SCADA-система А-Софт.
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.

Версия SCADA-системы: 1.0.3

Дата редакции: 27 мая 2026 года

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

СКАДА (далее – СКАДА) предназначена для разработки и исполнения прикладного программного обеспечения, предназначенного для сбора, архивирования, визуализации информации, выдачи управляющих воздействий, а также других родственных операций.

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

2.1 Состав аппаратных средств

Аппаратные требования СКАДА для её исполнения в зависимости от сложности проекта приведены в таблице 1.

Таблица 1

	Требования к аппаратной части
Проекты более 30000 точек	
АРМ	Процессор: Intel Core i3 2 ГГц; ОЗУ: 8 Гб; Жесткий диск: 40 Гб
Сервер	Процессор: Intel Xeon E5; ОЗУ: 128 Гб; Жесткий диск: 40 Гб
Сервер БД	Процессор: Intel Xeon E5; ОЗУ: 64 Гб; Жесткий диск: 1 Тб, RAID 10
Шлюз	Процессор: Intel Core i3 2 ГГц; ОЗУ: 2 Гб; Жесткий диск: 40 ГБ
Проекты до 30000 точек	
АРМ-сервер	Процессор: Intel Core i5 3 ГГц; ОЗУ: 16 Гб; Жесткий диск: 300 Гб, RAID 5
Сервер	Процессор: Intel Core i3 2 ГГц; ОЗУ: 16 Гб; Жесткий диск: 40 ГБ
Проекты до 1000 точек	
АРМ-сервер	Процессор: Intel Core i3 2 ГГц; ОЗУ: 8 Гб; Жесткий диск: 500 ГБ

2.2 Состав программных средств

Для обеспечения работоспособности СКАДА необходимо наличие операционной системы Astra Linux SE 1.7 и выше.

2.3 Требования к квалификации персонала

Специалисты, занимающиеся настройкой СКАДА, должны иметь знания для работы с системой Astra Linux на уровне пользователя.

3. АРХИТЕКТУРА СИСТЕМЫ

СКАДА представляет собой SCADA-систему, построенную по принципам модульности и масштабируемости.

Основой системы является модульное ядро. 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.

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 СКАДА как библиотеку.

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

Архитектура СКАДА представлена на рисунке 1.

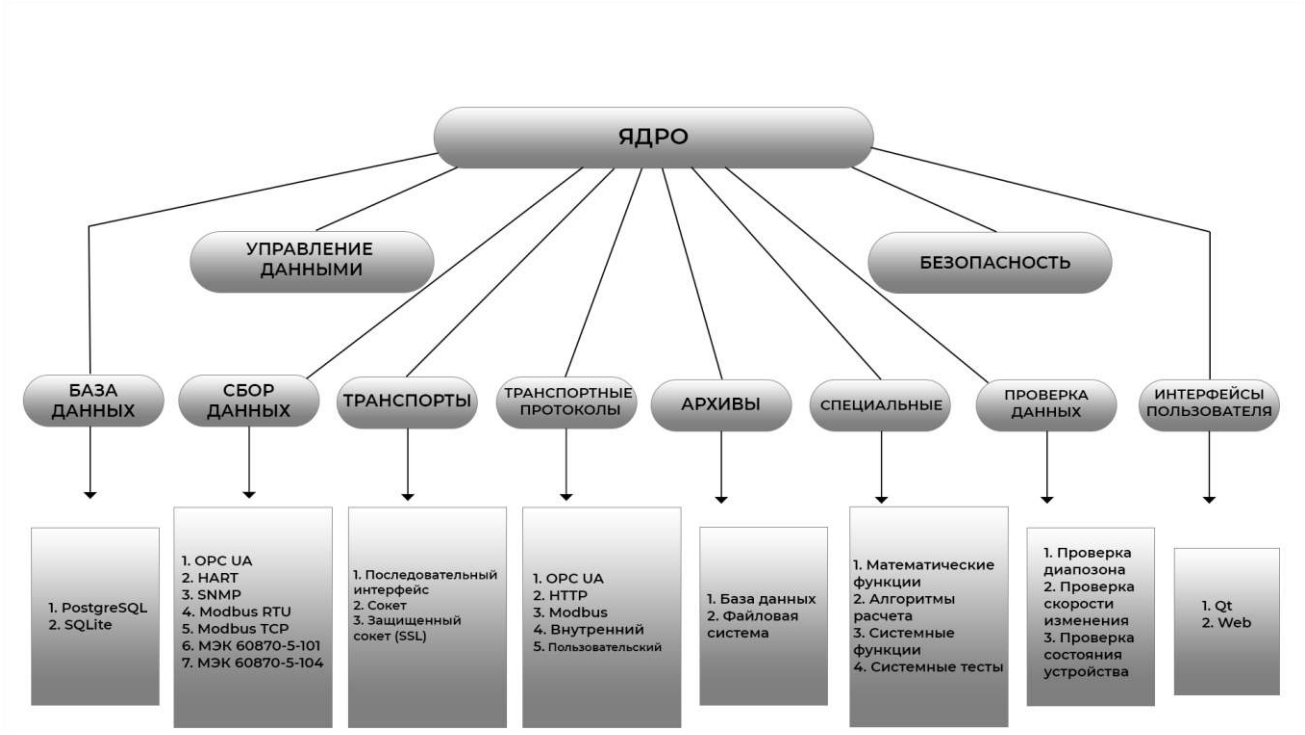


Рисунок 1

3.1. Модульность

Для придания гибкости и высокой степени масштабируемости СКАДА построена по модульному принципу. Тесная интеграция модулей с ядром улучшает стабильность системы в целом, благодаря повторному использованию отлаженного кода.

Модули СКАДА 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 СКАДА.

На основе модулей реализованы следующие функциональные части СКАДА:

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

3.2. Подсистемы

Архитектурно СКАДА 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.

Всего СКАДА содержит 10 подсистем из них 8 подсистем являются модульными. К списку 10 подсистем могут добавляться новые подсистемы посредством модулей. Подсистемы СКАДА:

- Безопасность.
- Управление модулями.
- Базы данных (модульная).

- Транспорты (модульная).
- Транспортные протоколы (модульная).
- Сбор данных (модульная).
- Архивы (модульная).
- Специальные (модульная).
- Интерфейсы пользователя (модульная).
- Проверка данных (модульная).

4. УСТАНОВКА ПРОГРАММЫ

4.1 Установка дополнительных пакетов на ОС Astra Linux

В меню «Пуск» выбрать «Панель управления» или «Настройки»(в зависимости от вида рабочего стола) → «Программы» → «Менеджер пакетов Synaptic» (рисунок 2).

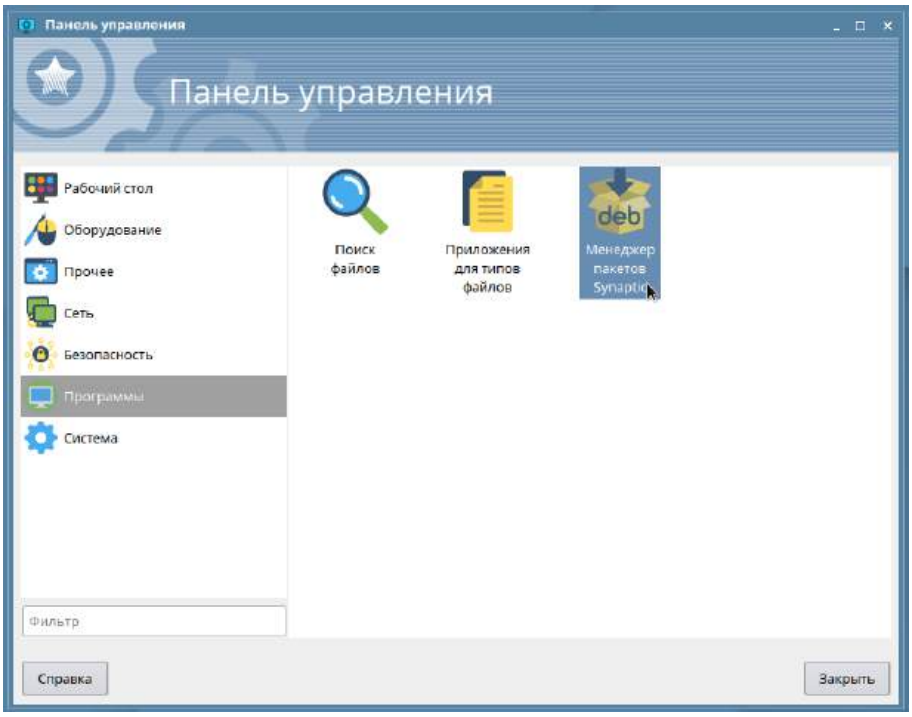


Рисунок 2

Для корректного запуска и работы в системе должны быть установлены следующие библиотеки:

- libqt5core
- libqt5widgets
- libqt5gui
- libqt5help
- libqt5printsupport
- libqt5opengl
- libpcre
- libpthread
- libgd
- libgost
- libcrypt
- zlib
- libintl
- gettext
- libsqlite3

- libpq
- libpcap
- bison
- libnetsnmp
- libssl
- libqscintilla2
- libvlc

Далее в окне «Менеджера пакетов Synaptic» необходимо нажать на иконку «Поиск» (рисунок 3) и в появившемся окне ввести название устанавливаемого пакета.

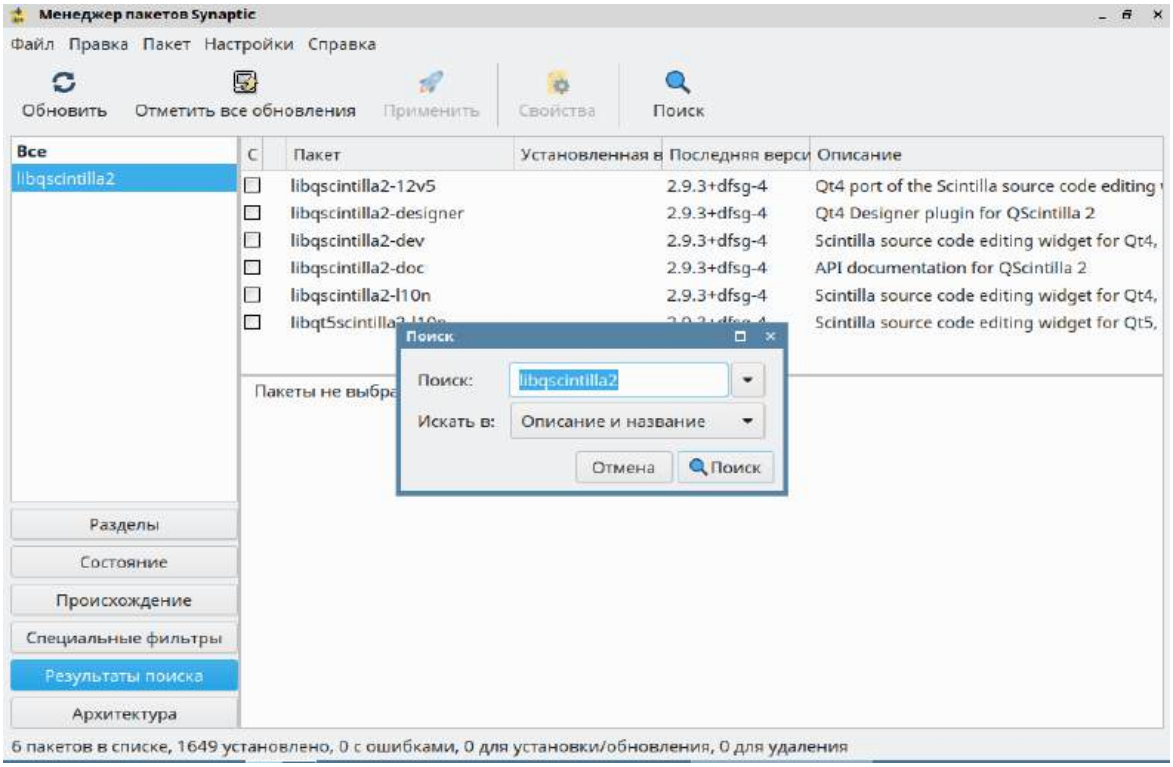


Рисунок 3

После этого, в отобразившемся списке выбрать устанавливаемый пакет двойным нажатием мыши, и согласиться с установкой дополнительных библиотек, нажав на кнопку «Применить». Проверив список изменений, нажать кнопку «Применить». Система потребует вставить диск с дистрибутивом ОС Astra Linux. После чего начнется установка выбранных пакетов.

4.2 Установка СКАДА

1.Скопировать архив asoft.distr...tar.bz2 в необходимую директорию (например: /home/asoft/scada).

2.Распаковать архив -> появится директория asoft.

3.Перейти в директорию, в которой распакован архив (например: /home/asoft/scada/asoft).

4.Выполнить команду ./configure.bin

4.3 Удаление СКАДА

1.Перейти в директорию /home/asoft/scada/asoft

2.Выполнить команду ./configure.bin -c

3.Удалить директорию /home/asoft/scada/asoft

5. КОНФИГУРАЦИЯ И НАСТРОЙКА СИСТЕМЫ

Конфигурирование СКАДА осуществляется при помощи модуля конфигурации - UI.QTCfg. Данный модуль предоставляет развитый интерфейс конфигурации позволяющий управлять, как локальной станцией, так и удалёнными станциями в локальной и глобальной сетях, включая безопасное соединение.

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

Кроме БД данные о конфигурации могут содержаться в конфигурационном файле СКАДА, а также передаваться посредством параметров командной строки при вызове СКАДА. Сохранение конфигурации в конфигурационном файле осуществляется наравне с БД. Типовым именем конфигурационного файла является /etc/scada.xml. Формат конфигурационного файла и параметры командной строки рассматриваются в 0.

Многие настройки и конфигурация объектов СКАДА, которые исполняются или уже включены, не применяются сразу же по внесению изменений, поскольку конфигурация читается/применяется обычно только при включении или запуске. Следовательно, для применения изменений в таких случаях, достаточно включить/выключить включенный объект или перезапустить исполняющийся — остановить/запустить.

5.1 Запуск программы

Запуск СКАДА осуществляется командой из командной строки терминала:

1. Перейти в директорию scada (например: /home/asoft/scada/asoft)
2. Выполнить команду (sudo ./start)

или из графического меню:

«Пуск» → «Графика» → «SCADA» с помощью мыши.

После загрузки системы появится окно регистрации (рисунок 4).

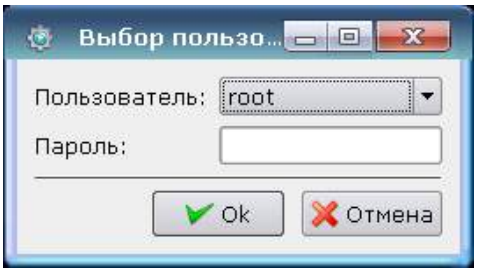


Рисунок 4

После регистрации в СКАДА (пользователь: root, пароль: root) на экране появится окно системного конфигуратора СКАДА (рисунок 5).

5.2 Конфигурация системных параметров

Конфигурация системных параметров размещается на пяти вкладках корневой страницы станции.

Вкладка "Станция" содержит основные информационные и конфигурационные параметры станции. Ее вид показан на рисунке 5.

