры осуществляется вокруг центра виджета.

4.2. Примитив текста (Text)

Реализована поддержка элемента текста со свойствами:

- Шрифт со свойствами: типа/класса шрифта, размера, усиления, наклонности, подчёркивания и перечёркивания.
- Цвет текста.
- Ориентация текста.
- Автоматический перенос по словам.
- Выравнивание текста по горизонтали и вертикали со всеми вариантами.
- Отображение фона в виде цвета и/или изображения.
- Отображение бордюра вокруг текста с указанным цветом, шириной и стилем.
- Формирование текста из атрибутов различного типа и свойств.

На рис. 11 представлена часть экрана с кадром, содержащим примеры текста с использованием различных параметров.

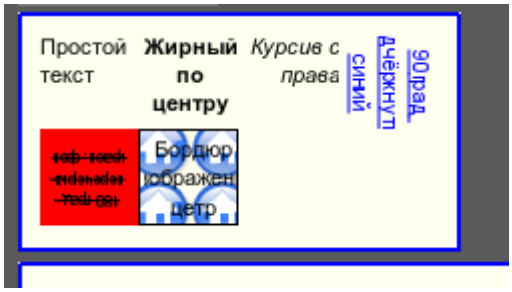


Рис.11. Реализация базового элемента текста в Vision.

4.3. Примитив элементов формы (FormEl)

Реализована поддержка элементов формы на кадрах СВУ. Реализованы заложенные свойства, включая следующие элементы формы:

Редактор строки - Представлено следующими видами: "Текст", "Combo", "Целое", "Вещественное", "Время", "Дата", "Время и Дата". Все виды редактора строки поддерживают подтверждение ввода.

Редактор текста - Представляет редактор плоского текста с подтверждением или отказом от ввода.

Поле флажка - Предоставляет поле бинарного флажка.

Кнопка - Предоставляет кнопку с поддержкой: цвета кнопки, изображения в кнопке и режима фиксации.

Выбор из списка - Предоставляет поле выбора элемента со списка указанных элементов.

Список - Предоставляет поле списка с контролем за текущим элементом.

Слайдер - Элемент слайдера.

Прогрес-бар - Полоска прогресс-бара.

Реализованы режимы: «Включен» и «Активен», а также передача изменений и событий в модель данных СВУ (движок).

На рис. 12 представлена часть экрана с кадром, содержащим вышеперечисленные элементы формы.

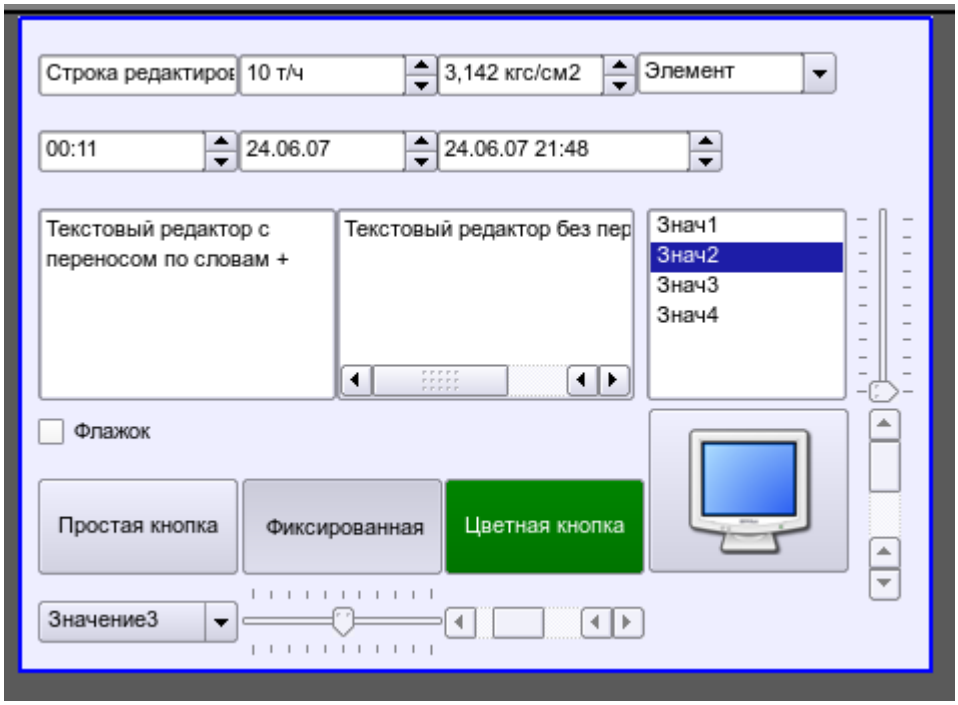


Рис.12. Реализация элементов формы в Vision.

4.4. Примитив отображения медиа-материалов (Media)

Реализована поддержка элемента отображения медиа-материалов со свойствами:

- Указания источника медиа данных (изображения или видео-материала).
- Просмотра изображений большинства известных форматов с возможностью их вписывания в размер виджета.
- Проигрывания простых анимированных форматов изображений и видео с возможностью управления скоростью проигрывания.
- Отображение фона в виде цвета и/или изображения.
- Отображение бордюра вокруг текста,с указанным цветом, шириной и стилем.
- Формирования активных областей и генерация событий при их активации.

На рис. 13 представлена часть экрана с кадром, содержащим примеры просмотра/проигрывания медиа-данных.

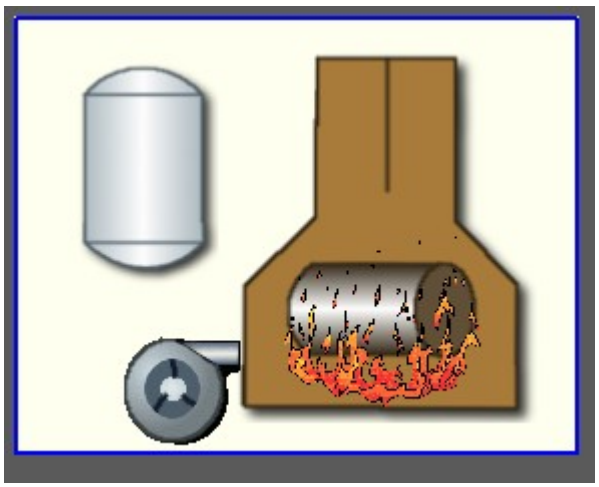


Рис.13. Реализация базового элемента отображения медиа-материалов в Vision.

4.5. Примитив построения диаграм/графиков (Diagram)

Реализована поддержка элемента построения диаграм/трендов со свойствами:

- Построение графиков/трендов:
  - Построения графика для: архивных данных, текущих данных и формирования промежуточного буфера отображения для параметров без архива.
  - Построение как одиночных графиков со значением параметра по оси ординат, так и сводных графиков, включающих до 10 параметров, с процентной шкалой.
  - Возможность адаптации графика параметра к значениям данных, подгон шкалы.
  - Широкий диапазон масштабирования и адаптации горизонтальной шкалы с автоматическим усреднением на уровне сервера и самого примитива.
  - Возможность отображение размерной сетки и маркеров по горизонтали и вертикали с адаптацией к диапазону отображения.
  - Поддержка активного режима с курсором и получением значений под курсором.

На рис. 14 представлена часть экрана с кадром, содержащим примеры диаграммы-тренда.

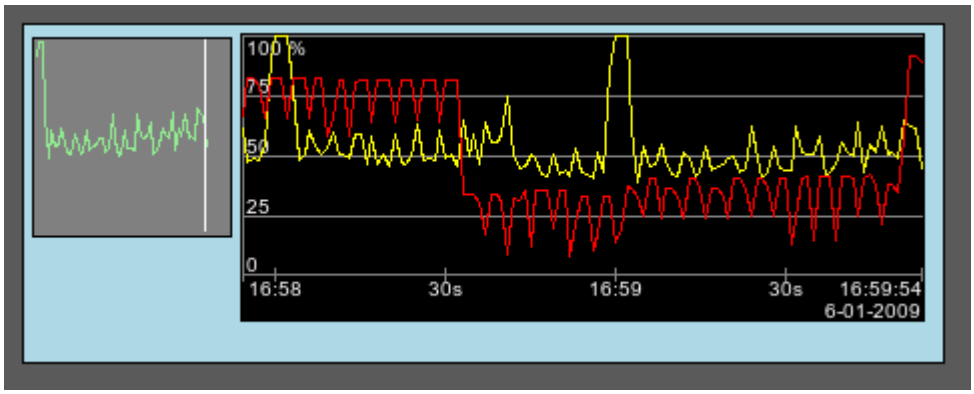


Рис.14. Реализация базового элемента отображения диаграммы-тренда в Vision.

4.6. Примитив формирования протокола (*Protocol*)

Реализована поддержка элемента формирования протокола со свойствами:

- Формирование протокола из архива сообщений за указанное время и глубину.
- Запрос данных из указанных архиваторов сообщений.
- Выборка данных из архивов по уровню важности и шаблону категории сообщений.
- Поддержка режима слежения за появлением сообщений в архиве сообщений.

На рис. 15 представлена часть экрана с кадром, содержащим пример протокола.

	Время	Уровень	Категория	Сообщение
1	06.01.2009 17:04:17	3	/DemoStation/sub_UI/mod_Vision/	Error open: /dev/input/by-path/platform-pcspkr-event-spkr
2	06.01.2009 17:03:39	1	/DemoStation/sub_UI/mod_VCAEn…	Старт модуля.
3	06.01.2009 17:03:39	4	/DemoStation/sub_Archive/va_CP…	Archivator <DBArch.1s> error or no started.
4	06.01.2009 17:03:33	1	/DemoStation/sub_Archive/	Пуск подсистемы.
5	06.01.2009 17:03:33	1	/DemoStation/sub_DAQ/mod_Logi…	Запуск контроллера!
6	06.01.2009 17:03:33	1	/DemoStation/sub_DAQ/mod_Syst…	Запуск контроллера!
7	06.01.2009 17:03:33	1	/DemoStation/sub_DAQ/mod_Bloc…	Запуск контроллера!

Рис.15 Реализация базового элемента отображения протоколов в Vision.

4.7. Примитив формирования отчётной документации (*Document*)

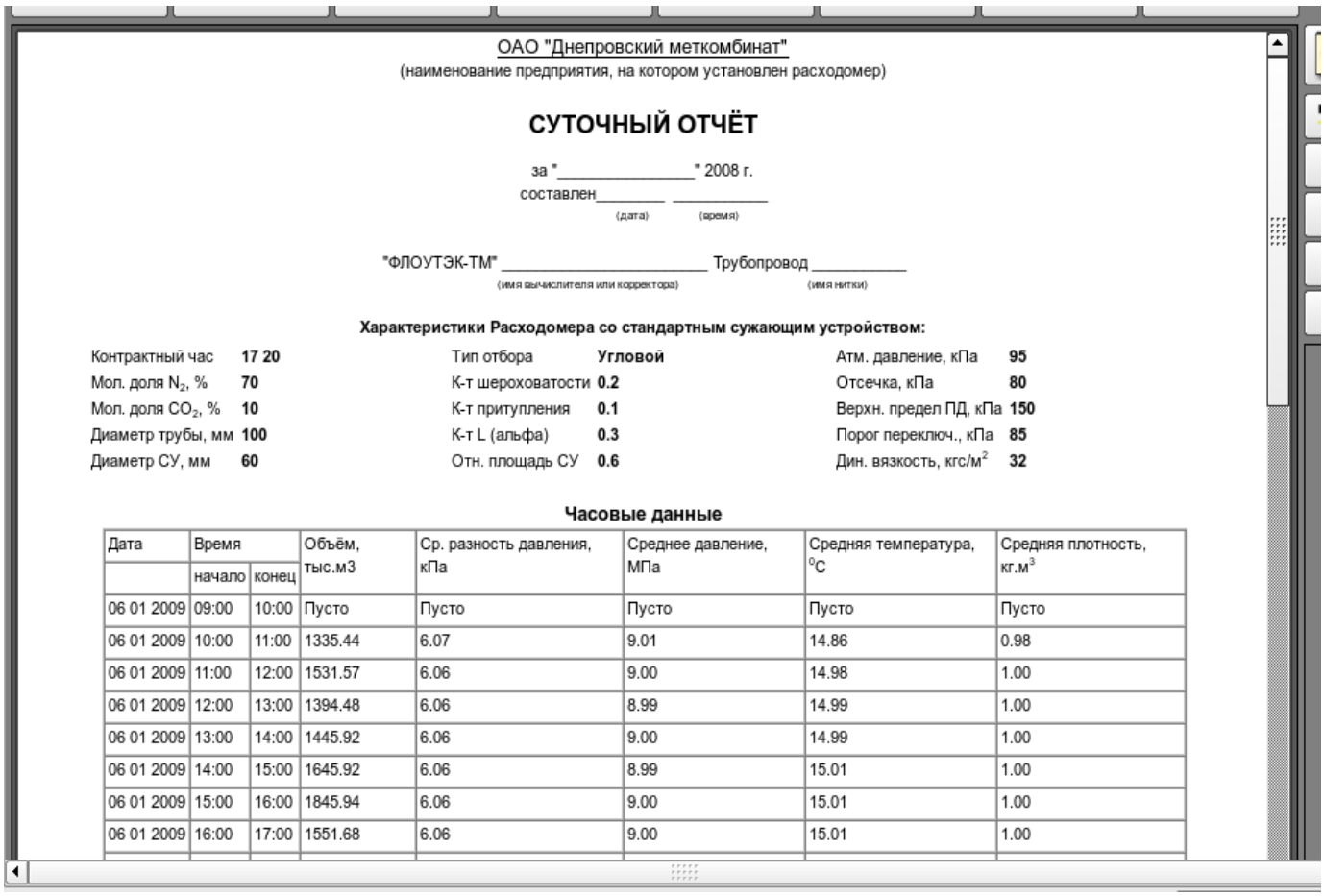
Реализована поддержка элемента формирования отчётной документации со свойствами:

- Гибкое формирование структуры документа на основе языка гипертекстовой разметки. Это предоставляет поддержку широких возможностей форматирования документов.
- Формирование документов по команде или по графику. Необходимо для формирования отчётной документации в архив с последующим просмотром архива.
- Формирование документа в режиме реального времени. Для формирования документов полностью динамически и на основе архивов за указанное время.
- Использование атрибутов виджета для передачи значений и адресов на архивы в документ. Позволяет использовать виджет документа как шаблон при формировании отчётов с другими входными данными.

В основе любого документа лежит XHTML-шаблон. XHTML-шаблон это тег 'body' WEB-страницы, содержащий статику документа в стандарте XHTML 1.0, и элементы исполняемых инструкций на одном из языков пользовательского программирования OpenSCADA в виде <?dp <procedure> ?>. Результирующий документ формируется путём исполнения процедур и вставки их результата в документ.

Источником значений исполняемых инструкций являются атрибуты виджета этого примитива, а также все механизмы языка пользовательского программирования. Атрибуты могут добавляться пользователем и линковаться на реальные атрибуты параметров или-же являться автономными, значения которых будут формироваться в скрипте виджета. В случае со слинкованными атрибутами могут извлекаться значения из истории, архива.

На рис. 16 представлен кадр, содержащий пример документа.



ОАО "Днепровский меткомбинат"

(наименование предприятия, на котором установлен расходомер)

СУТОЧНЫЙ ОТЧЁТ

за "_____" 2008 г.

составлен_____ _____

(дата) (время)

"ФЛОУТЭК-ТМ" _____ Трубопровод _____

(имя вычислителя или корректора) (имя нитки)

Характеристики Расходомера со стандартным сужающим устройством:

Контрактный час	**17 20**	Тип отбора	**Угловой**	Атм. давление, кПа	**95**
Мол. доля N_2, %	**70**	К-т шероховатости	**0.2**	Отсечка, кПа	**80**
Мол. доля CO_2, %	**10**	К-т притупления	**0.1**	Верхн. предел ПД, кПа	**150**
Диаметр трубы, мм	**100**	К-т L (альфа)	**0.3**	Порог переключ., кПа	**85**
Диаметр СУ, мм	**60**	Отн. площадь СУ	**0.6**	Дин. вязкость, кгс/м^2	**32**

Часовые данные

Дата	Время		Объём, тыс.м3	Ср. разность давления, кПа	Среднее давление, МПа	Средняя температура, ^{0}C	Средняя плотность, кг.м^3
	начало	конец					
06 01 2009	09:00	10:00	Пусто	Пусто	Пусто	Пусто	Пусто
06 01 2009	10:00	11:00	1335.44	6.07	9.01	14.86	0.98
06 01 2009	11:00	12:00	1531.57	6.06	9.00	14.98	1.00
06 01 2009	12:00	13:00	1394.48	6.06	8.99	14.99	1.00
06 01 2009	13:00	14:00	1445.92	6.06	9.00	14.99	1.00
06 01 2009	14:00	15:00	1645.92	6.06	8.99	15.01	1.00
06 01 2009	15:00	16:00	1845.94	6.06	9.00	15.01	1.00
06 01 2009	16:00	17:00	1551.68	6.06	9.00	15.01	1.00

Рис.16 Реализация базового элемента отображения отчётной документации в Vision.

4.8. Примитив контейнера (Box)

Реализована поддержка примитива контейнера, по совместительству выполняющего роль страниц проектов. Данный примитив является единственным элементом-контейнером, который может включать в себя ссылки на кадры из библиотеки, формируя тем самым пользовательские элементы нужной конфигурации. Примитив реализует предусмотренные проектом свойства. Перечислим по пунктам свойства данного примитива:

Контейнер - Позволяет формировать нужные объекты путём группировки базовых в рамках данного примитива.

Страница - Элементы, построенные на данном примитиве, могут выполнять роль страницы пользовательского интерфейса.

Контейнер страниц - Свойство замещения собственного содержимого другой страницей в процессе исполнения. Используется для формирования фреймов на страницах пользовательского интерфейса. Например, главная страница традиционной SCADA системы с объектами сигнализации строится именно таким образом.

Фон - Поддерживает возможность указания фона в виде цвета или изображения.

Бордюр - Поддерживает возможность изображения бордюра с указанным цветом, толщиной и стилем.

Пример редактирования кадра, основанного на данном примитиве, приведен на рис. 1, а на рис. 9 изображена страница, содержащая контейнер страниц, построенный на основе данного примитива.

5. Общая конфигурация модуля

Для настройки собственного поведения в неочевидных ситуациях модулем предоставляется возможность настройки отдельных параметров посредством интерфейса управления OpenSCADA (рис. 17). Таковыми параметрами являются:

- Начальный пользователь конфигуратора - указывает, от имени какого пользователя открывать конфигуратор без запроса пароля.
- Перечень проектов для запуска их автоматического исполнения с запуском модуля. Для предоставления возможности указания открытия окна исполнения проекта на нужном дисплее много дисплейных систем предусмотрен формат записи имени проекта "PrjName-1", где 1 - номер целевого дисплея.
- Имя удалённой OpenSCADA станции с движком визуализации СВУ.
- Ссылка на страницу конфигурации перечня внешних OpenSCADA станций.

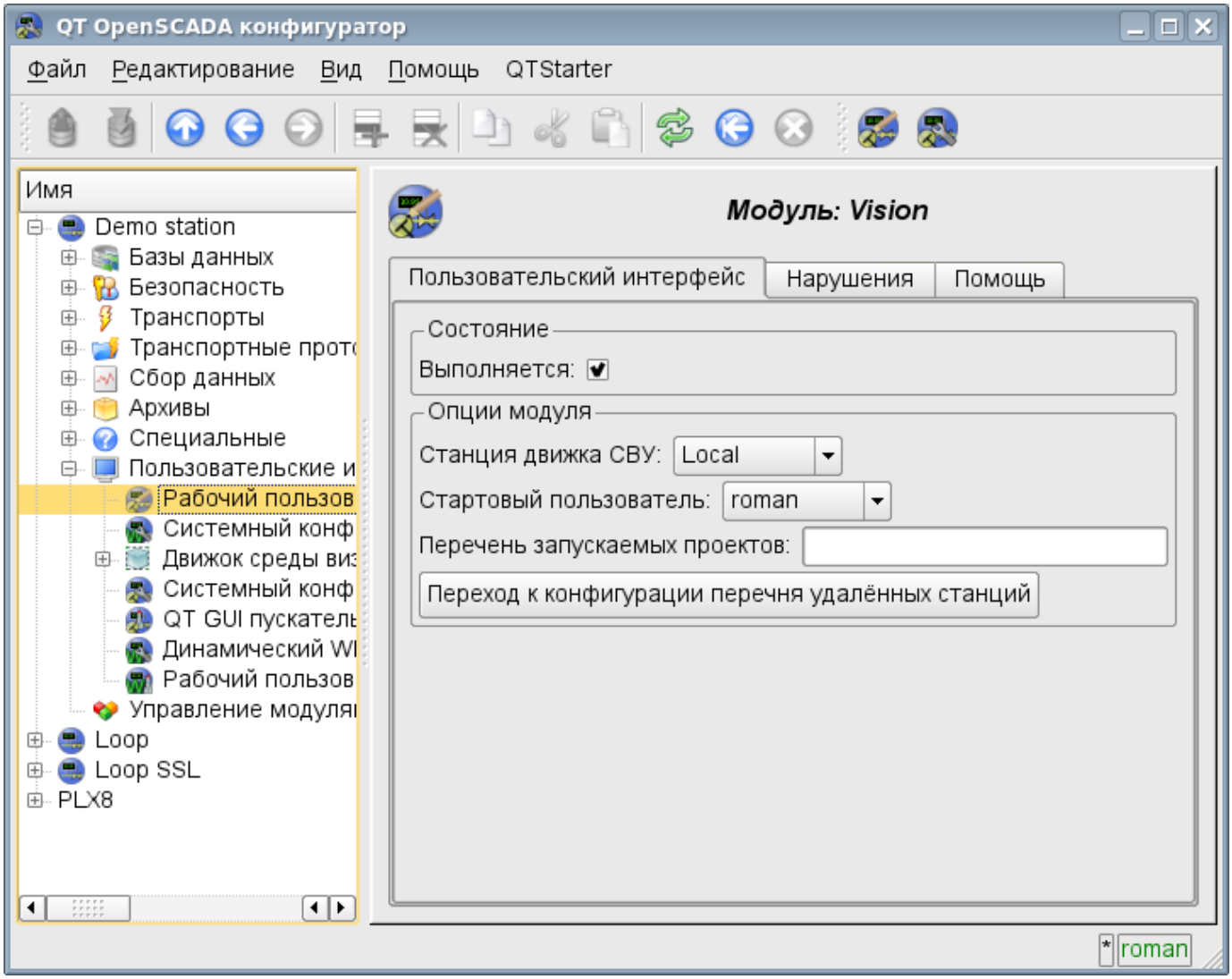


Рис.17. Страница конфигурации модуля.