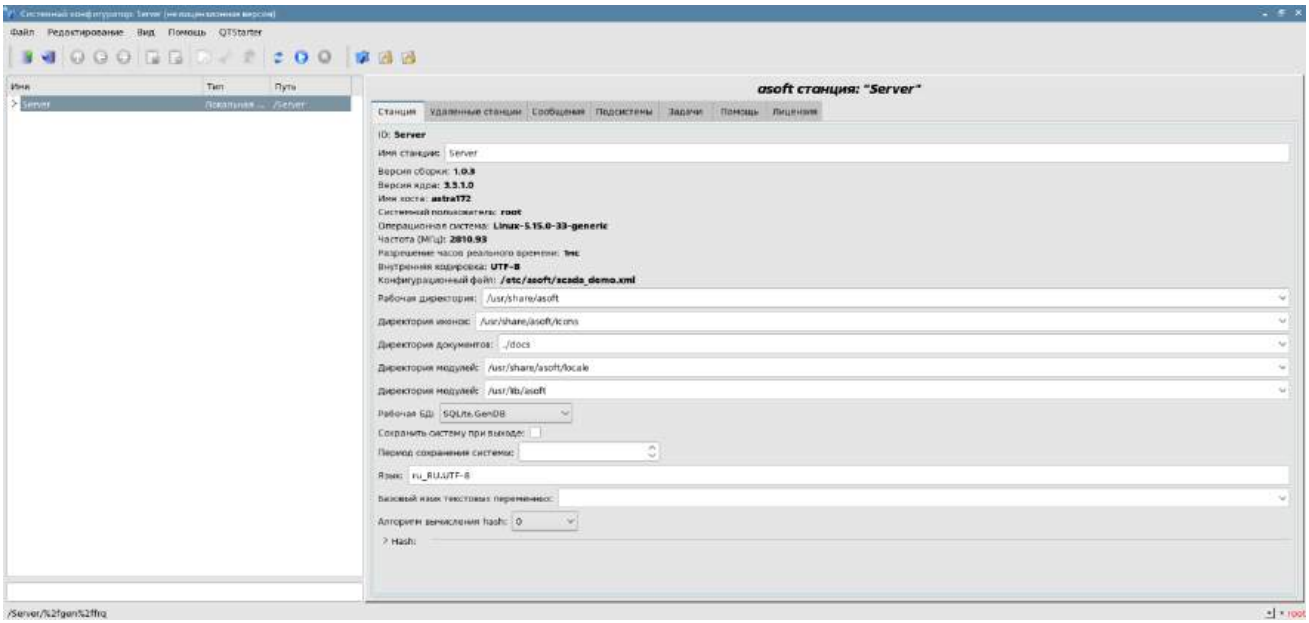


Рисунок 5

На вкладке "Станция" представлены следующие поля:

- *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;

- *Версия сборки* - содержит информацию о текущей версии программы;

- *Версия ядра* - содержит информацию о текущей версии ядра системы;

- *Имя хоста* - содержит информацию об имени машины, на которой запущена станция.

- *Системный пользователь* - содержит информацию о пользователе, от имени которого выполняется программа в ОС;

- *Операционная система* - содержит информацию об 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 (нс)* - 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 неправильно указано значение этого поля;

- *Директория документов* - указывает на директорию, содержащую справочную документацию по СКАДА;

- *Директория модулей* - указывает на директорию модулей для СКАДА. Если значение этого поля некорректно, то при запуске на экране будет отображена только информация в консоли о корректном запуске ядра СКАДА;

- *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 период в секундах, с которым необходимо 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. 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;

- *Алгоритм вычисления hash* – позволяет выбрать алгоритм вычисления контрольной суммы элемента. Для выбора доступны алгоритмы в соответствии с ГОСТ 34.11-94 и ГОСТ 34.11-2012. Подсчет контрольных сумм осуществляется в режиме исполнения и выводится в окно системного конфигуратора для следующих модулей: Сбор данных – Modbus, OPC, Логический уровень; Базы данных; Транспорты; Редактор графического интерфейса, а также вычисляется общая контрольная сумма для всех контролируемых конфигураций. Вкладка "Сообщения" - это раздел группы параметров, управляющих работой с сообщениями станции (рисунок 6), включает в себя:

- *Таблица «Уровни сообщений»* – 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. Для каждого уровня сообщения возможно выбрать логирование – указав «Trye» в соответствующей ячейке;

- *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 СКАДА механизмами ОС;

- *На стандартный выход (stdout)* – указывает на использование 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 стандартного механизма вывода ошибок, обычно тоже направляется в консоль;

- *В архив* – указывает на необходимость вывода сообщений в архив сообщений СКАДА. Этот параметр обычно включен, а его выключение приводит к фактическому отключению архивирования сообщений на станции.

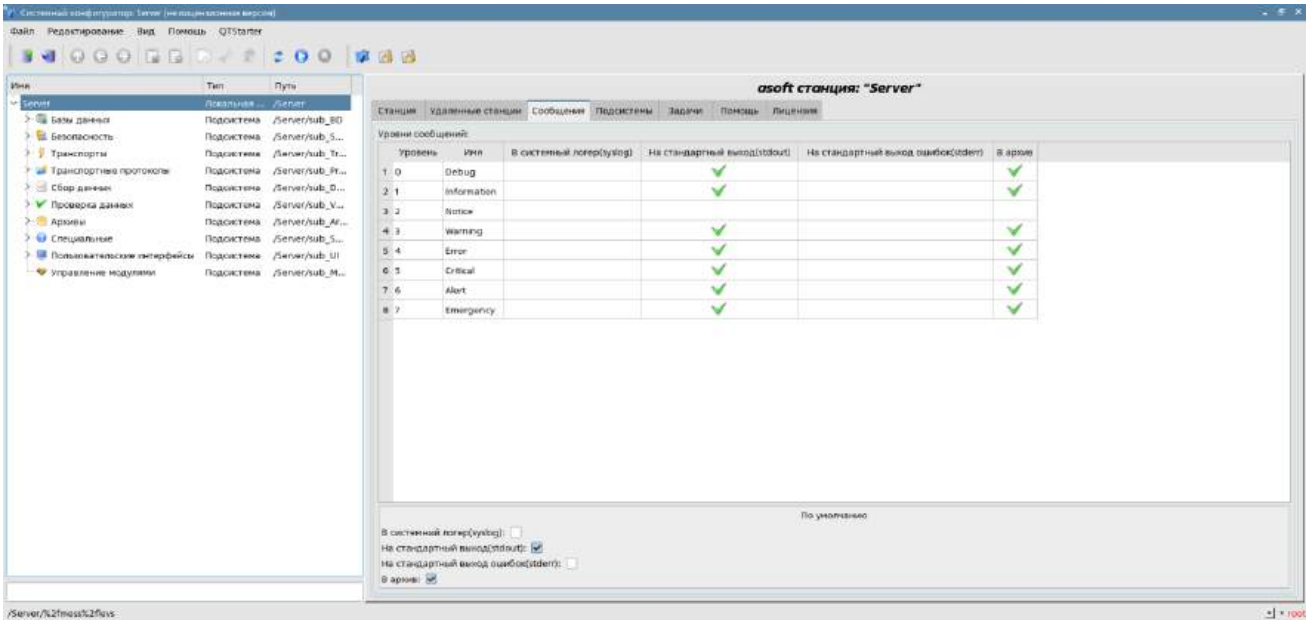


Рисунок 6

Вкладка "Подсистемы" содержит список подсистем и позволяет выполнять прямые переходы к ним с помощью контекстного меню. Ее вид показан на рисунке 7.

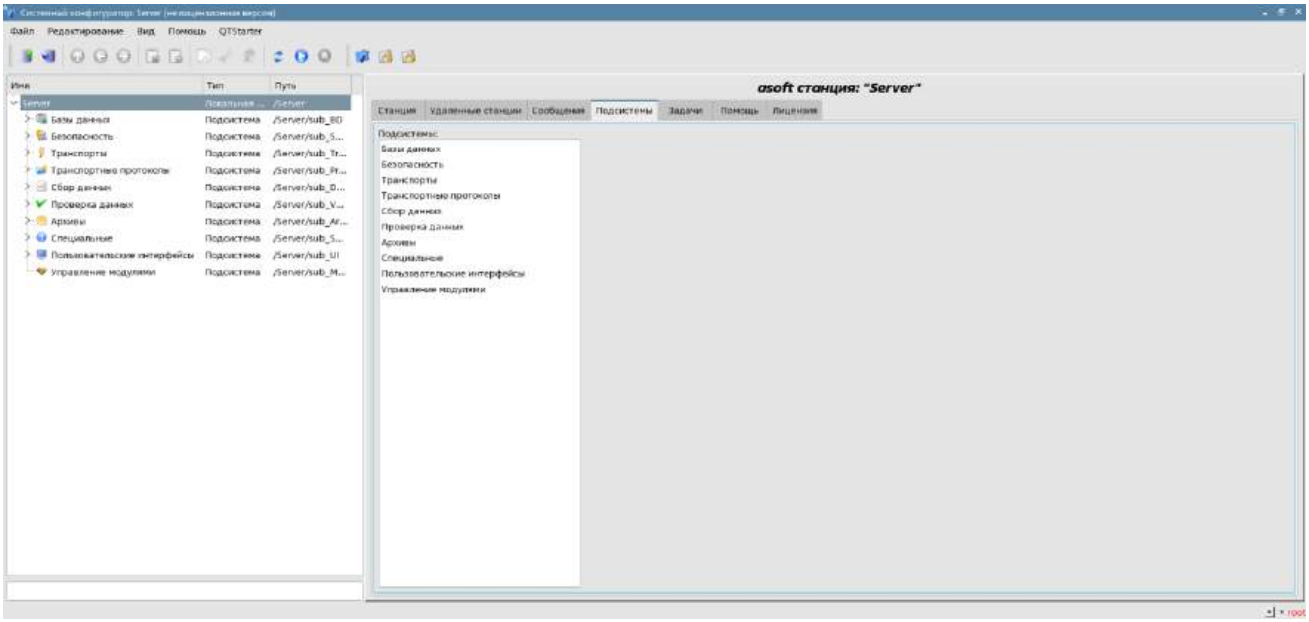


Рисунок 7

Вкладка "Задачи" содержит таблицу со списком задач открытых различными компонентами СКАДА. Из таблицы можно получит различную информацию о задачах, а также в колонке «УСТ.CPU» назначить процессоры для задач, на многопроцессорных системах.

Вид вкладки "Задачи" показан на рисунке 8.

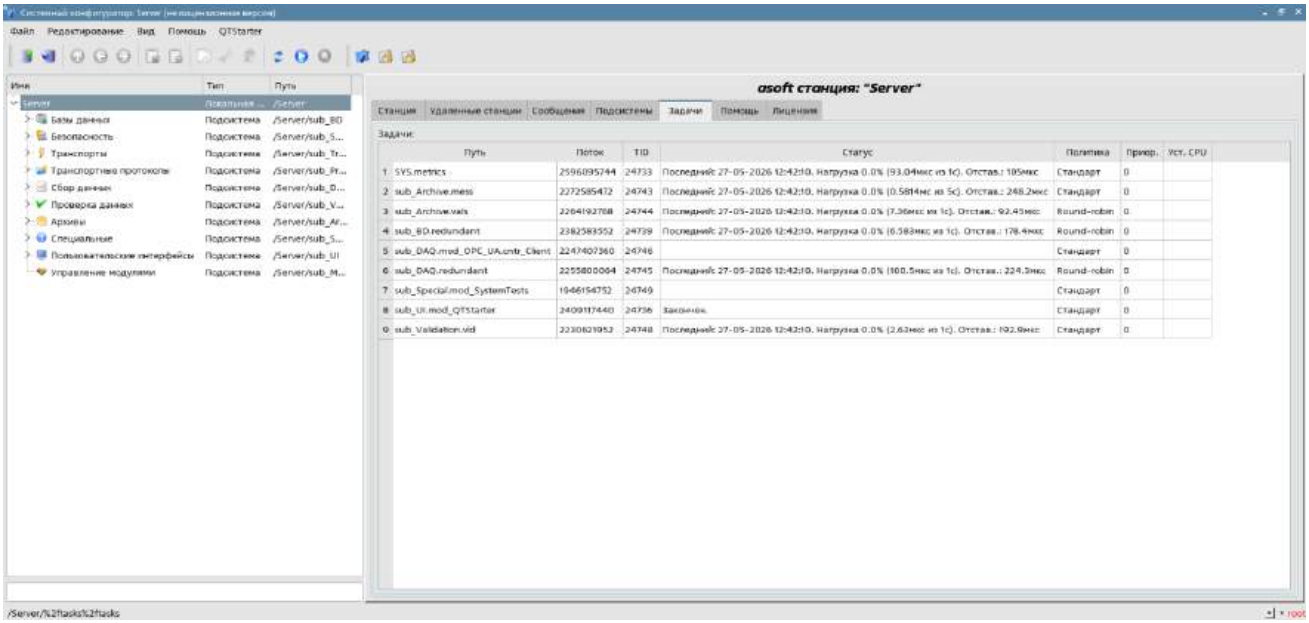


Рисунок 8

Вкладка "Помощь" содержит краткую помощь для данной страницы. Ее вид показан на рисунке 9.

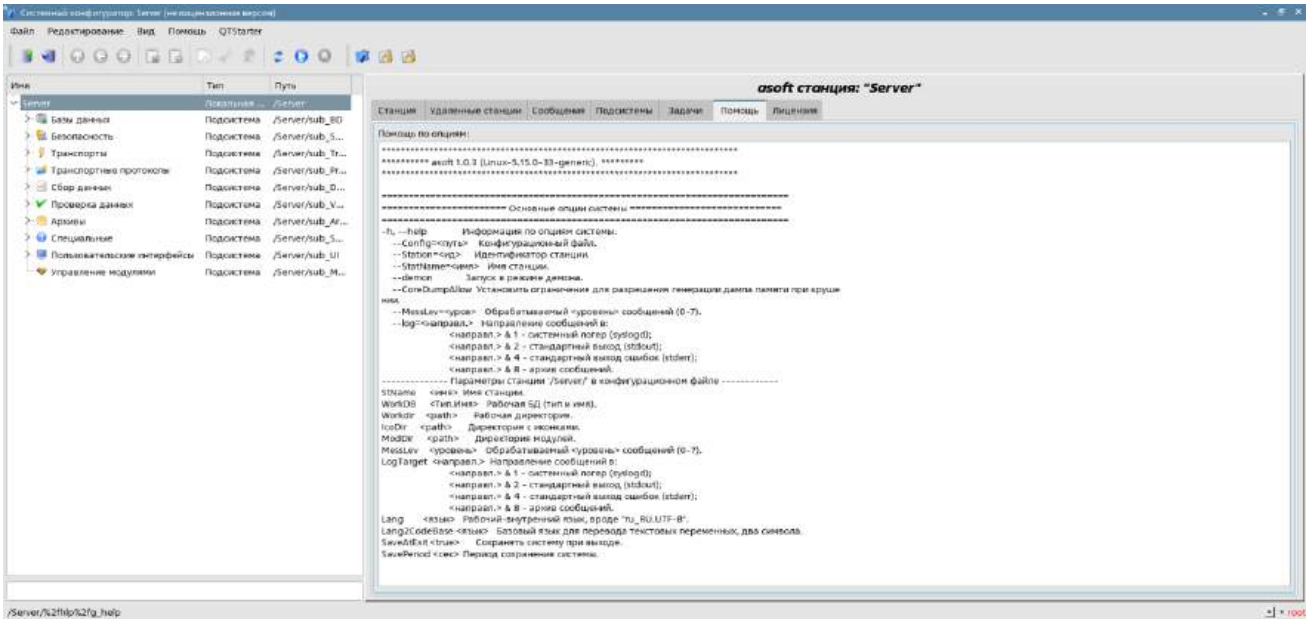


Рисунок 9

Кроме того, справочная информация о работе с системой, конфигурировании и настройкам модулей подсистемы доступны для чтения в пункте меню «Помощь» → «Справка» или нажатием клавиши «F1».