Модуль подсистемы “Пользовательские интерфейсы” <WebVision>

Модуль:	WebVision
Имя:	Рабочий пользовательский интерфейс (WEB)
Тип:	Пользовательские интерфейсы
Источник:	ui_WebVision.so
Версия:	0.9.0
Автор:	Роман Савоченко
Разработчик:	Роман Савоченко, Максим Лысенко, Ксения Яшина
Описание:	Web визуальный рабочий пользовательский интерфейс для исполнения проектов среды визуализации и управления (СВУ).
Лицензия:	GPL

Модуль WebVision предоставляет механизм конечной визуализации среды визуализации и управления (СВУ) в систему OpenSCADA. Модуль основан на WEB технологиях (XHTML, JavaScript, CSS, AJAX). В своей работе модуль использует данные движка СВУ (модуль VCAEngine).

Среда визуализации и управления (СВУ) является неотъемлемой составляющей SCADA системы. Она применяется на клиентских станциях с целью доступного предоставления информации об объекте управления и выдачи управляющих воздействий на объект. В различных практических ситуациях и условиях могут применяться СВУ, построенные на различных принципах визуализации. Например, это могут быть библиотеки виджетов QT, GTK+, wxWidgets или гипертекстовые механизмы на основе технологий HTML, XHTML, XML, CSS и JavaScript или сторонние приложения визуализации, реализованные на различных языках программирования Java, Python и т.д. Любой из этих принципов имеет свои преимущества и недостатки, комбинация которых может стать непреодолимым препятствием в возможности использования СВУ в том или ином практическом случае. Например, технологии вроде библиотеки QT позволяют создавать высокореактивные СВУ, что несомненно важно для станций оператора управления технологическим процессом (ТП). Однако необходимость инсталляции данного клиентского ПО в отдельных ситуациях может сделать использование его невозможным. С другой стороны Web-технологии не требуют инсталляции на клиентские системы и являются предельно многоплатформенными (достаточно создать ссылку на Web-сервер в любом Web-браузере), что наиболее важно для различных инженерных и административных станций. С другой стороны реактивность и надёжность таких интерфейсов ниже, что практически исключает их использования на станциях оператора ТП.

Система OpenSCADA имеет гибкую архитектуру, которая позволяет создавать внешние интерфейсы, в том числе и пользовательские, на любой основе и на любой вкус. Например, среда конфигурации системы OpenSCADA доступна как на QT библиотеке, так и на Web-основе.

В такой ситуации независимое создание реализаций СВУ на различной основе может повлечь за собой невозможность использования данных конфигурации одной СВУ в другой. Что неудобно и ограничено с пользовательской стороны, а также накладно в плане реализации и последующей поддержки. С целью избежать этих проблем, а также создать в кратчайшие сроки полный спектр различных типов СВУ, основан проект создания концепции СВУ. Результатом этого проекта и стал данный модуль непосредственной визуализации на основе WEB технологий, модуль непосредственной визуализации Vision и движок СВУ VCAEngine.

1. Назначение

Данный модуль непосредственной визуализации СВУ предназначен только для исполнения интерфейсов СВУ в среде WEB-технологий.

Интерфейс пользователя формируется в WEB-браузере путём обращения к WEB-серверу и получения от него XHTML-документа по протоколу HTTP. В данном случае в роли WEB-сервера выступает система OpenSCADA, которая поддерживает стандартные коммуникационные механизмы TCP-сетей (модуль Transport.Sockets), протокол передачи гипертекста (модуль Protocol.HTTP), а также шифрование трафика между браузером и сервером (Transport.SSL). Исходя из этого, для получения доступа к интерфейсу пользователя, предоставляемого этим модулем, необходимо в OpenSCADA настроить транспорт (Transport.Sockets или Transport.SSL) в связке с протоколом HTTP (Protocol.HTTP). В поставке с системой OpenSCADA идут конфигурационные файлы, содержащие настройки Transport.Sockets для портов 10002 и 10004. Следовательно, интерфейс модуля в конфигурации OpenSCADA по умолчанию будет доступен по URL: http://localhost:10002 или http://localhost:10004.

В финальной версии этого модуля СВУ, построенная на основе данного модуля, обеспечит:

- три уровня сложности в формировании интерфейса визуализации, позволяющие органично осваивать и применять инструментарий по методике от простого к сложному:
  - формирование из шаблонных кадров, путём назначения динамики (без графической конфигурации);
  - графическое формирование новых кадров путём использования готовых элементов визуализации из библиотеки (мнемосхемы);
  - формирование новых кадров, шаблонных кадров и элементов отображение в библиотеки.
- построение интерфейсов визуализации различной сложности, начиная от простых плоских интерфейсов мониторинга и заканчивая полноценными иерархическими интерфейсами, используемыми в SCADA системах;
- предоставление различных способов формирования и конфигурации пользовательского интерфейса, основанных на различных интерфейсах графического представления (QT, Web, Java ...) или же посредством стандартного интерфейса управления системой OpenSCADA;
- смену динамики в процессе исполнения;
- построение новых шаблонных кадров на уровне пользователя и формирование специализированных под область применения библиотек кадров (например, включение кадров параметров, графиков и других элементов, с увязкой их друг с другом) в соответствии с теорией вторичного использования и накопления;
- построение новых пользовательских элементов визуализации и формирование специализированных под область применения библиотек кадров в соответствии с теорией повторного использования и накопления;
- описание логики новых шаблонных кадров и пользовательских элементов визуализации как простыми связями, так и лаконичным, полноценным языком пользовательского программирования;
- возможность включения в пользовательские элементы визуализации функций (или кадров вычисления функций) объектной модели OpenSCADA, практически связывая представление с алгоритмом вычисления (например, визуализируя библиотеку моделей аппаратов ТП для последующего визуального построения моделей ТП);
- разделение данных пользовательских интерфейсов и интерфейсов представления этих данных, позволяющее строить интерфейс пользователя в одной среде, а исполнять во многих других (QT, Web, Java ...);
- возможность подключения к исполняющемуся интерфейсу для наблюдения и коррекции действий (например, при обучении операторов и контроля в реальном времени за его действиями);
- визуальное построение различных схем с наложением логических связей и последующим централизованным исполнением в фоне (визуальное построение и исполнение математических моделей, логических схем, релейных схем и иных процедур);
- предоставление функций объектного API в систему OpenSCADA, может использоваться

для управления свойствами интерфейса визуализации из пользовательских процедур;
- построение серверов кадров, элементов визуализации и проектов интерфейсов визуализации с возможностью обслуживания множественных клиентских соединений;
- простая организация клиентских станций на различной основе (QT, Web, Java ...) с подключением к центральному серверу;
- полноценный механизм разделения полномочий между пользователями, позволяющий создавать и исполнять проекты с различными правами доступа к его компонентам;
- гибкое формирование правил сигнализаций и уведомления с учётом и поддержкой различных способов уведомления;
- поддержка пользовательского формирования палитры и шрифтовых предпочтений для интерфейса визуализации;
- поддержка пользовательского формирования карт событий под различное оборудование управления и пользовательские предпочтения;
- поддержка профилей пользователей, позволяющая определять различные свойства интерфейса визуализации (цветовая гамма, шрифтовые особенности, предпочтительные карты событий);
- гибкое хранение и распространение библиотек виджетов, кадров и проектов интерфейсов визуализации в БД, поддерживаемых системой OpenSCADA; практически пользователю нужно только зарегистрировать полученную БД с данными.

2. Исполнение интерфейсов СВУ

Исполнение интерфейса СВУ заключается в запуске нового сеанса проекта или подключения к существующему на уровне движка СВУ (рис.2). Перед запросом на подключение к сеансу выполняется запрос на аутентификацию пользователя. Далее модуль непосредственной визуализации отражает и управляет данными сеанса. Главное окно режима исполнения данного модуля имеет вид, представленный на рис.3.

Интерфейс окна исполнения полностью строится динамически скриптом JavaScript, исходя из содержимого сеанса проекта путём прямых XML запросов к серверу.

Реализовано обновление содержимого открытых страниц интерфейса визуализации с периодичностью одна секунда. В процессе обновления выполняется:
- запрос списка открытых страниц, с признаком модификации страницы, у модели и проверка соответствия реально открытых страниц этому списку;
- запрос ветви данных модифицированных страниц;
- обновление содержимого модифицированных страниц и их виджетов в соответствии с полученными измененными данными.

Механизм запроса только изменённых данных основан на абсолютном счётчике исполнения сессии. При внесении реальных изменений в атрибуты виджетов выполняется запоминание значения этого счётчика, что и позволяет идентифицировать изменённые атрибуты. Такой подход позволяет повысить производительность и уменьшить нагрузку на трафик, в случае доступа к движку СВУ через сеть.

Иерархически модулем предусматривается возможность размещения страниц проекта как на главном окне исполнения WEB-браузера (рис.3), так и вкладывая внутрь виджетов контейнеров.

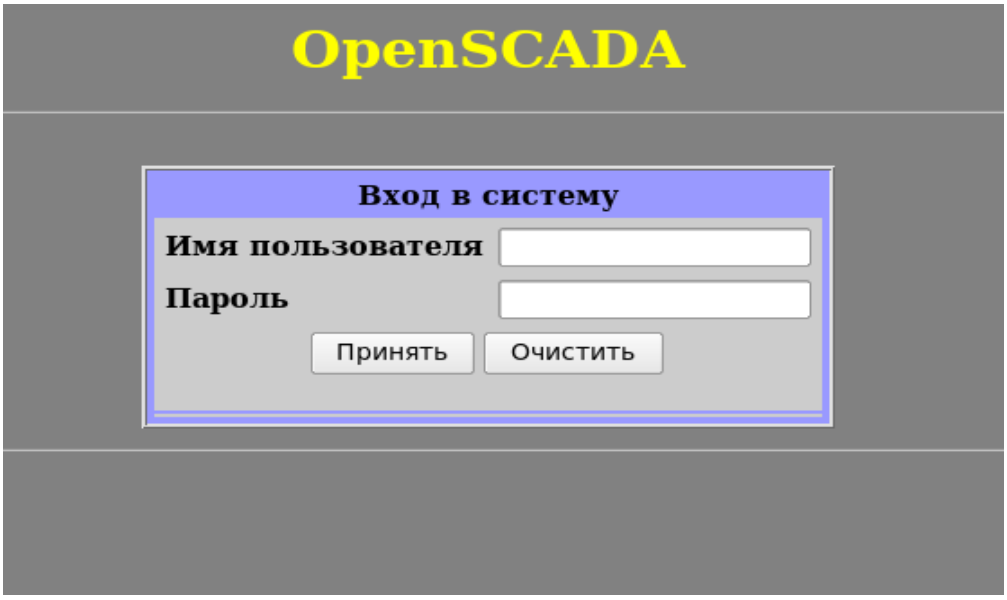


Рис.1. Страница аутентификации.

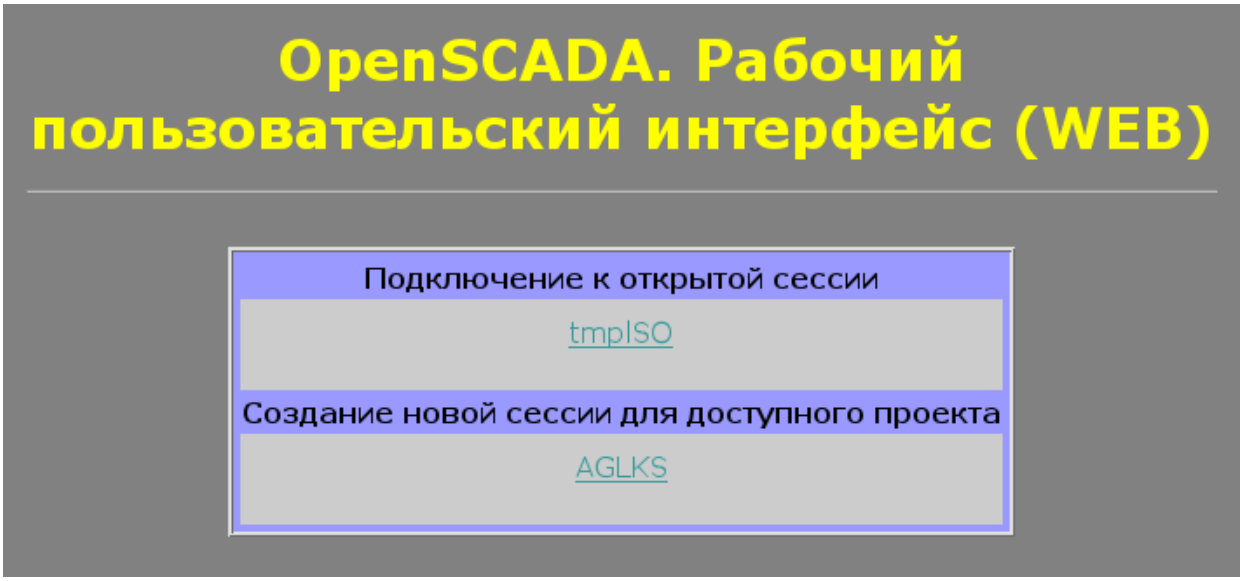


Рис.2. Подключение или создание нового сеанса исполнения проекта СВУ.

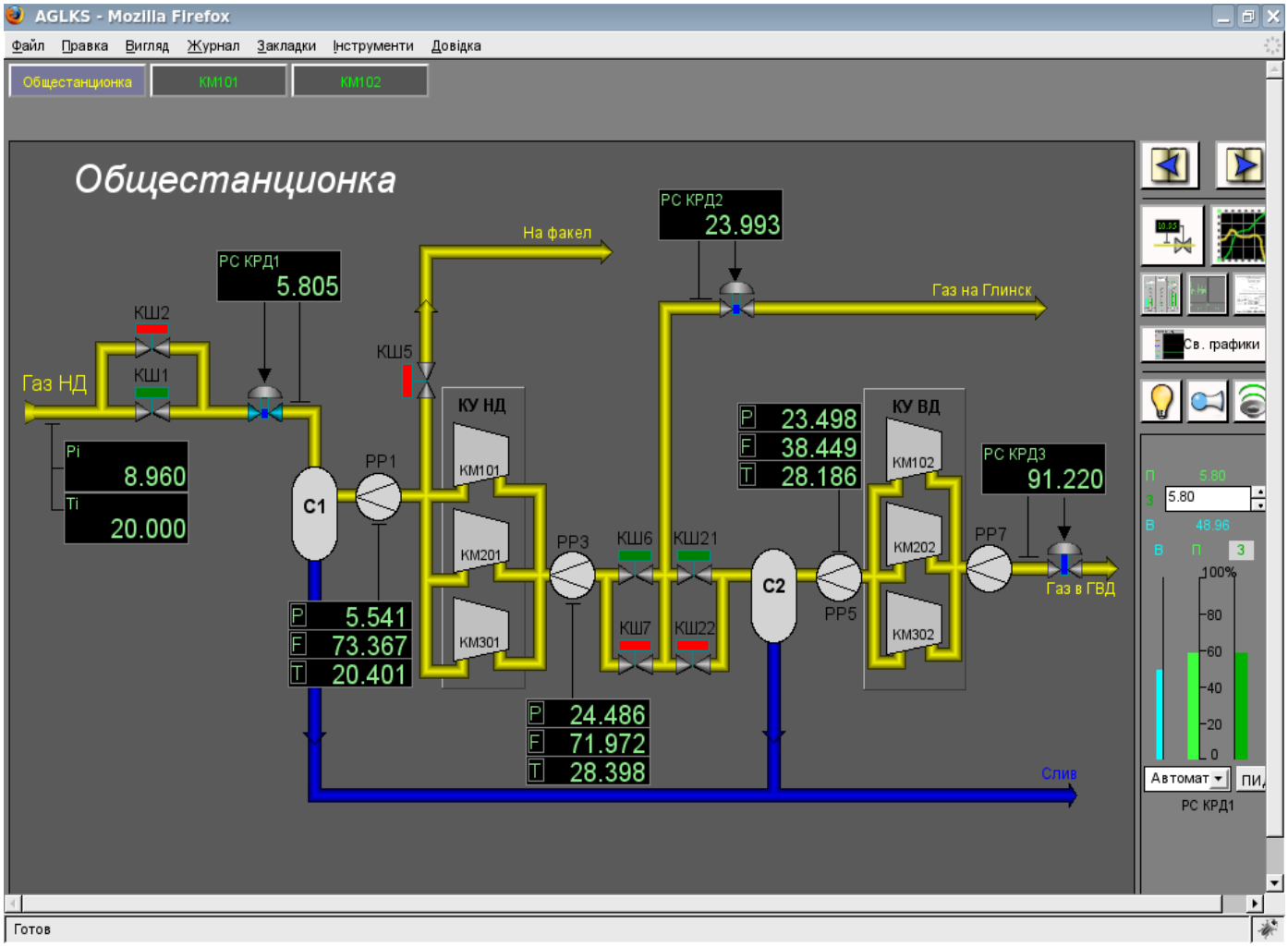


Рис.3. Главное окно режима исполнения.

3. Представление базовых элементов (примитивов)

В данной версии этого модуля реализованы не все образы примитивов заложенные этим проектом. В общем же проектом заложены примитивы:

Id	Наименование	Функция
ElFigure	Элементарные графические фигуры	Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Предусматривается поддержка следующих элементарных фигур: • Линия. • Дуга. • Кривая безье. • Заливка замкнутого пространства. Для всех фигур, содержащихся в виджете устанавливаются единые свойства толщины, цвета и т.д., но это не исключает возможность указания вышеперчисленных атрибутов конкретно для каждой фигуры отдельно.
FormEl	Элементы формы.	Включает поддержку стандартных компонентов формы: • Редактирование строки. • Редактирование текста. • Флажок. • Кнопка. • Поле выбора из списка. • Список. • Слайдер. • Строка прокрутки.
Text	Текст	Элемент текста(метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием.
Media	Медиа	Элемент отображения растровых и векторных изображений различных форматов, проигрывания анимированных изображений, проигрывание аудио фрагментов и просмотр видео-фрагментов. Возможно в него стоит включить поддержку OpenGL!
Diagram	Диаграмма	Элемент диаграммы с поддержкой возможности отображения нескольких потоков трендов и различных режимов отображения, от минималистического до полноэкранного, двухмерного, трёхмерного, кругового и т.д.
Protocol	Протокол	Элемент протокола, визуализатора системных сообщений, с поддержкой различных режимов работы при различных размерах и установках.
Document	Документ	Элемент формирования отчётов, журналов и другой документации на основе указанных данных.
Function	Функция API объектной модели OpenSCADA	Невизуальный, на стороне исполнения, виджет, позволяющий включать вычислительные функции объектной модели OpenSCADA в СВУ.
Box	Контейнер	Содержит механизм размещения других виджетов с целью формирования новых, более сложных виджетов и страниц конечной визуализации.

Более детально рассмотрим реализацию каждого примитива.

3.1. Примитив элементарная фигура (ElFigure)

Реализована поддержка элементарных фигур: линии, эллиптической дуги, кривой Безье и заливка замкнутых контуров цветом и изображением. Для элементарных фигур реализованы следующие операции:

- создание/удаление фигур;
- копирование фигур;
- перемещение и изменение размеров фигур с помощью мыши и клавиатуры;
- возможность связывать элементарные фигуры друг с другом, получая более сложные, для которых доступны все свойства исходных элементарных фигур;
- возможность одновременного перемещения нескольких фигур;
- заливка замкнутого контура цветом и/или изображением;
- генерация событий клавиш мыши, в момент клика мышью на залитые контура;
- масштабирование;
- поворот.

На рис. 4 представлена часть экрана с кадром, содержащим вышеперечисленные элементарные фигуры.