П р и м е ч а н и е. Для модификации полей корневой страницы станции могут потребоваться права привилегированного пользователя. Получить такие права можно, включив пользователя в группу суперпользователя "root", или, войдя на станцию от имени суперпользователя "root".

Поля идентификаторов всех объектов СКАДА недопустимы для прямого редактирования, поскольку являются ключом для хранения данных объектов в БД.

Поменять идентификатор объекта можно с помощью команды переноса и последующей вставки объекта в конфигураторе.

5.3 Подсистема "БД"

5.3.1 Общие сведения

Для хранения данных системы используются базы данных (БД). В целях систематизации доступа и управления базами данных в системе СКАДА 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: PostgreSQL или 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 СКАДА 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 СКАДА. 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.

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 СКАДА 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.Конфигурирование подсистемы «БД»

Для конфигурации подсистемы предусмотрена корневая страница подсистема "БД", содержащая вкладки "Резервирование", "Модули" и "Помощь". Вкладка "Резервирование" содержит поля для настройки резервирования БД. Ее вид показан на рисунке 10. Вкладка "Модули" 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 пользователя в группу "БД".

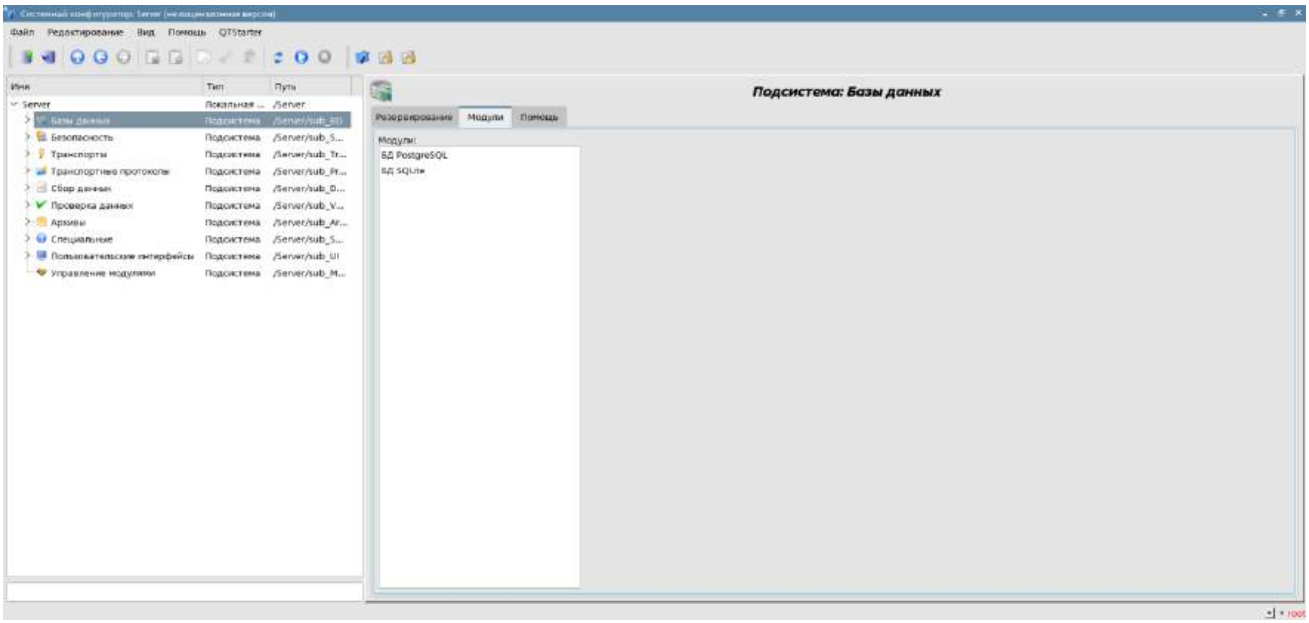


Рисунок 10

Каждый модуль подсистемы "БД" предоставляет конфигурационную страницу с вкладками "БД" и "Помощь". Вкладка "БД" содержит список БД, зарегистрированных в модуле, и флажок признака полного удаления БД (рисунок 11). Если этот флажок установлен, БД будет полностью удалена при её закрытии. Иначе БД будет просто закрыта.

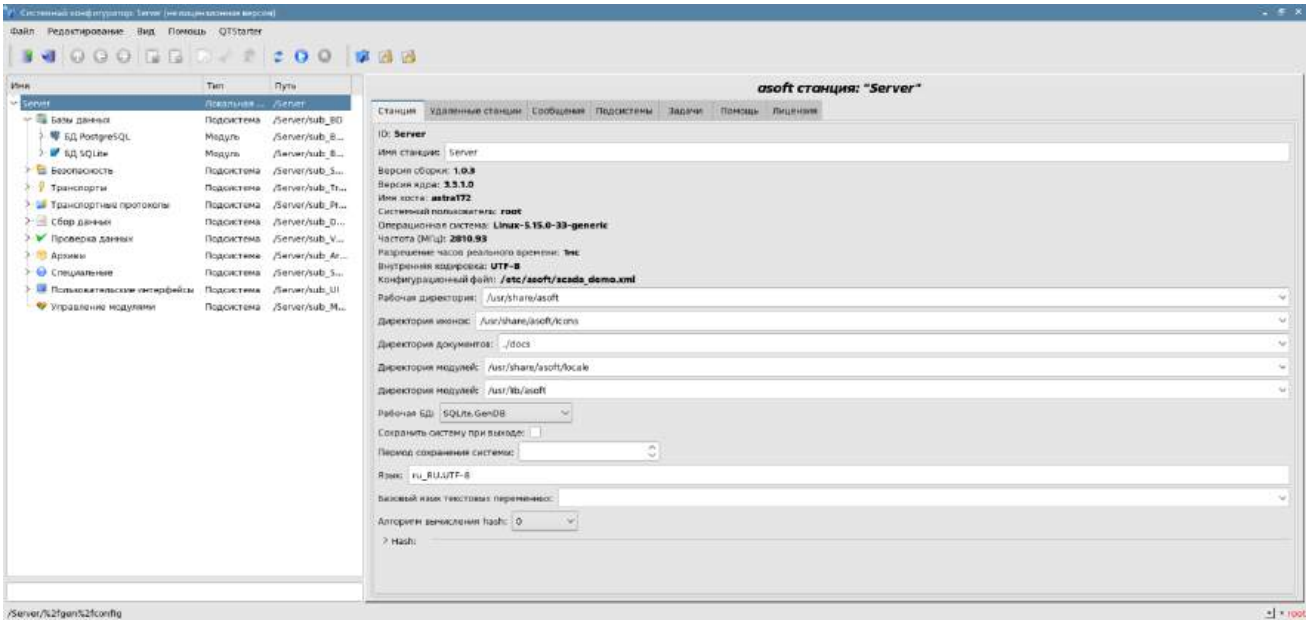


Рисунок 11

В контекстном меню списка БД пользователю предоставляется возможность добавления, удаления и перехода к нужной БД. Во вкладке "Помощь" содержится информация о модуле подсистемы "БД":

– *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.

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.

Вид вкладки "База данных" показан на рисунке 12.

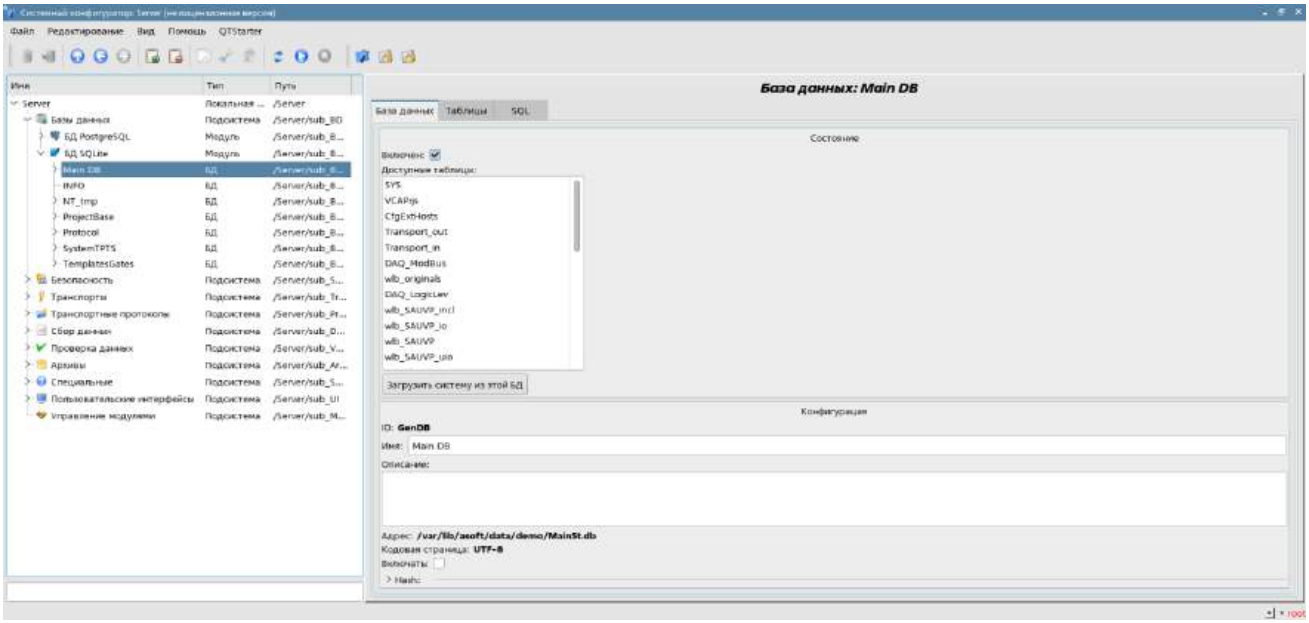


Рисунок 12

Вкладка "База данных" содержит основные настройки БД, заключенные в двух разделах.

Раздел "Состояние" - содержит свойства, характеризующие состояние БД:

- *Включен* - состояние БД "Включен";

- *Доступные таблицы* - 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) формате. Описание формата записи адреса БД, как правило, доступно во всплывающей подсказке этого поля. Форматы адресов для используемых БД приведены в таблице 2.

- *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 следует перевести БД при загрузке.

Таблица 2

Тип БД	Формат адреса
PostgreSQL	[<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: [;;user1;123456;SCADA;;10] В случае удалённого доступа к БД нужно использовать адрес хоста и порт сервера БД. Например: [server.nm.org;;user1;123456;SCADA;;10]
SQLite	[<FileDBPath>]. Где: FileDBPath - полный путь к файлу БД (./DATA/MainSt.db). Используйте пустой путь для создания временной базы данных на диске. Используйте ":memory:" для создания временной базы данных в памяти.

Вкладка "Таблицы" содержит список открытых таблиц. Наличие открытых таблиц говорит о том, что программа сейчас работает с таблицами, или таблицы открыты пользователем для изучения их содержимого. После завершения работы с таблицами программа их закрывает и вкладка «Таблицы» становится пустой.

В контекстном меню перечня открытых таблиц можно открыть таблицу для просмотра и редактирования (команда "Перейти"), удалить выбранную таблицу (команда "Удалить").

Страница просмотра содержимого таблицы содержит только одну вкладку "Таблица". Вид ее показан на рисунке 13.

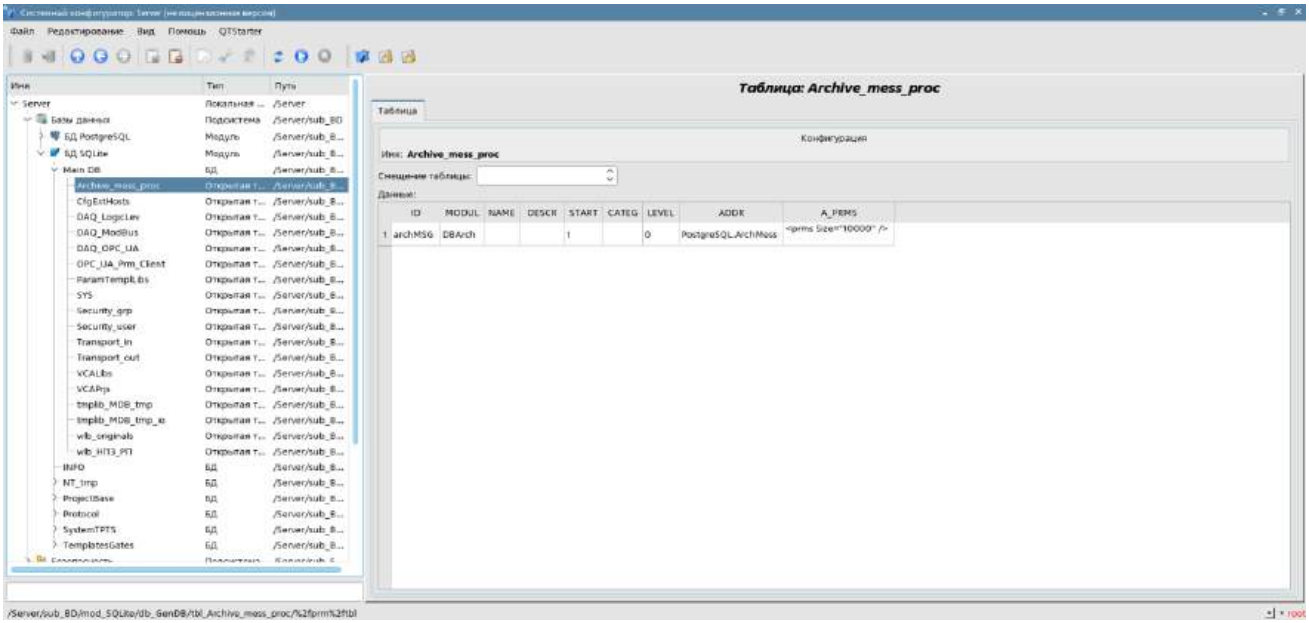


Рисунок 13

Вкладка "Таблица" содержит поле имени таблицы и таблицу с содержимым. Таблице содержимого предоставляются функции:

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

Для редактирования ячейки необходимо осуществить на ней двойной щелчок мышью (рисунок 14). После этого можно ввести новое значение в ячейку (рисунок 15).