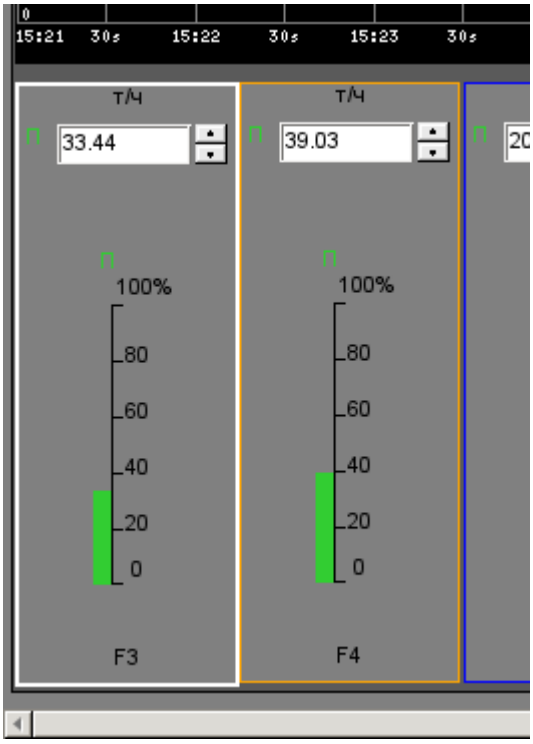


Рис.4 Реализация элементарных фигур в WebVision.

3.2. Примитив текста (Text)

Реализована поддержка элемента текста со свойствами:

- Шрифт со свойствами: типа/класса шрифта, размера, усиления, наклонности, подчёркивания и перечёркивания.
- Цвет текста.
- Ориентация текста.
- Автоматический перенос по словам.
- Выравнивание текста по горизонтали и вертикали со всеми вариантами.
- Отображение фона в виде цвета и/или изображения.
- Отображение бордюра вокруг текста, с указанным цветом, шириной и стилем.
- Формирование текста из атрибутов различного типа и свойств.

На рис. 5 представлена часть экрана с кадром, содержащим примеры текста с использованием различных параметров.

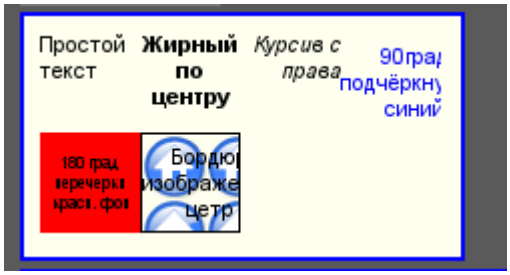


Рис.5. Реализация базового элемента текста в WebVision.

3.3. Примитив элементов формы (FormEl)

Реализована поддержка элементов формы на кадрах СВУ. Реализованы заложенные свойства, включая следующие элементы формы:

Редактор строки — Представлено следующими видами: «Текст», “Combo”, «Целое», «Вещественное», «Время», «Дата», «Время и Дата».

Редактор текста — Представляет редактор плоского текста с подтверждением или отказом от ввода.

Поле флажка — Предоставляет поле бинарного флажка.

Кнопка — Предоставляет кнопку с поддержкой: цвета кнопки, изображения в кнопке и режима фиксации.

Выбор из списка — Предоставляет поле выбора элемента, со списка указанных элементов.

Список — Предоставляет поле списка с контролем за текущим элементом.

Слайдер — Элемент слайдера (не реализован).

Прогрес-бар — Полоска прогресс-бара (не реализован).

Реализованы режимы: «Включен» и «Активен», а также передача изменений и событий в модель данных СВУ (движок). Для всех реализованных представлений поддерживается активный режим, т.е. элементы могут быть использованы для создания форм пользовательского ввода.

На рис. 6 представлена часть экрана с кадром, содержащим вышеперечисленные элементы формы.

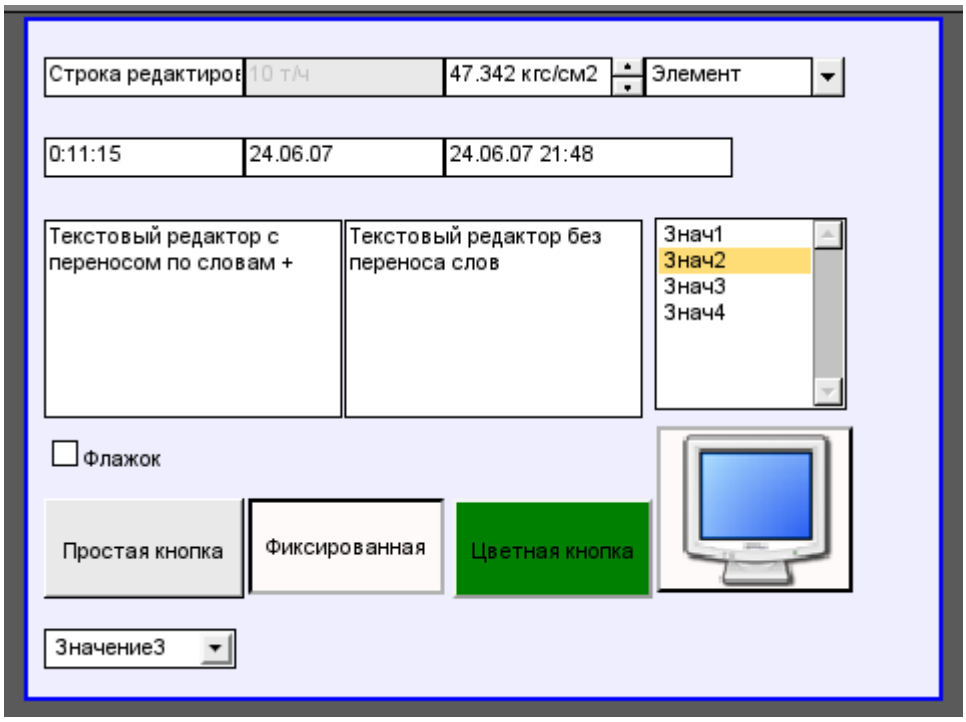


Рис.6. Реализация элементов формы в WebVision.

3.4. Примитив отображения медиа-материалов (Media)

Реализована поддержка элемента отображения медиа-материалов со свойствами:

- Указания источника медиа данных (изображения или видео-материала).
- Просмотра изображений большинства известных форматов с возможностью их вписывания в размер виджета.
- Проигрывания простых анимированных форматов изображений и видео.
- Отображение фона в виде цвета и/или изображения.
- Отображение бордюра вокруг текста, с указанным цветом, шириной и стилем.
- Формирования активных областей и генерация событий при их активации.

На рис. 7 представлена часть экрана с кадром, содержащим примеры просмотра/проигрывания медиа-данных.

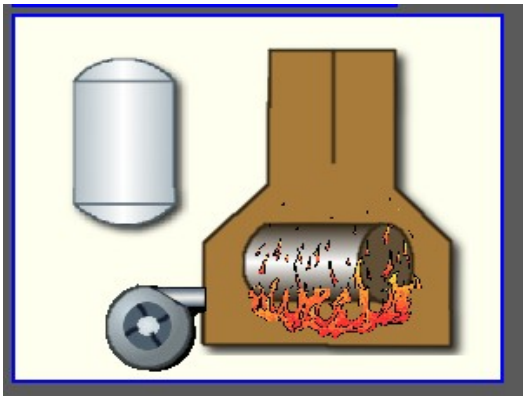


Рис.7. Реализация базового элемента отображения медиа-материалов в WebVision.

3.5. Примитив построения диаграмм/графиков (Diagram)

Реализована поддержка элемента построения диаграмм/трендов со свойствами:

- Построение графиков/трендов:
  - Построения графика для: архивных данных, текущих данных и формирования промежуточного буфера отображения для параметров без архива.
  - Построение как одиночных графиков со значением параметра по оси ординат, так и сводных графиков, включающих до 10 параметров, с процентной шкалой.
  - Возможность адаптации графика параметра к значениям данным, подгон шкалы.
  - Широкий диапазон масштабирования и адаптации горизонтальной шкалы, с автоматическим усреднением на уровне сервера и самого примитива.
  - Возможность отображение размерной сетки и маркеров по горизонтали и вертикали, с адаптацией к диапазону отображения.
  - Возможность установки курсора тренда мышью.

На рис. 8 представлена часть экрана с кадром, содержащим примеры диаграммы-тренда.

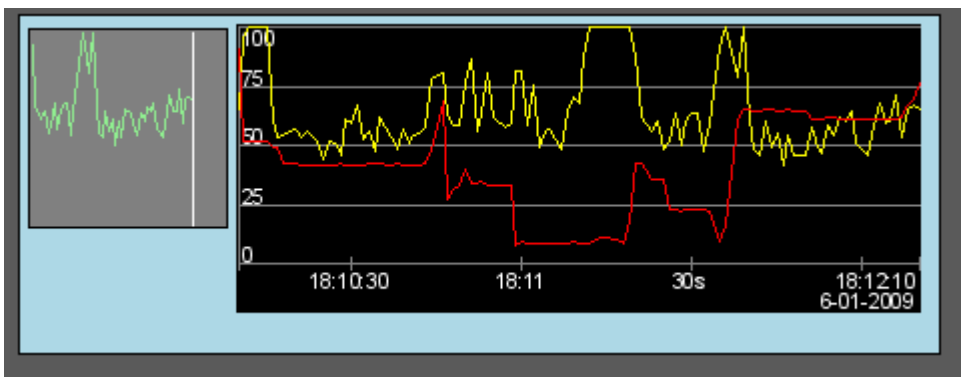


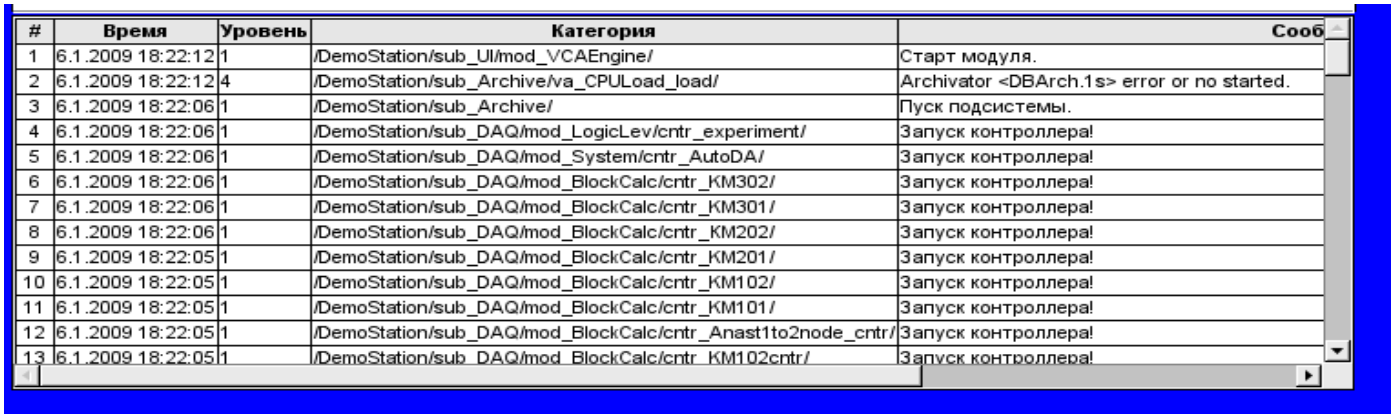
Рис.8. Реализация базового элемента отображения диаграммы-тренда в WebVision.

3.6. Примитив формирования протокола (*Protocol*)

Реализована поддержка элемента формирования протокола со свойствами:

- Формирование протокола из архива сообщений за указанное время и глубину.
- Запрос данных из указанных архиваторов сообщений.
- Выборка данных из архивов по уровню важности и шаблону категории сообщений.
- Поддержка режима слежение за появлением сообщений в архиве сообщений.

На рис. 9 представлена часть экрана с кадром, содержащим пример протокола.



#	Время	Уровень	Категория	Сооб
1	6.1.2009 18:22:12	1	/DemoStation/sub_UI/mod_VCAEngine/	Старт модуля.
2	6.1.2009 18:22:12	4	/DemoStation/sub_Archive/va_CPULoad_load/	Archivator <DBArch.1s> error or no started.
3	6.1.2009 18:22:06	1	/DemoStation/sub_Archive/	Пуск подсистемы.
4	6.1.2009 18:22:06	1	/DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/	Запуск контроллера!
5	6.1.2009 18:22:06	1	/DemoStation/sub_DAQ/mod_System/cntr_AutoDA/	Запуск контроллера!
6	6.1.2009 18:22:06	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/	Запуск контроллера!
7	6.1.2009 18:22:06	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/	Запуск контроллера!
8	6.1.2009 18:22:06	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM202/	Запуск контроллера!
9	6.1.2009 18:22:05	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM201/	Запуск контроллера!
10	6.1.2009 18:22:05	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/	Запуск контроллера!
11	6.1.2009 18:22:05	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/	Запуск контроллера!
12	6.1.2009 18:22:05	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/	Запуск контроллера!
13	6.1.2009 18:22:05	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102cntr/	Запуск контроллера!

Рис.9 Реализация базового элемента отображения протоколов в WebVision.

3.7. Примитив формирования отчётной документации (*Document*)

Реализована поддержка элемента формирования отчётной документации со свойствами:

- Гибкое формирование структуры документа на основе языка гипертекстовой разметки. Это предоставляет поддержку широких возможностей форматирования документов.
- Формирование документов по команде или по графику. Необходимо для формирования отчётной документации в архив с последующим просмотром архива.
- Формирование документа в режиме реального времени. Для формирования документов полностью динамически и на основе архивов за указанное время.
- Использование атрибутов виджета для передачи значений и адресов на архивы в документ. Позволяет использовать виджет документа как шаблон при формирования отчётов с другими входными данными.

В основе любого документа лежит XHTML-шаблон. XHTML-шаблон это тег "body", WEB-страницы, содержащий статику документа в стандарте XHTML 1.0 и элементы исполняемых инструкций на одном из языков пользовательского программирования OpenSCADA в виде <?dp <procedure> ?>. Результирующий документ формируется путём исполнения процедур и вставки их результата в документ.

Источником значений исполняемых инструкций являются атрибуты виджета этого примитива, а также все механизмы языка пользовательского программирования. Атрибуты могут добавляться пользователем и линковаться на реальные атрибуты параметров или-же являться автономными, значения которых будут формироваться в скрипте виджета. В случае со слинкованными атрибутами могут извлекаться значения из истории, архива.

На рис. 10 представлен кадр, содержащий пример документа.

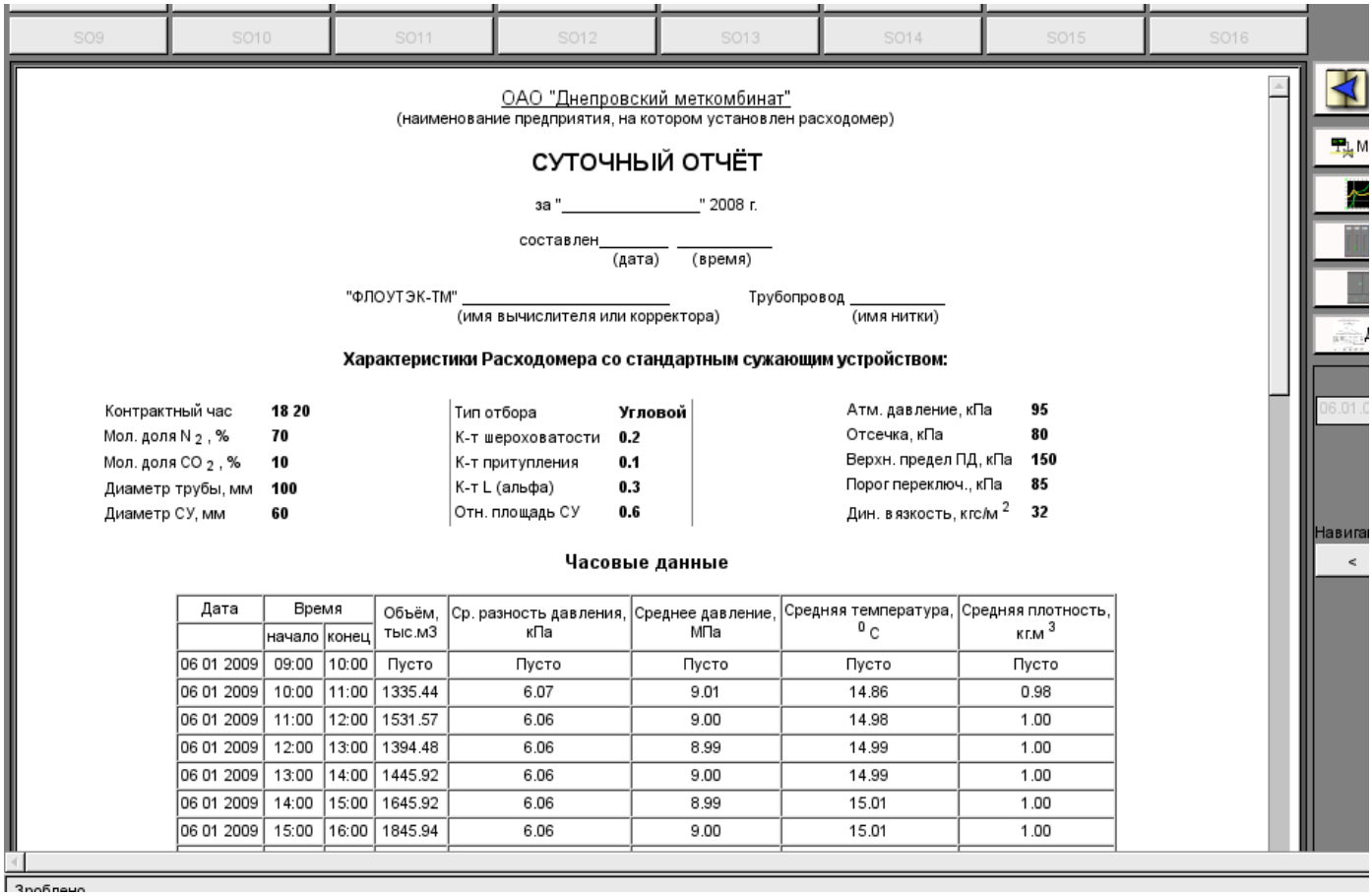


Рис.10 Реализация базового элемента отображения отчётной документации в WebVision.

3.8. Примитив контейнера (Box)

Реализована поддержка примитива контейнера, по совместительству выполняющего роль страниц проектов. Данный примитив является единственным элементом-контейнером, который может включать в себя ссылки на кадры из библиотеки, формируя тем самым пользовательские элементы нужной конфигурации. Примитив реализует предусмотренные проектом свойства. Перечислим по пунктам свойства данного примитива:

Контейнер — Позволяет формировать нужные объекты путём группировки базовых в рамках данного примитива.

Страница — Элементы построенные на данном примитиве могут выполнять роль страницы пользовательского интерфейса.

Контейнер страниц — Свойство замещения собственного содержимого другой страницей, в процессе исполнения. Используется для формирования фреймов на страницах пользовательского интерфейса. Например, главная страница традиционной SCADA системы с объектами сигнализации строится именно таким образом.

Фон — Поддерживает возможность указания фона в виде цвета или изображения.

Бордюр — Поддерживает возможность изображения бордюра с указанным цветом, толщиной и стилем.

4. Общая конфигурация модуля

Для настройки собственного поведения, в не очевидных ситуациях, модулем предоставляется возможность настройки отдельных параметров посредством интерфейса управления OpenSCADA (рис. 11). Таковыми параметрами являются:

- Время жизни сессии аутентификации.