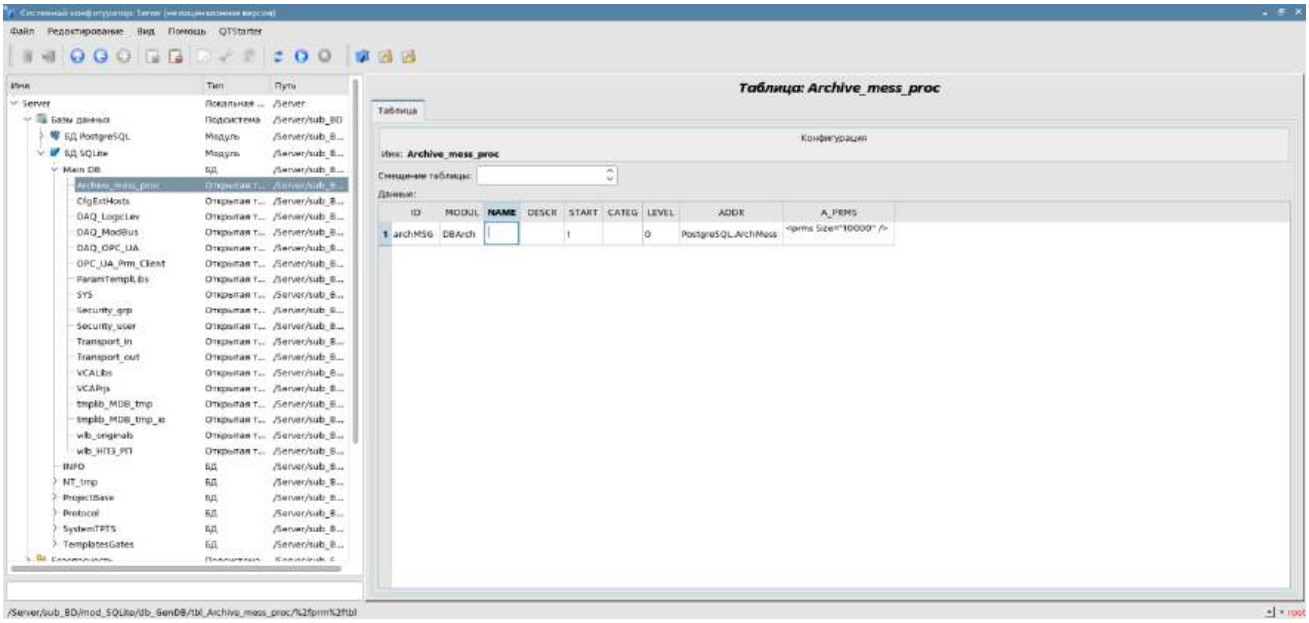


Рисунок 14

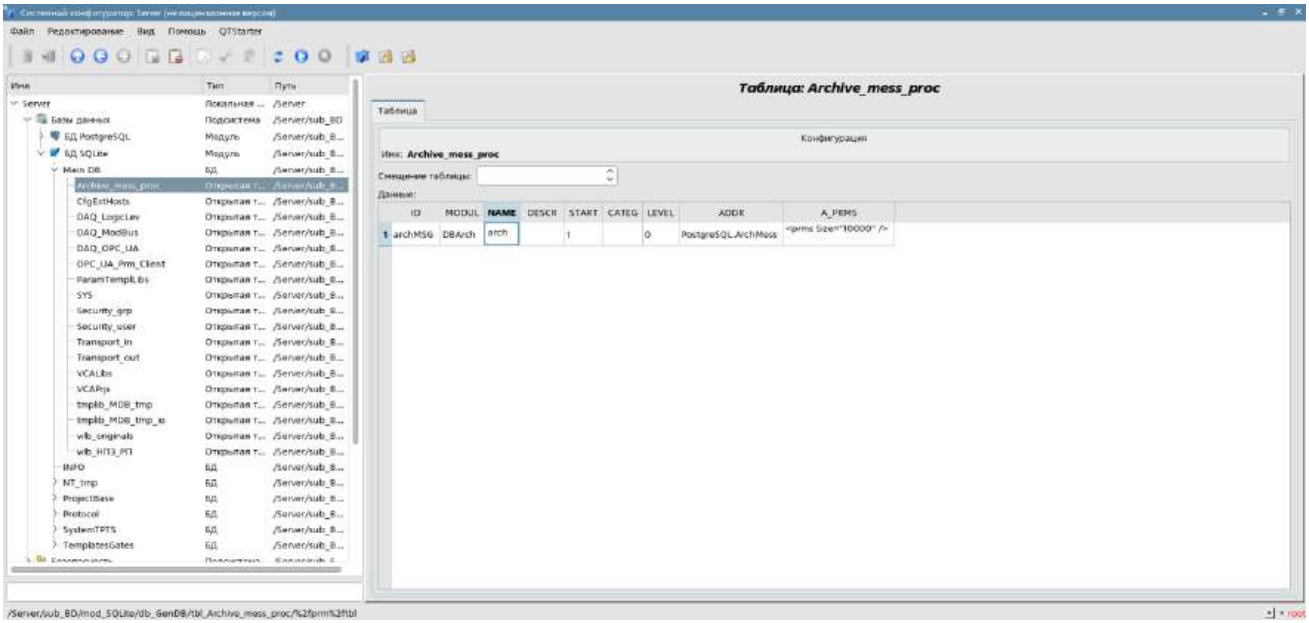


Рисунок 15

После завершения редактирования ячейки необходимо нажать клавишу «Enter» на клавиатуре, либо щелкнуть мышью по любой другой ячейке. В результате в ячейке сохранится введённое значение (рисунок 16). Если во время редактирования нажать клавишу «Esc» на клавиатуре, то вновь введенное значение не будет сохранено в ячейке.

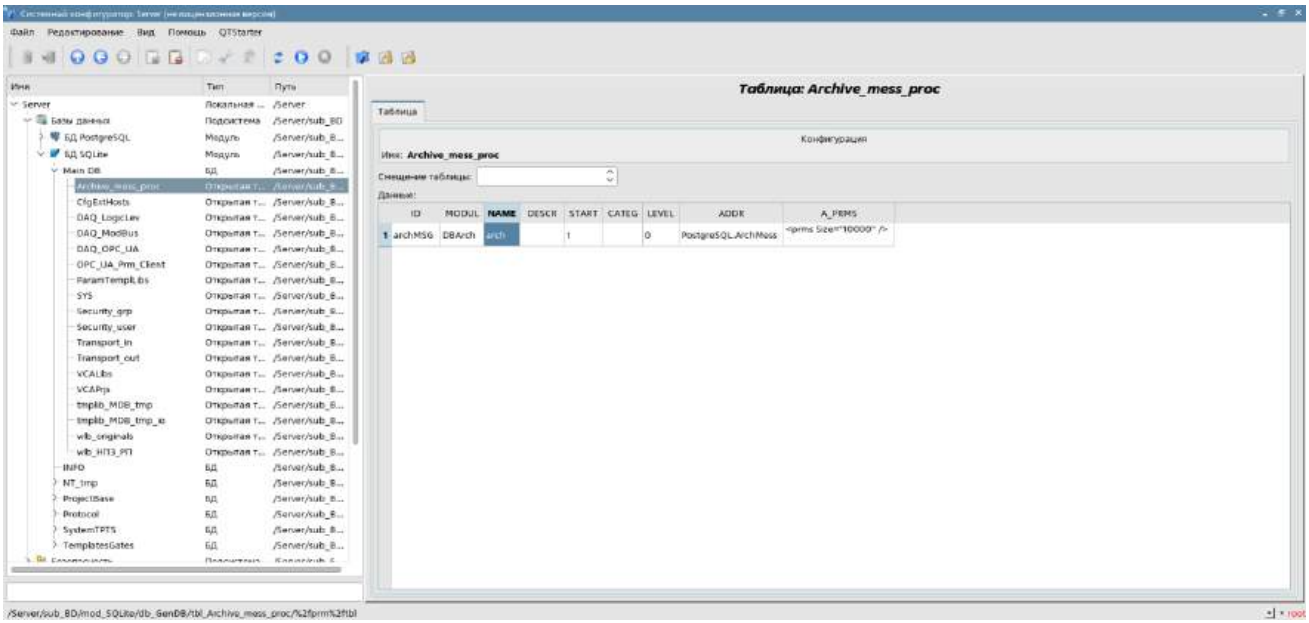


Рисунок 16

Для удаления записи (строки) нужно навести курсор мыши на нужную строку, нажать правую клавишу мыши и в появившемся списке выбрать пункт «Удалить запись» (рисунок 17).

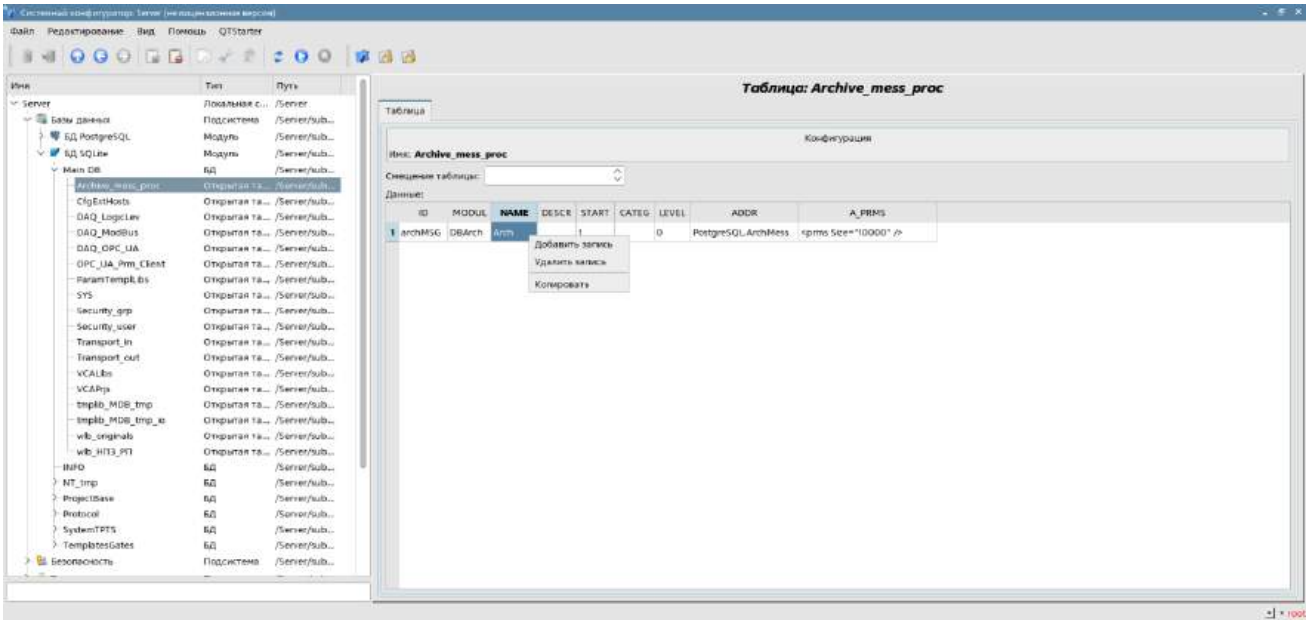


Рисунок 17

В результате данная строка будет удалена из таблицы, а остальные строки будут автоматически смещены.

Для добавления записи (строки) нужно щелкнуть правой клавишей мыши в любом месте таблицы и в появившемся списке выбрать пункт «Добавить запись» (рисунок 17).

В результате новая строка (запись) будет добавлена в конец таблицы (рисунок 18).

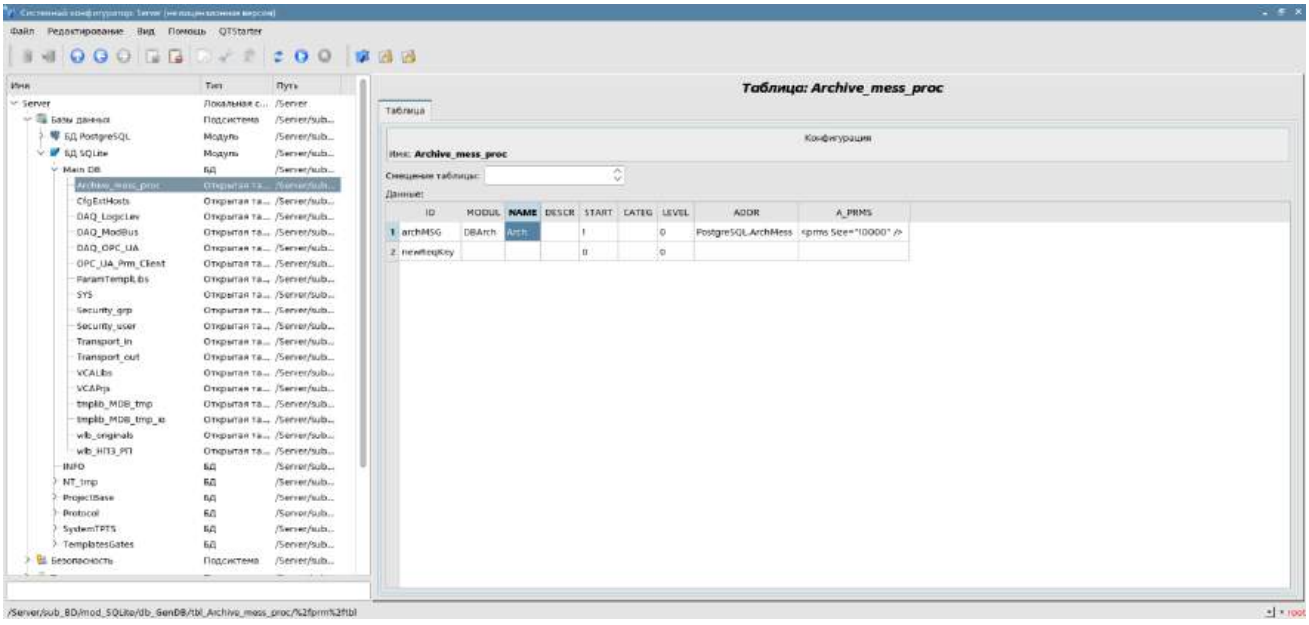


Рисунок 18

5.4 Подсистема "Безопасность"

5.4.1 Общие сведения

Для разграничения доступа в СКАДА предусмотрена подсистема "Безопасность". Основными функциями подсистемы "Безопасность" являются:

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

Подсистема "Безопасность" не является модульной.

5.4.2 Конфигурирование подсистемы «Безопасность»

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Безопасность", содержащая вкладки "Пользователи и группы пользователей" и "Помощь". Вид вкладки "Пользователи и группы пользователей" показан на рисунке 19.

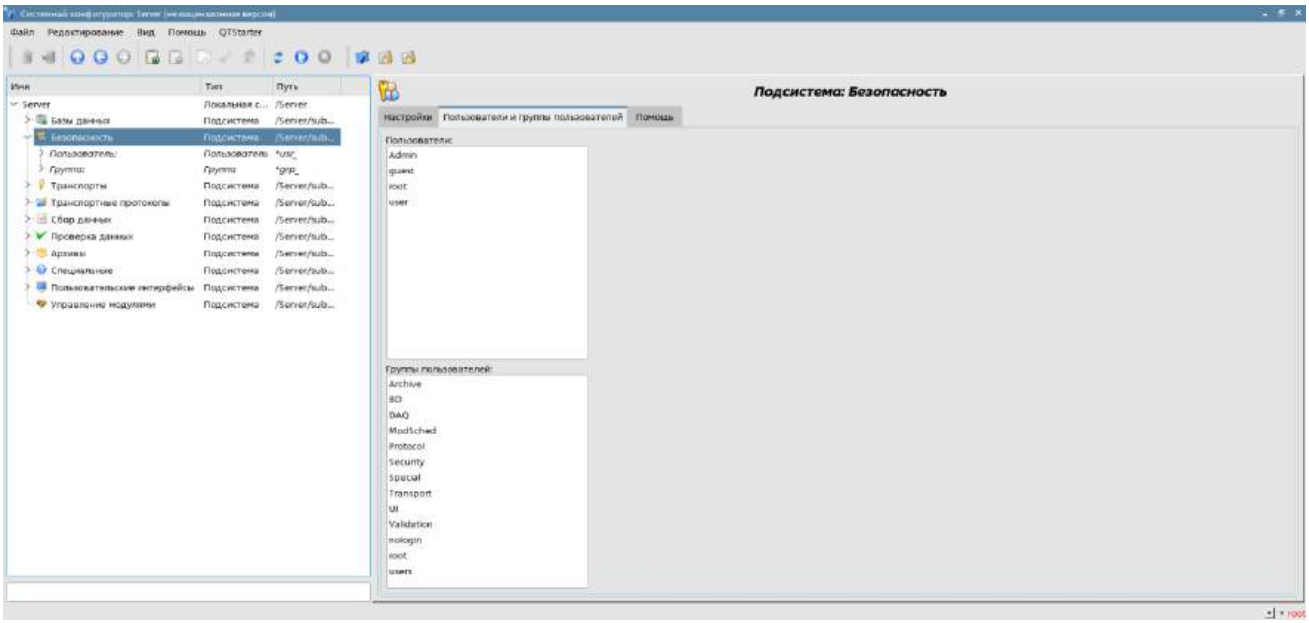


Рисунок 19

Вкладка "Пользователи и группы пользователей" 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.