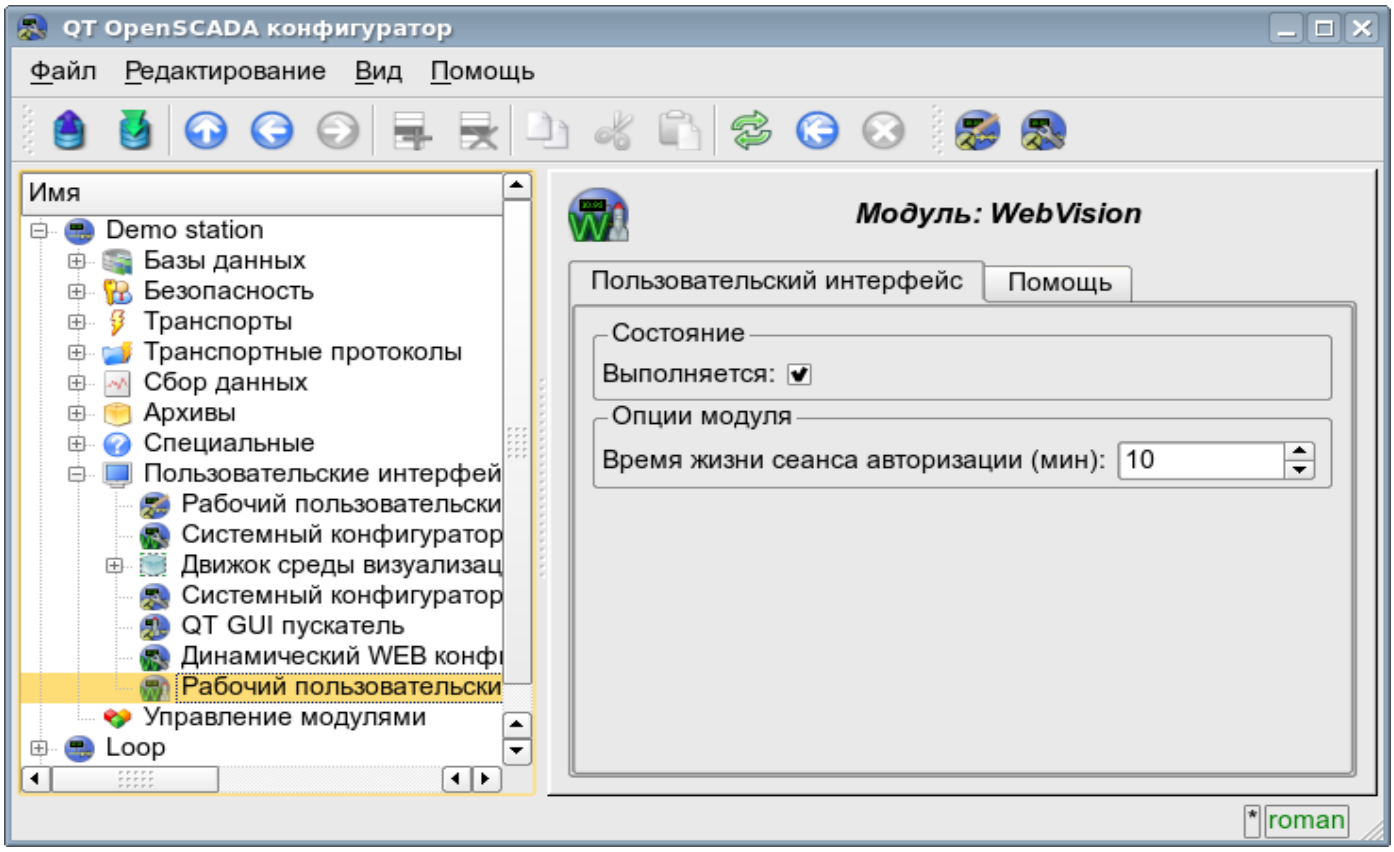


Рис.11. Страница конфигурации модуля.

Заключение

На данном этапе модуль может быть использован для построения реальных интерфейсов пользователя с поддержкой основных функций. Однако отдельные проблемы могут возникать как по причине недоработанности, так и различия браузеров. Сейчас достаточно качественно обеспечена работоспособность на браузерах: FireFox, Konqueror, Opera и Google Chromium.

Модуль подсистемы “Пользовательские интерфейсы” <WebUser>

Модуль:	WebUser
Имя:	Web-интерфейс от пользователя
Тип:	Пользовательские интерфейсы
Источник:	ui_WebUser.so
Версия:	0.6.0
Автор:	Роман Савоченко
Описание:	Позволяет создавать собственные пользовательские web-интерфейсы на любом языке OpenSCADA.
Лицензия:	GPL

Модуль WebUser предоставляет пользователю механизм создания Web-страниц, а также позволяет обрабатывать иные Web-запросы на одном из внутренних языков OpenSCADA, обычно JavaLikeCalc, не прибегая к низкоуровневому программированию OpenSCADA.

Кроме принадлежности модуля системе OpenSCADA он также принадлежит и является модулем модуля транспортного протокола <HTTP>. Собственно, вызов WebUser производиться из Protocol.HTTP. Вызов производится посредством расширенного механизма коммуникации через экспортированные в модуле WebUser функции: HttpGet() и HttpSet().

Адресация страниц начинается со второго элемента URI. Это связанно с тем, что первый элемент URI используется для идентификации самого модуля пользовательского Web-интерфейса. Например URL: *http://localhost.localdomain:10002/WebUser/UserPage* можно расшифровать как вызов пользовательской страницы “UserPage” Web модуля WebUser на хосте localhost.localdomain через порт 10002. В случае отсутствия второго элемента URI и указания отображать индекс пользовательских страниц в конфигурации, формируется индекс страниц (рис.1).

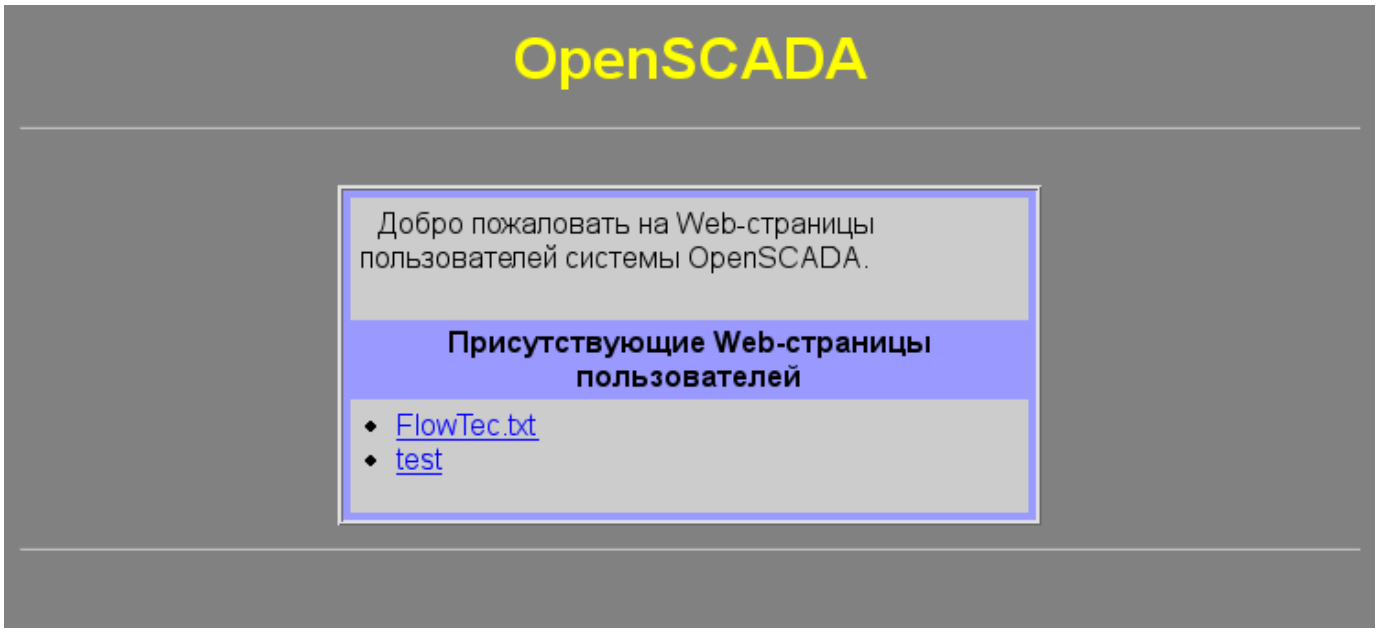


Рис.1. Индекс пользовательских страниц.

Главная вкладка конфигурации модуля (рис.2) содержит состояние модуля, предоставляет возможность выбора страницы по умолчанию и позволяет сформировать перечень пользовательских страниц.

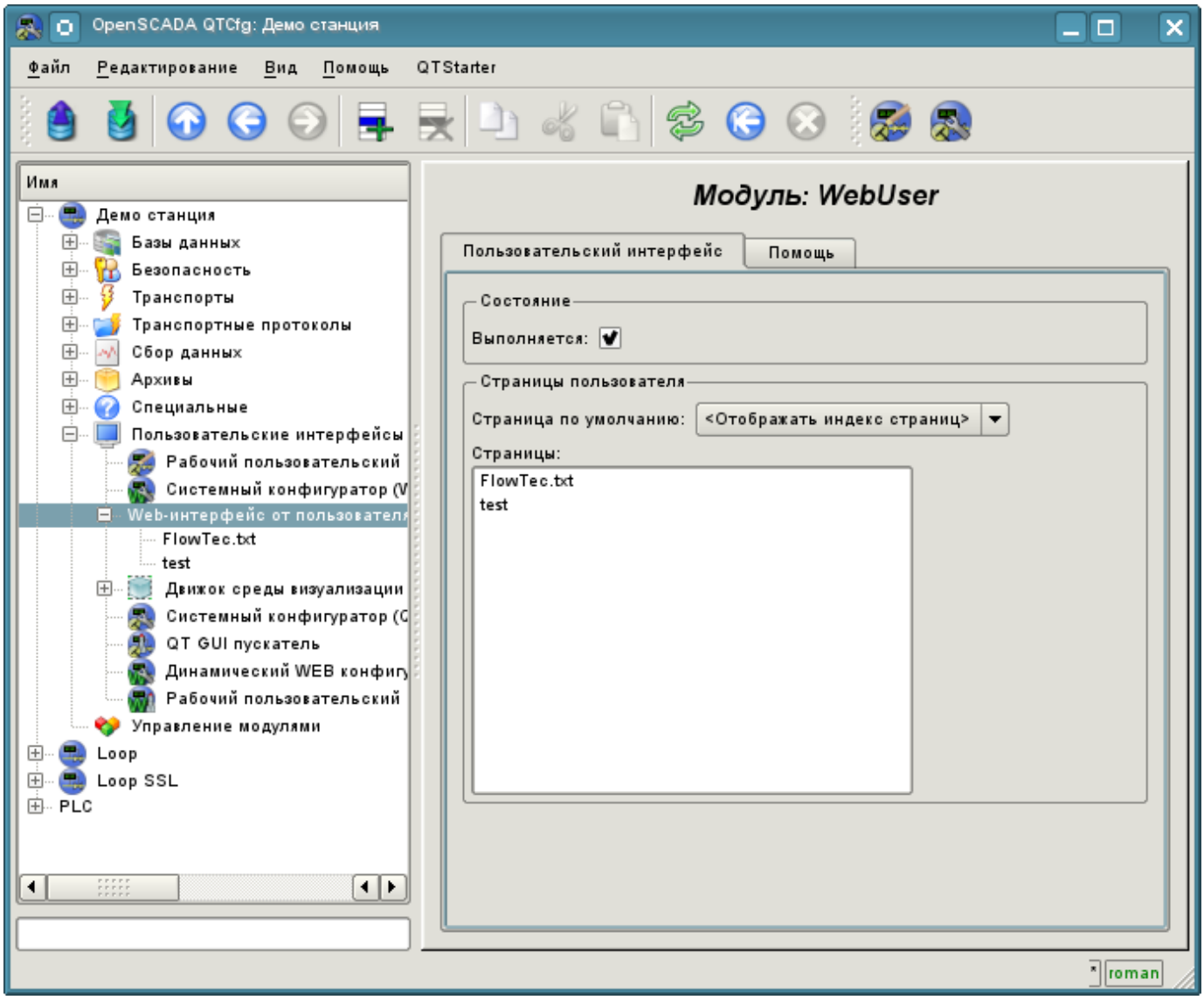


Рис.2. Основная вкладка конфигурации модуля.