Для конфигурации учетных данных пользователя предоставляется страница, содержащая только вкладку "Пользователь". Ее вид показан на рисунке 20.

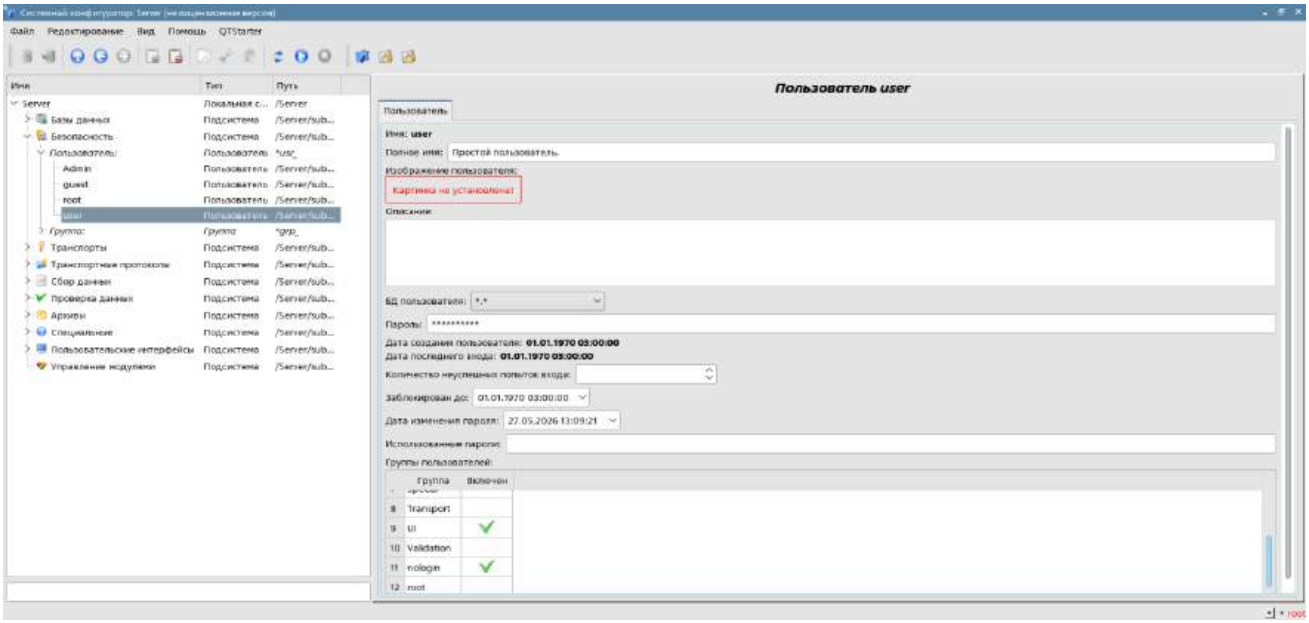


Рисунок 20

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, выбирается из списка щелчком мыши;

- *Пароль* - поле для изменения пароля пользователя. Всегда отображает "******";

- *Группы пользователей* - таблица с перечнем групп пользователей станции и признаком принадлежности пользователя к группам.

Чтобы добавить нового пользователя необходимо щелкнуть правой кнопкой мыши в области «Пользователи:» вкладки «Пользователи и группы пользователей». В появившемся меню выбрать пункт «Добавить» (рисунок 21).

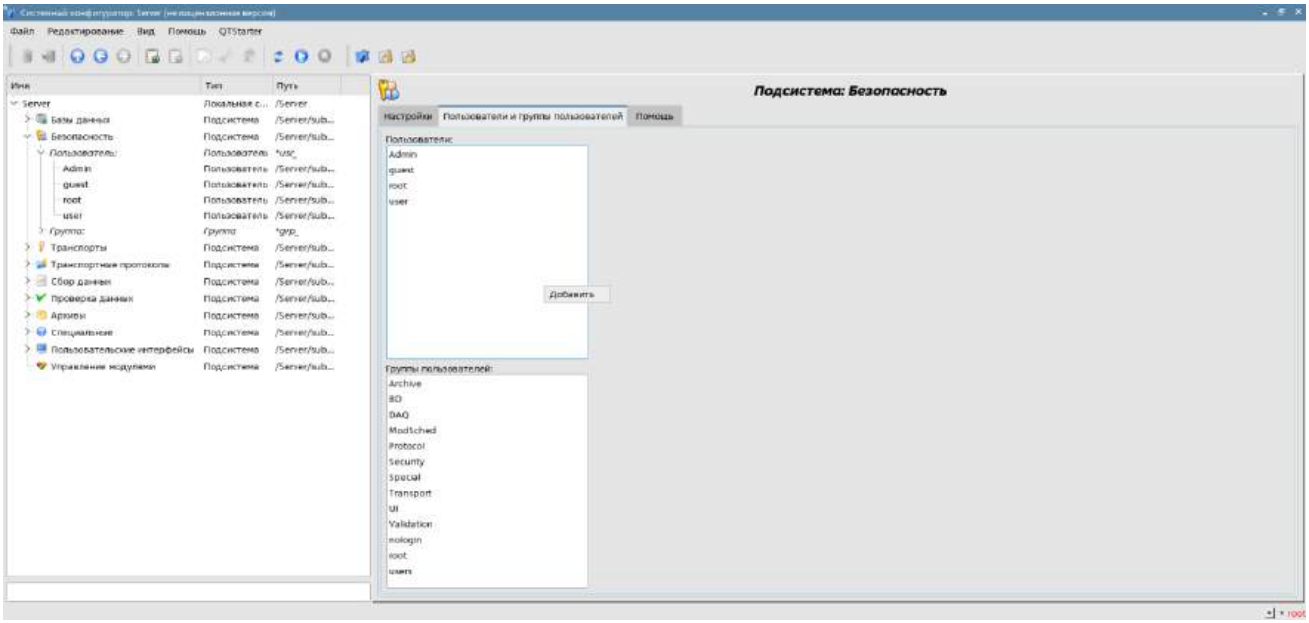


Рисунок 21

Появится окно «Добавление нового элемента», изображённое на рисунке 22. Здесь нужно ввести имя нового пользователя, например «test_user», и нажать кнопку «Ok».

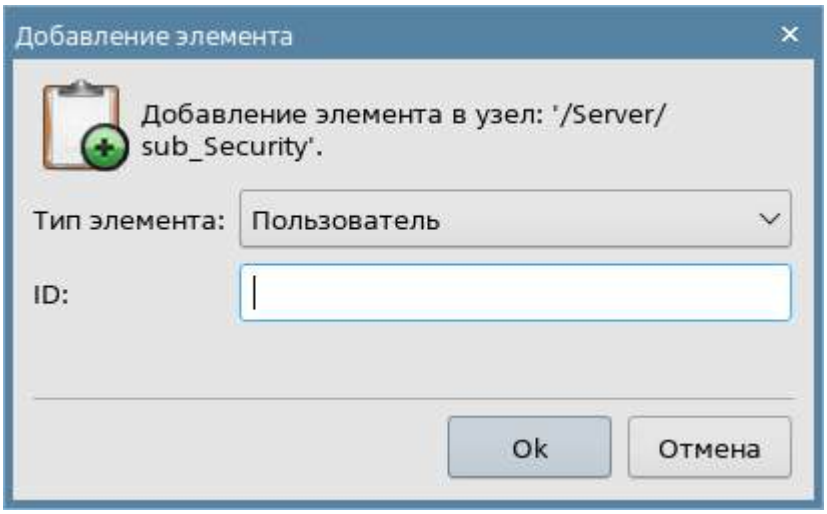


Рисунок 22

После этого новый пользователь «test_user» появится в списке пользователей, как показано на рисунке 23.

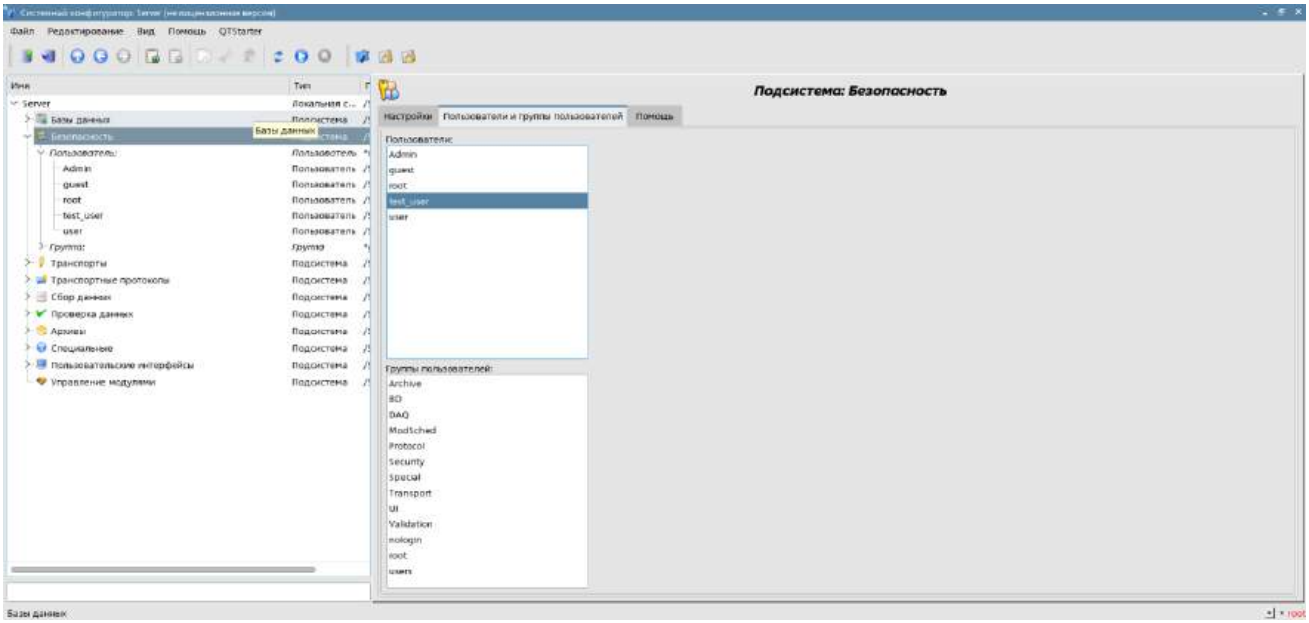


Рисунок 23

Для перехода к окну конфигурации учетных данных пользователя необходимо щелкнуть правой кнопкой мыши на новом пользователе «test_user» и в появившемся окне выбрать пункт «Перейти» (рисунок 24).

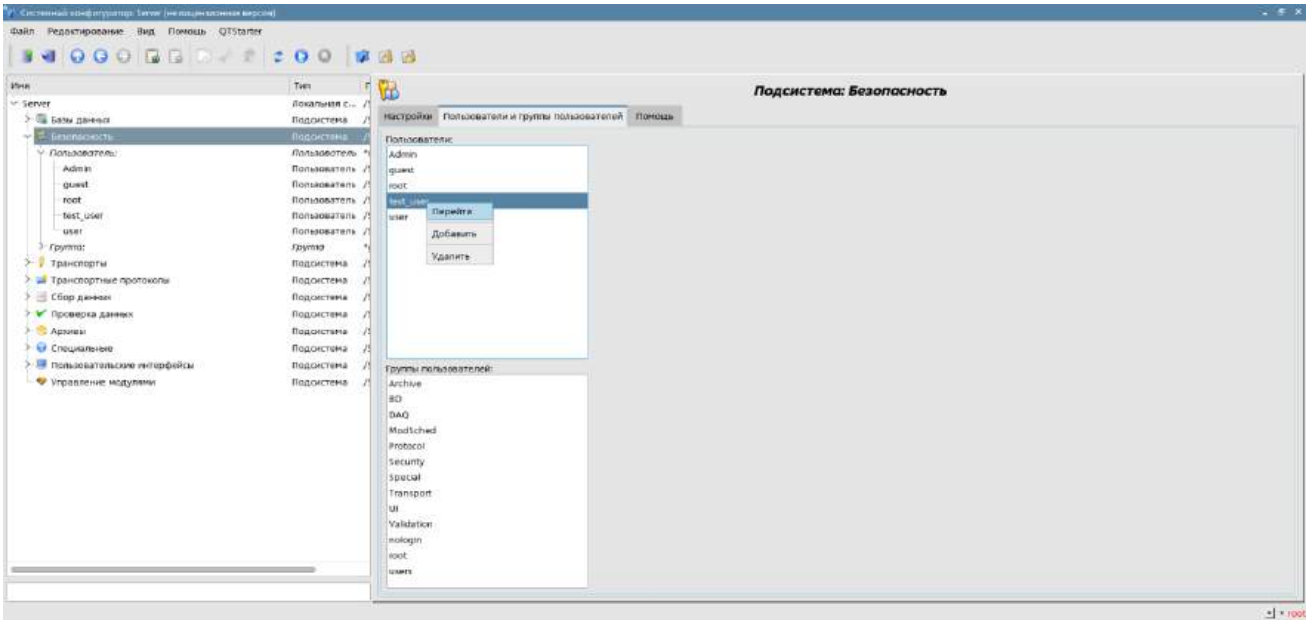


Рисунок 24

В появившемся окне производится настройка параметров нового пользователя. В поле «Полное имя» нужно ввести описание  пользователя и нажать кнопку напротив поля или клавишу «Enter» на клавиатуре.

Чтобы установить изображение пользователя нужно щелкнуть правой кнопкой мыши на поле «Изображение пользователя:» и нажать «Загрузить изображение» (рисунок 25).

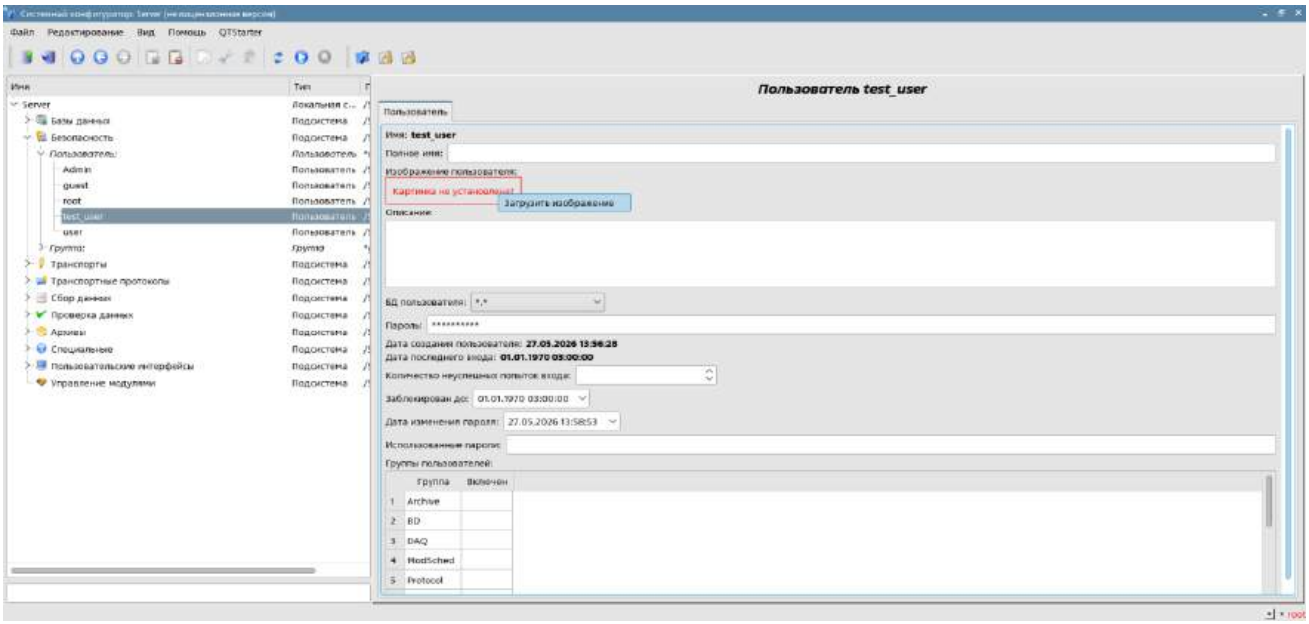


Рисунок 25

Для выбора БД, в которой будут храниться данные пользователя, необходимо щелкнуть левой кнопкой мыши по  комбобоксу в поле «БД пользователя» и в появившемся списке выбрать нужную БД (рисунок 26).

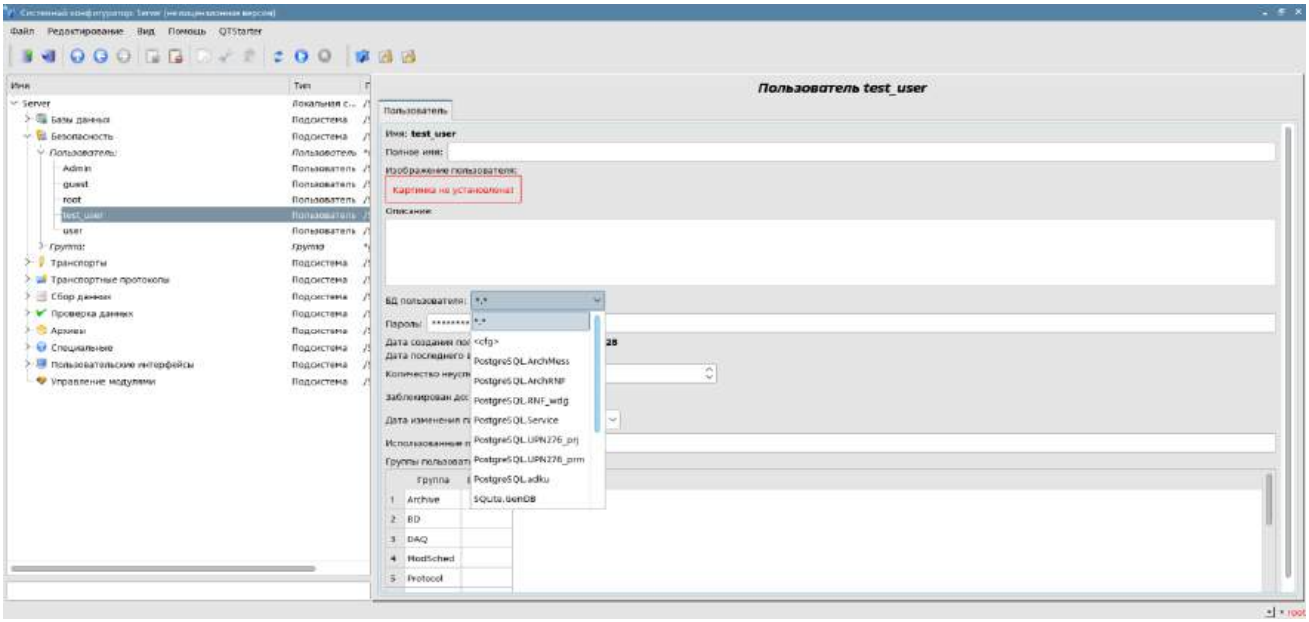


Рисунок 26

В поле пароль нужно ввести пароль для нового пользователя и нажать  кнопку напротив поля (рисунок 27) или клавишу «Enter» на клавиатуре.

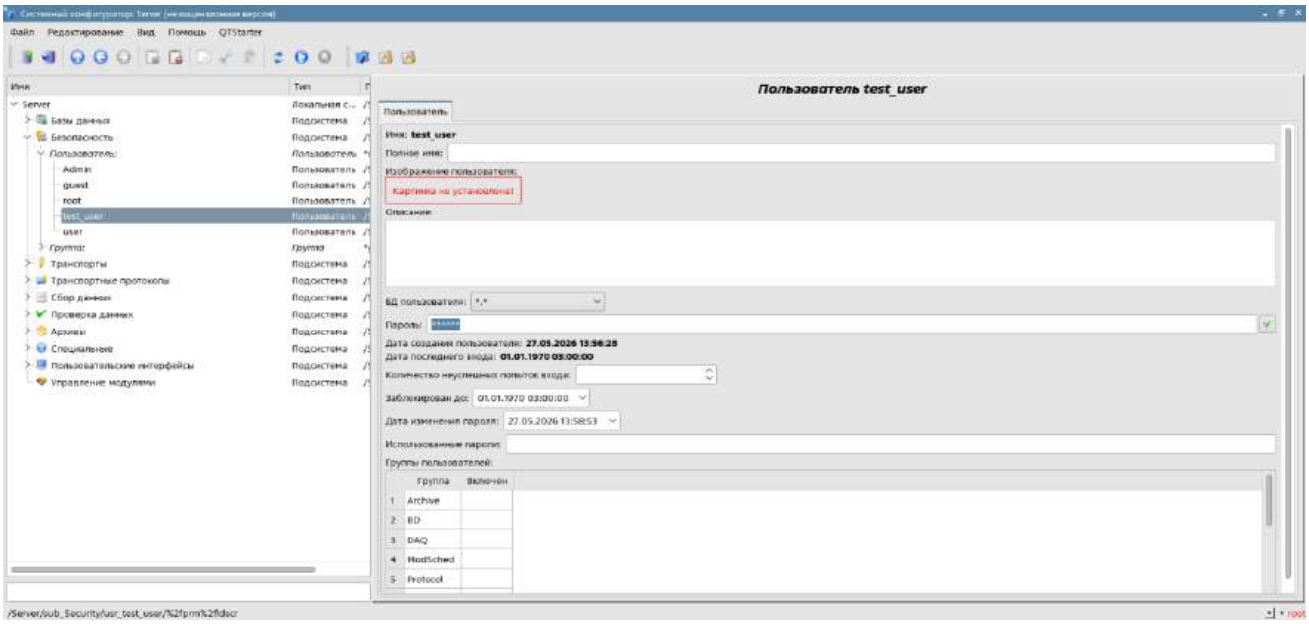


Рисунок 27

По умолчанию новый пользователь добавляется в группу «users». Чтобы добавить нового пользователя в другую группу используется поле «Группы пользователей». Например, для добавления в группу «Archive» необходимо произвести двойной щелчок левой кнопкой мыши на пустой ячейке «Включен» напротив «Archive». В ней появится комбобокс с надписью «Нет» (рисунок 28).

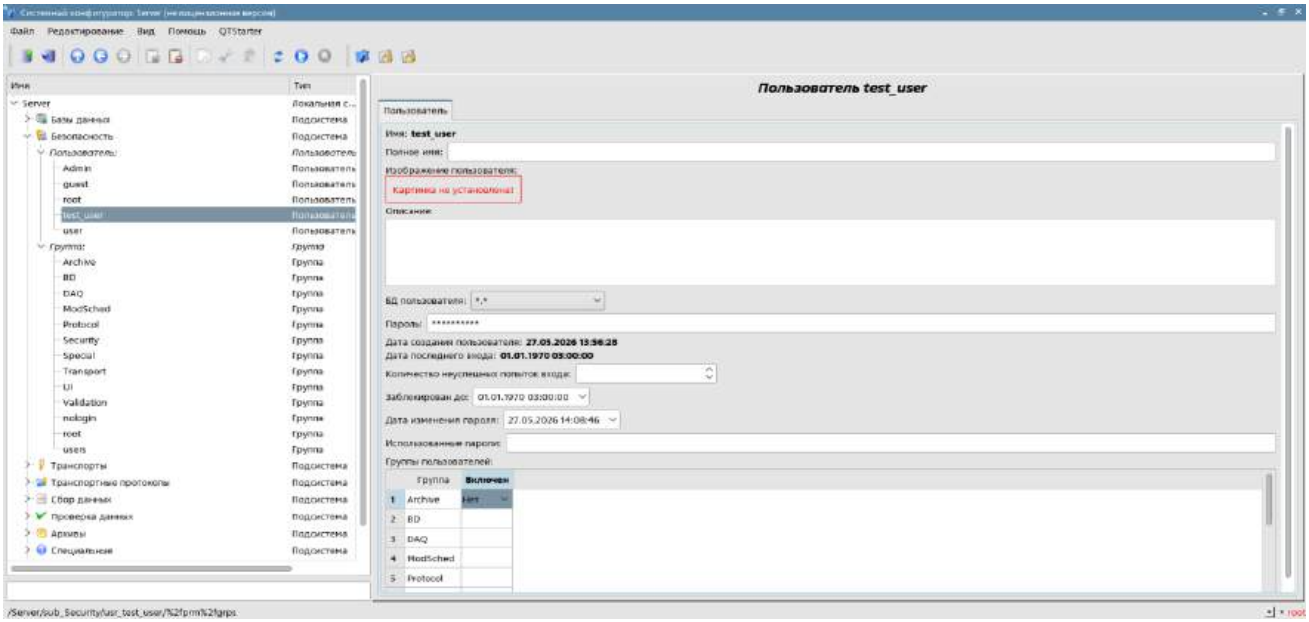
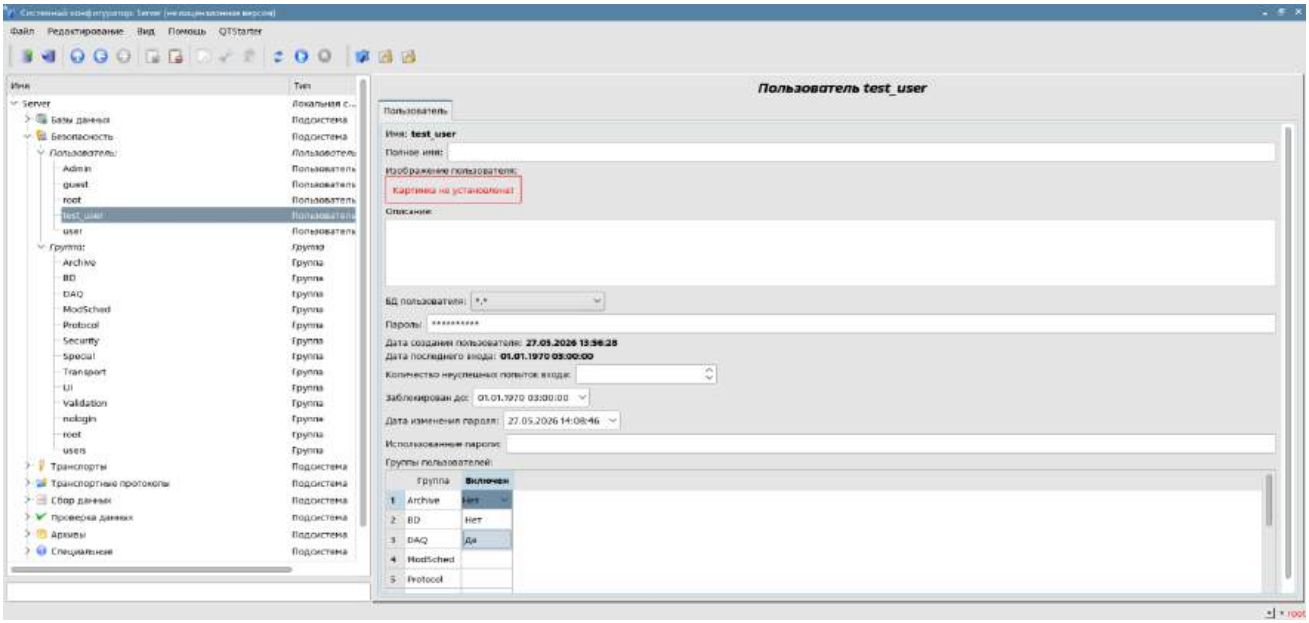


Рисунок 28

Далее нужно щелкнуть левой кнопкой мыши на этом комбобоксе и в появившемся списке выбрать «Да» (рисунок 29).



Рисуок 29

В результате пользователь «test_user» будет выбран в группу «Archive» (рисунок 30).

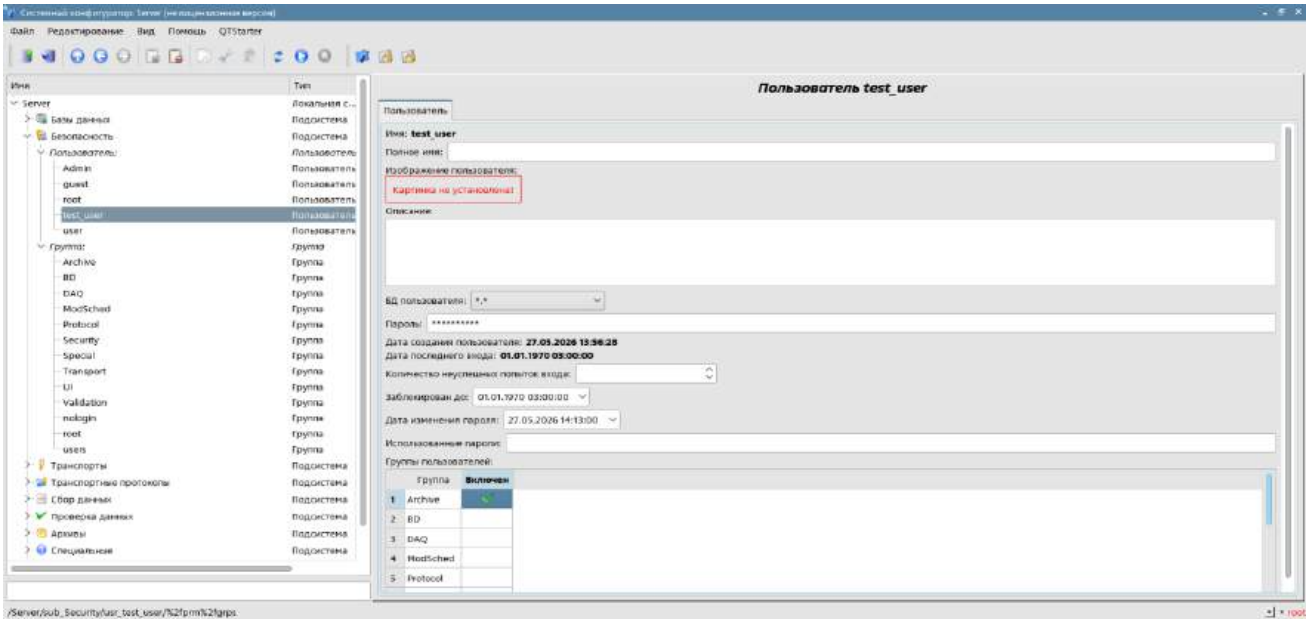


Рисунок 30

Для конфигурации группы пользователей предоставляется страница, содержащая вкладку "Группа". Ее вид показан на рисунке 31.