1. WEB - страницы

Модуль предоставляет возможность создания реализаций множества Web-страниц в объекте "Пользовательская страница" (рис.3).

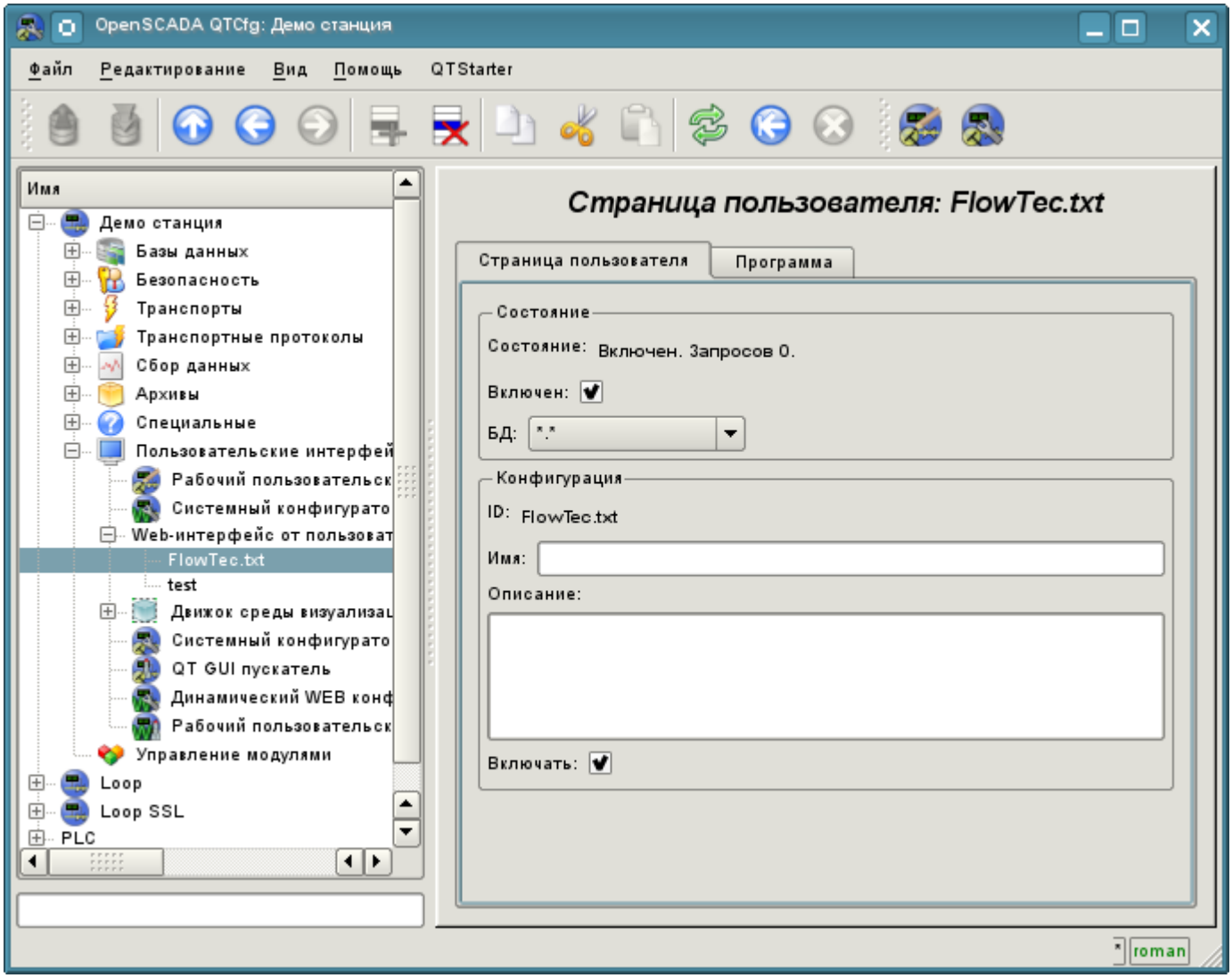


Рис.3. Главная страница конфигурации пользовательской страницы.

Главная вкладка содержит основные настройки пользовательского протокола:

- Раздел "Состояние" - содержит свойства, характеризующие состояние пользовательской страницы:
  - *Включен* - состояние страницы "Включена".
  - *БД* - БД, в которой хранится конфигурация.
- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:
  - *ID* - информация об идентификаторе страницы.
  - *Имя* - указывает имя страницы.
  - *Описание* - краткое описание страницы и её назначения.
  - *Включать* - указывает на состояние "Включен", в которое переводить страницу при загрузке.

Все запросы к пользовательским страницам направляются в процедуру обработки запросов пользовательской страницы, которая представлена на вкладке "Программа" объекта страницы пользователя (рис.4).

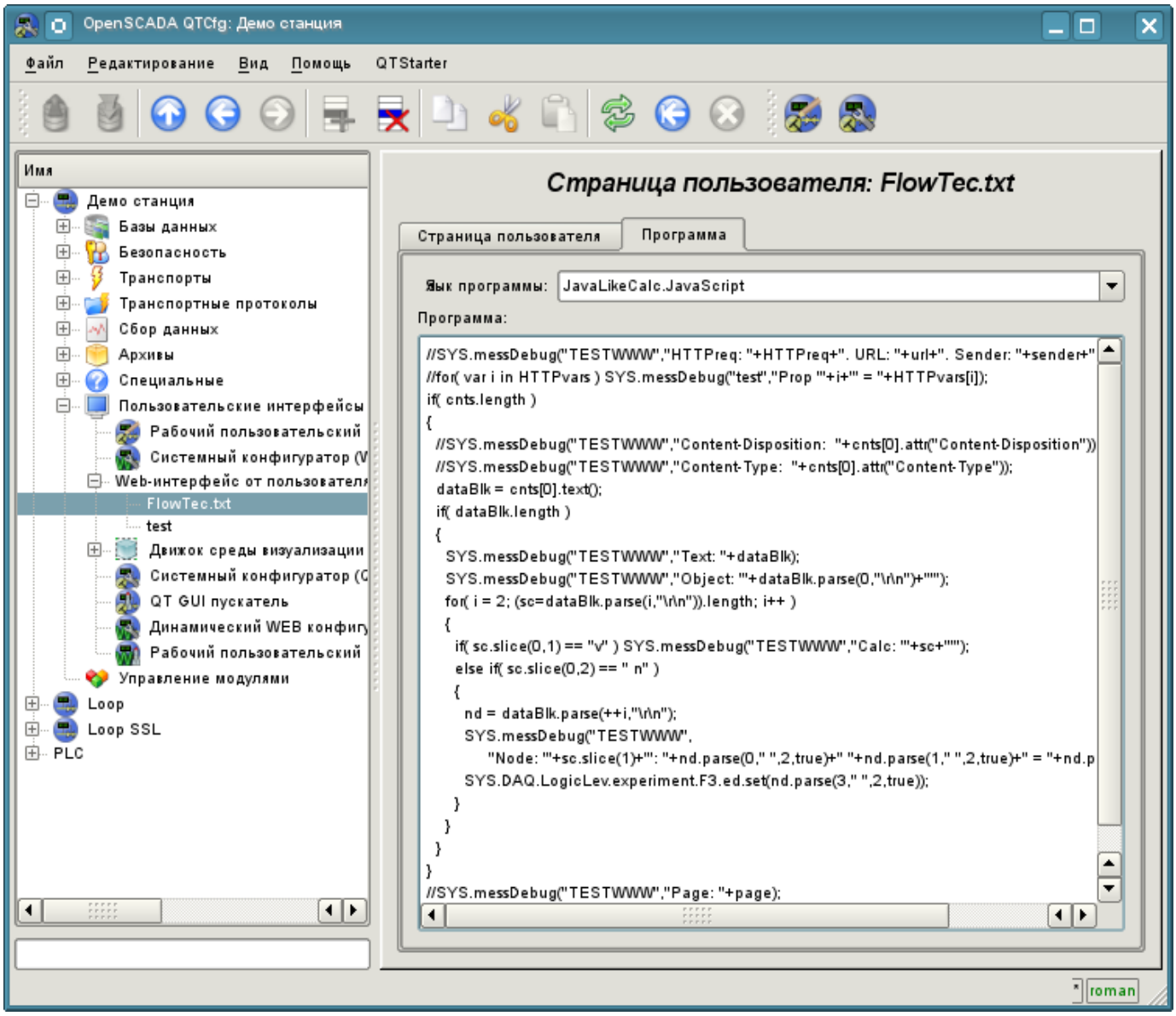


Рис.4. Вкладка "Программа" объекта пользовательской страницы.

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

Для процедуры обработки предопределены следующие переменные обмена:

- *rez* - результат обработки (по умолчанию - "200 OK");
- *HTTPreq* - метод HTTP запроса (GET,POST);
- *url* - URI запроса;
- *page* - содержимое страницы Get/Post, как для запроса так и для ответа;
- *sender* - отправитель запроса;
- *user* - аутентифицированный пользователь;
- *HTTPvars* - HTTP переменные в Object;
- *URLprms* - параметры URL в Object;
- *cnts* - элементы содержимого для POST в Array<XMLNodeObj>.

Общий сценарий запроса пользовательской страницы:

- Внешняя сетевая станция формирует HTTP запрос с URI вида "/WebUser/<UserPage>",

который попадает на транспорт OpenSCADA с значением поля конфигурации "Протокол", равного "HTTP".

- Транспорт направляет запрос на модуль транспортного протокола Protocol.HTTP.
- Модуль транспортного протокола в соответствии с первым элементом URI направляет запрос данному модулю.
- Данный модуль выбирает объект страницы пользователя, которая указана во втором элементе URI.
- Выполняется инициализация переменных HTTP-протокола для процедуры страницы:
    - *HTTPreq* - устанавливается в значение строки "GET" или "POST" в зависимости от типа запроса;
    - *url* - адрес запрашиваемого ресурса (URI);
    - *page* - содержимое передаваемой страницы для метода "POST";
    - *sender* - адрес отправителя запроса;
    - *user* - адрес аутентифицированного пользователя, если аутентификация имела место;
    - *HTTPvars* - разобранный перечень переменных протокола HTTP в виде свойств объекта;
    - *URLprms* - разобранный перечень параметров URL в виде свойств объекта;
    - *cnts* - разобранные элементы содержимого для POST в Array<XMLNodeObj> с содержимым элементов в тексте и свойствами в атрибутах XMLNodeObj.
- Вызов процедуры на исполнение, которая, обработав запрос, формирует содержимое страницы в "page" и результат запроса в "rez".
- В завершение формируется ответ с кодом возврата HTTP из "rez" и содержимым из "page".