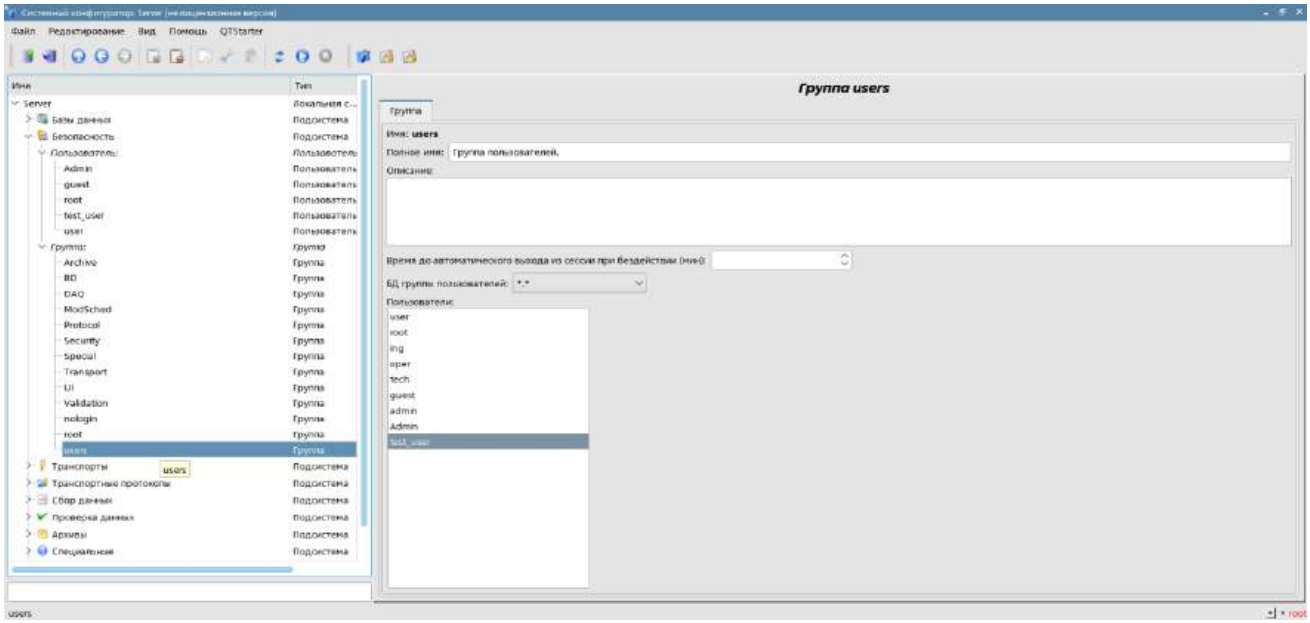


Рисунок 31

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, выбирается из списка щелчком мыши;

- *Пользователи* - список пользователей, включенных в данную группу. С помощью контекстного меню списка можно добавить или удалить пользователя в группе.

Для добавления новой группы пользователей необходимо открыть вкладку «Пользователи и группы пользователей», щелкнуть правой кнопкой мыши на поле «Группы пользователей» и нажать «Добавить» (рисунок 32).

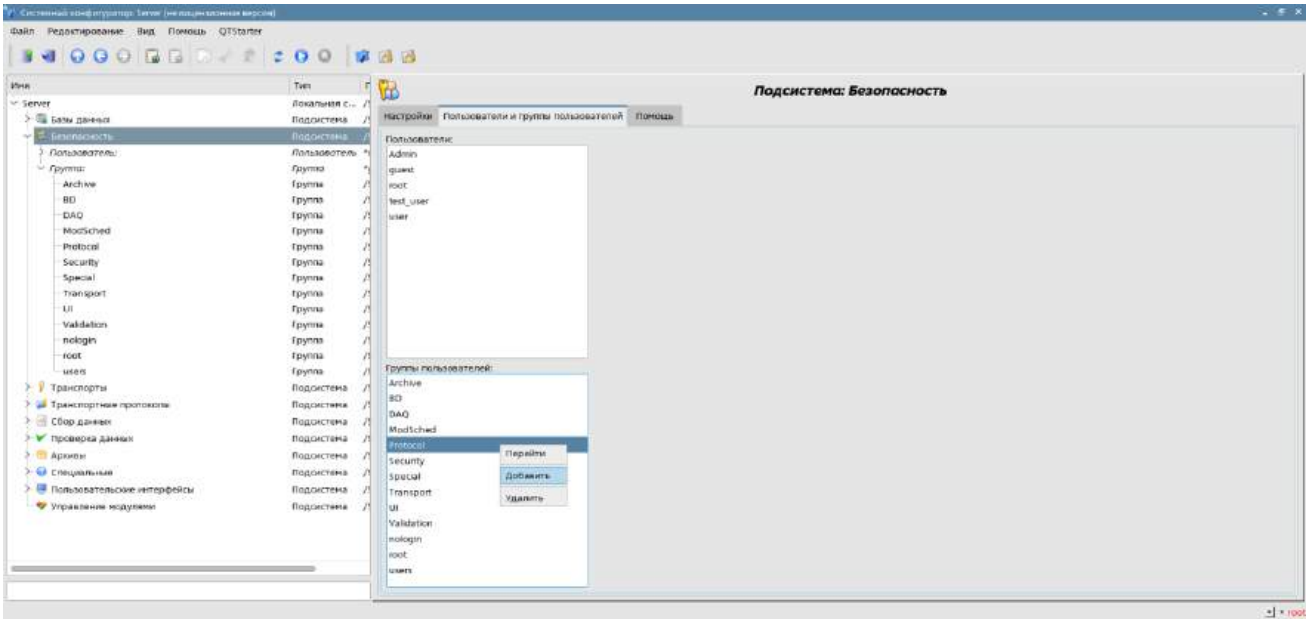


Рисунок 32

В результате будет отображено окно «Добавление нового элемента» (рисунок 33).

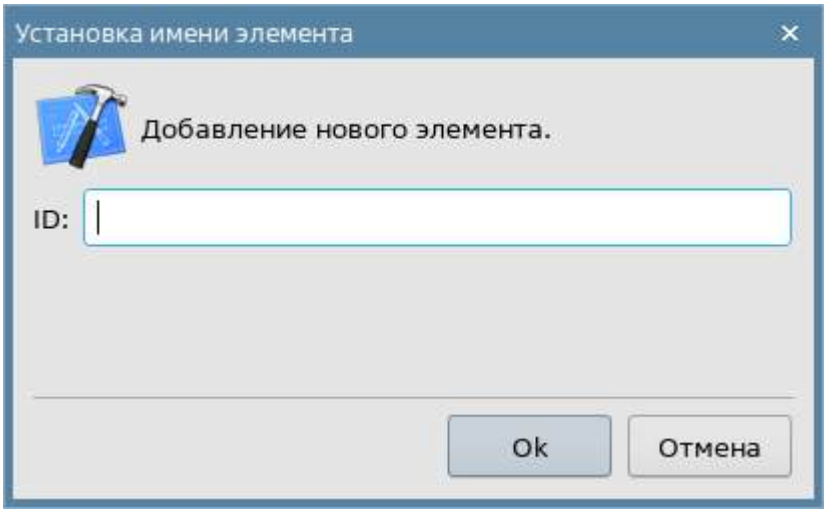


Рисунок 33

Здесь необходимо ввести имя новой группы, например «Test» (рисунок 34), а затем нажать кнопку «Ok», либо клавишу «Enter» на клавиатуре.

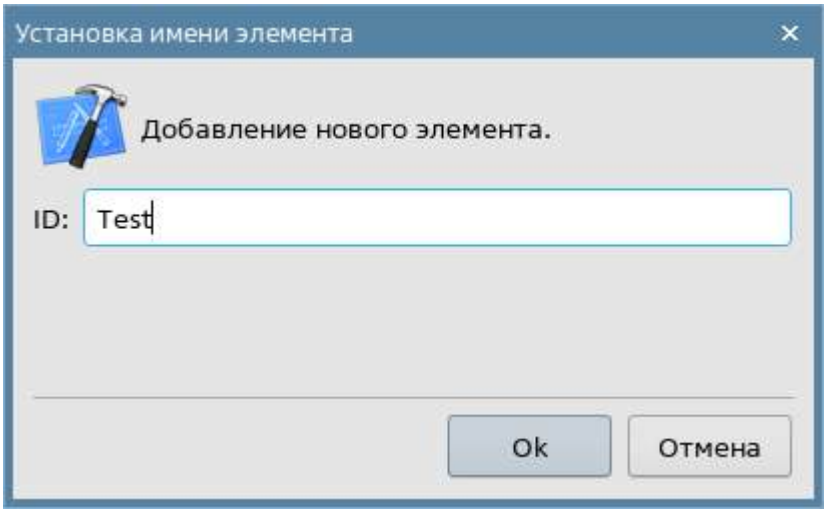


Рисунок 34

Теперь в списке групп пользователей будет отображаться имя новой созданной группы (рисунок 35).

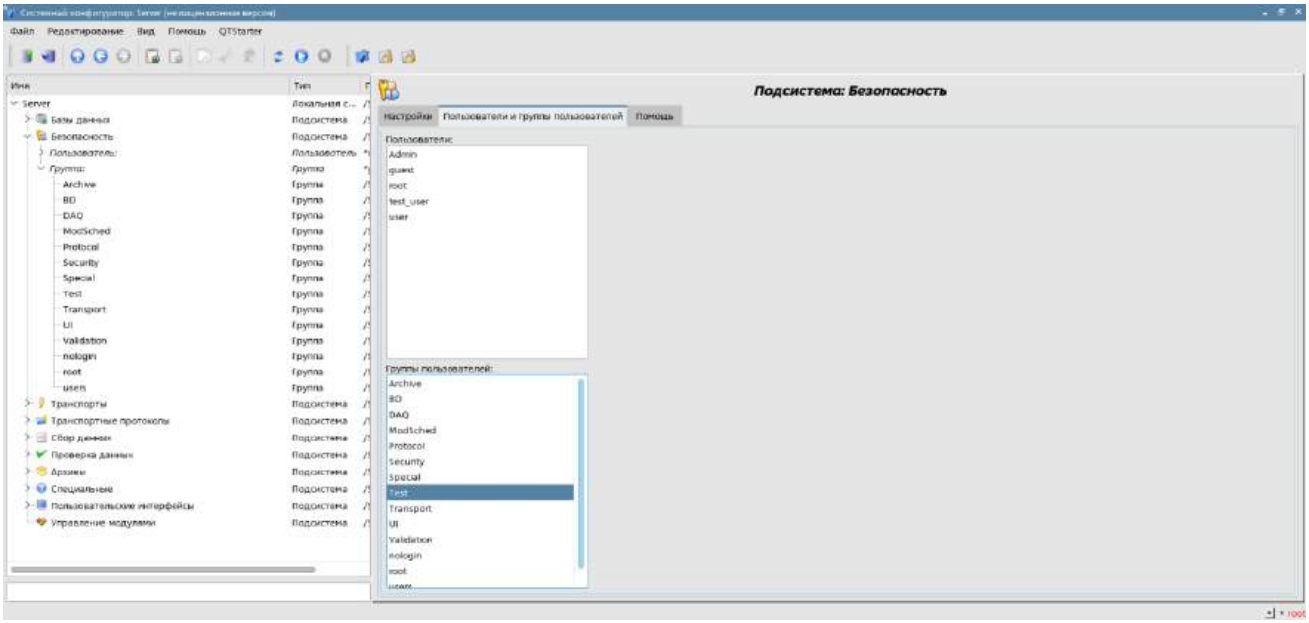


Рисунок 35

Для задания конфигурации новой группы нужно щелкнуть правой клавишей мыши по названию группы и в появившемся меню выбрать пункт «Перейти» (рисунок 36).

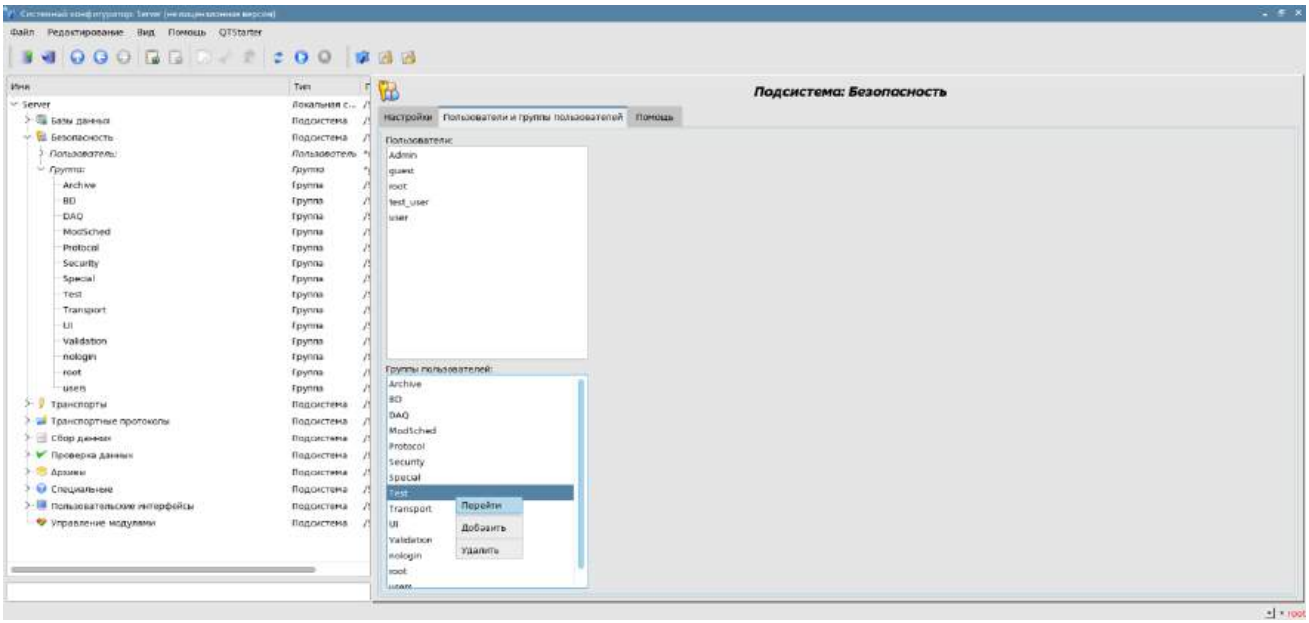


Рисунок 36

В поле «Полное имя» необходимо ввести описание новой группы, например «Группа тестовых пользователей» и нажать кнопку или клавишу «Enter» на клавиатуре (рисунок 37).

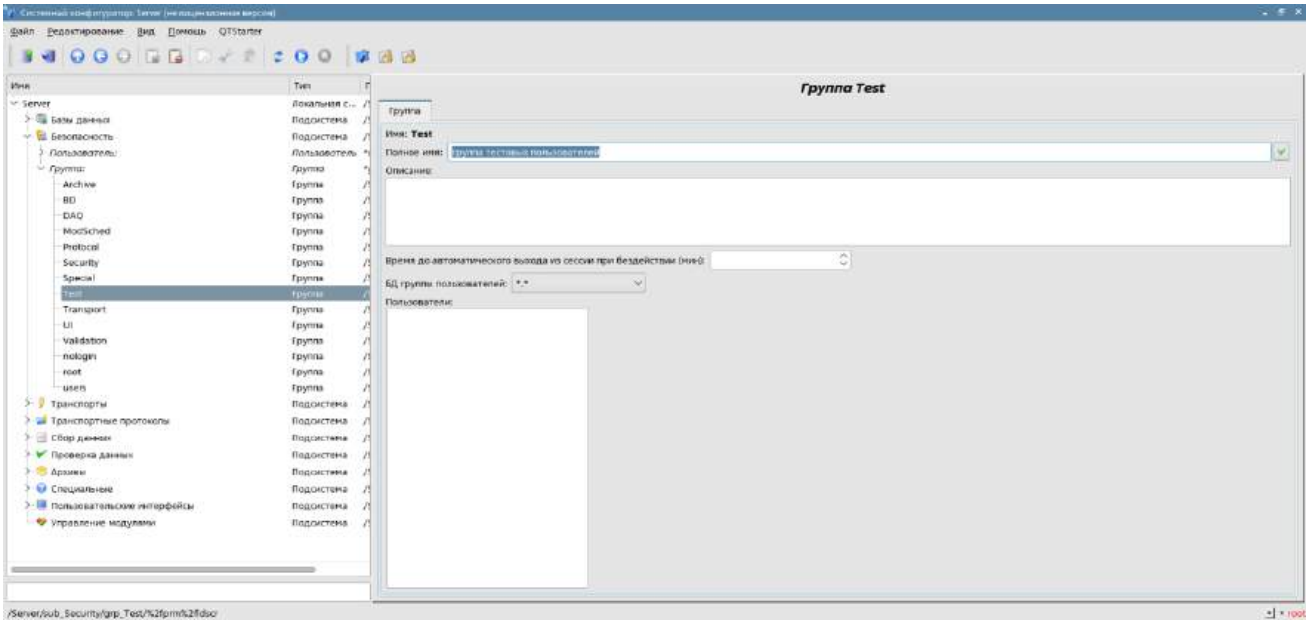


Рисунок 37

Для выбора БД, в которой будет храниться информация о новой группе, необходимо щелкнуть левой клавишей мыши по *.* комбобоксу и в появившемся списке выбрать нужную БД (рисунок 38).

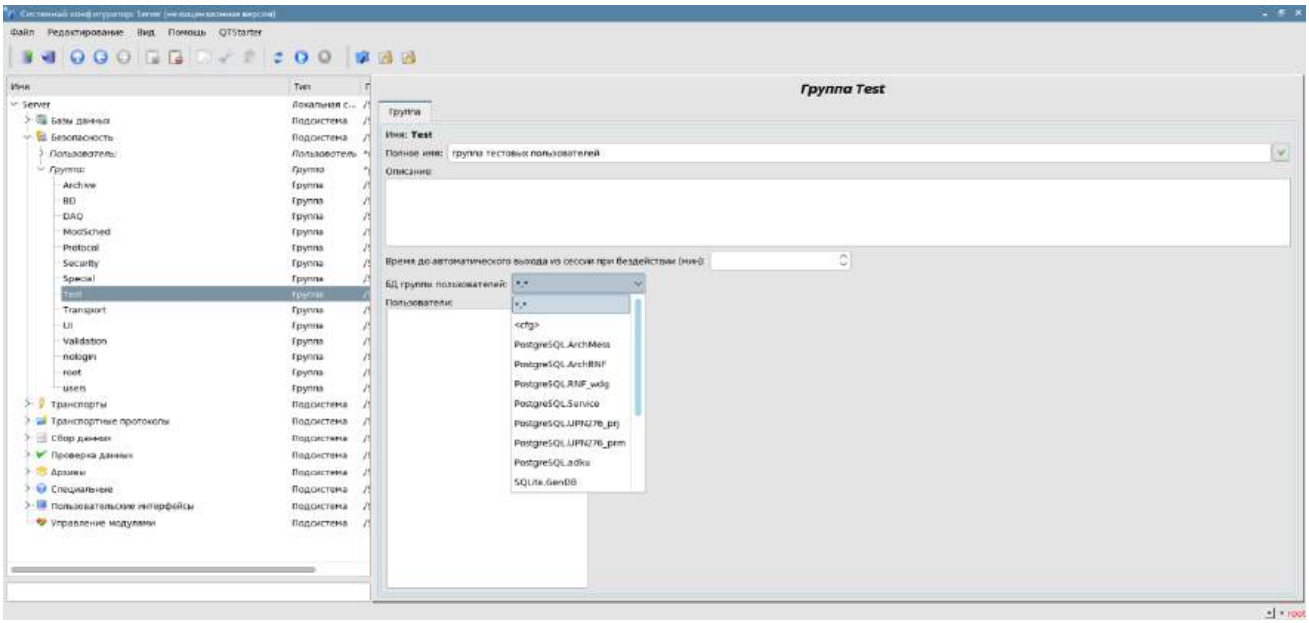


Рисунок 38

Для добавления пользователей в новую группу нужно щелкнуть правой клавишей мыши на поле «Пользователи» и нажать «Добавить» (рисунок 39).

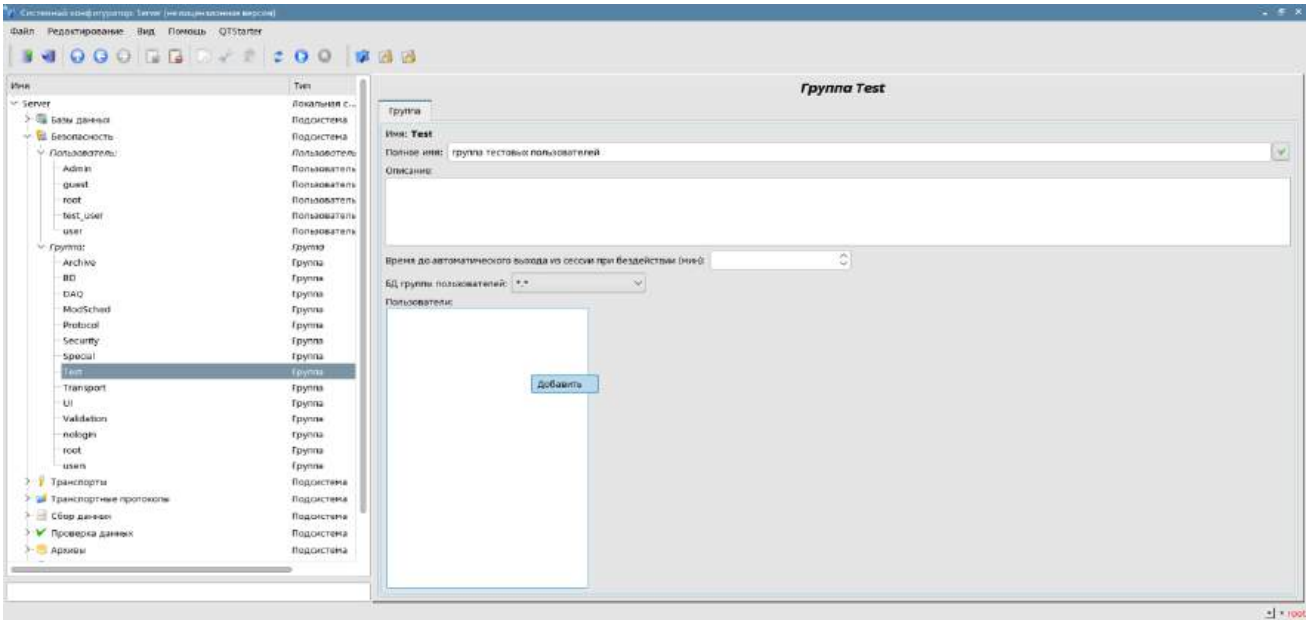


Рисунок 39

В появившемся окне «Добавление нового элемента» нужно ввести имя пользователя, например «test_user» (рисунок 40).

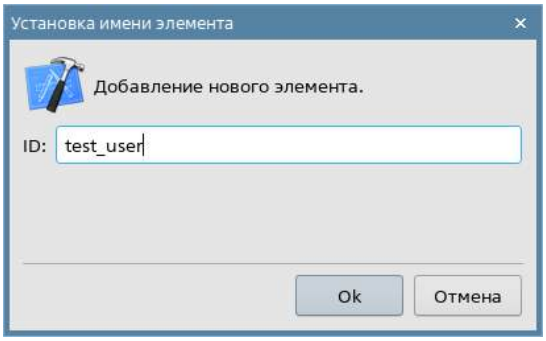


Рисунок 40

В результате пользователь «test_user» будет добавлен в группу «Test» (рисунок 41).

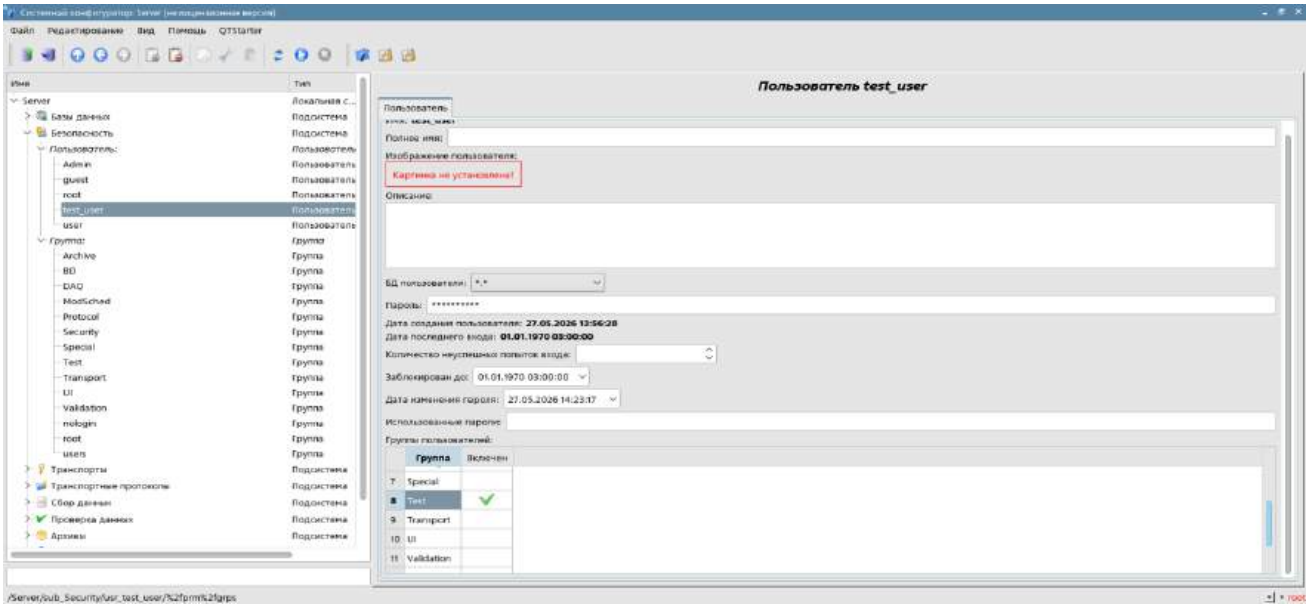


Рисунок 41

5.5 Подсистема "Транспорты"

5.5.1 Общие сведения

Подсистема «Транспорты» предназначена для обмена неструктурированными данными между СКАДА и внешними системами. В роли внешних систем могут выступать и удалённые СКАДА. 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 входных и выходных транспортов. Входной транспорт предназначен для обслуживания внешних запросов и отправки ответов. Выходной транспорт, наоборот, предназначен для отправки сообщений и ожидания ответа.

В СКАДА реализованы следующие модули транспорта:

- последовательный интерфейс;
- сокеты;
- SSL.

Тип транспорта *последовательный интерфейс* используется для обмена данными

через последовательные интерфейсы типа RS232, RS485, GSM и др. В режиме модема модулем поддерживается смешанный режим работы. Смешанный режим подразумевает наличие входного транспорта, который ожидает внешних подключений, а также выходного транспорта на том же устройстве. Т.е. входной транспорт будет игнорировать все запросы при наличии установленного выходным транспортом соединения, в то же время выходной транспорт не будет осуществлять попыток установить соединение при наличии подключения к входному транспорту или соединения другого выходного транспорта, например, с другим номером телефона.

Модуль транспорта сокеты предоставляет транспорт, основанный на сокетах. Поддерживаются входной и выходной транспорты, основанные на интернет сокетах: TCP, UDP и UNIX-сокете.

Сконфигурированный и запущенный входной 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

производится обслуживание клиента. Серверный сокет в этот момент переходит к ожиданию запросов от нового клиента. Так достигается параллельное обслуживание клиентов. Каждый входной 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.

Модуль транспорта SSL осуществляет поддержку транспортов, основанных на слое безопасных сокетов (SSL). В основе модуля лежит библиотека OpenSSL. Поддерживаются входящие и исходящие транспорты протоколов SSLv2, SSLv3 и TLSv1.

Сконфигурированный и запущенный входящий транспорт открывает серверный SSL-сокет для ожидания соединения клиентов. SSL-сокет являются многопоточными, т.е. при подключении клиента создаётся клиентское SSL-соединение и новый поток, в котором производится обслуживание клиента. Серверный SSL-сокет, в этот момент, переходит к ожиданию запросов от нового клиента. Таким образом, достигается параллельное обслуживание клиентов.

Каждый входной 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.

Сконфигурированный и запущенный выходной транспорт открывает SSL соединение с указанным сервером. При разрыве соединения, выходной транспорт отключается. Для возобновления соединения транспорт нужно опять запустить.

Для полноценной работы модуля необходимы сертификаты и приватные ключи. В случае с входным SSL-транспортом (сервером) они обязательны. В случае с выходным SSL-транспортом 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.

5.5.2 Конфигурирование подсистемы «Транспорты»

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Транспорты", содержащая вкладки "Подсистема", "Модули" и "Помощь".

Вид вкладки "Подсистема" показан на рисунке 42.

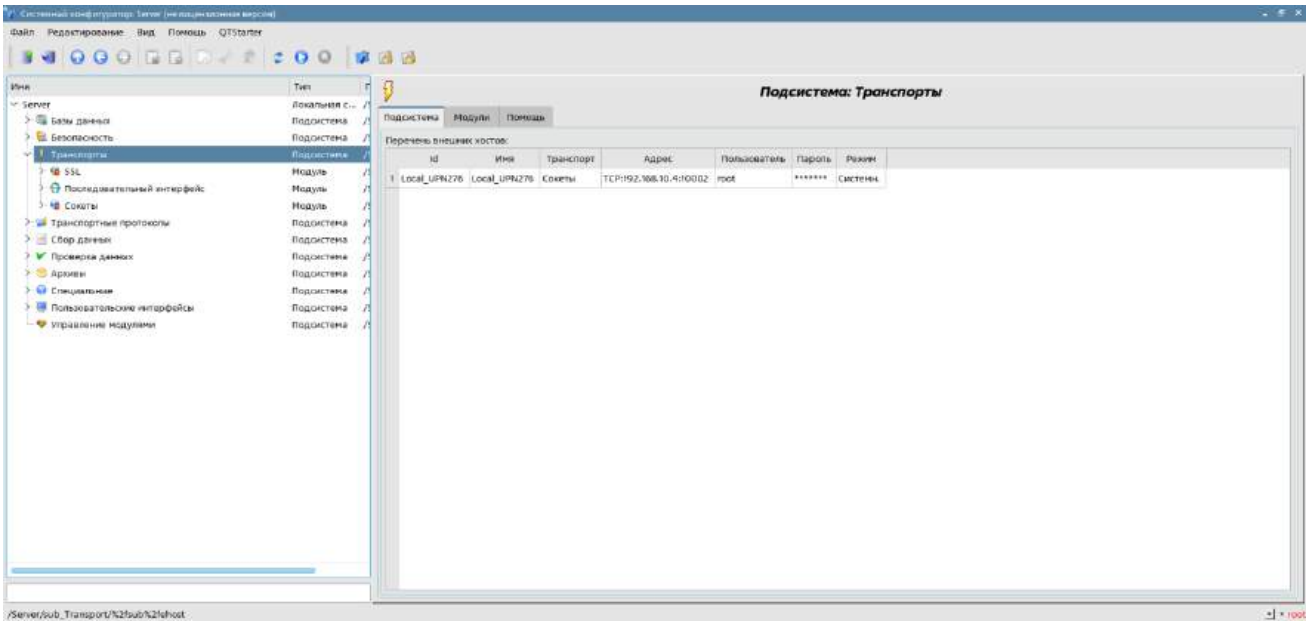


Рисунок 42

Вкладка "Подсистема" содержит таблицу конфигурации внешних для данной СКАДА 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 это значит, что список пользовательских внешних станций индивидуален для каждого пользователя.

Пользовательские внешние станции используются компонентами графического интерфейса, например, UI.QTCfg 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". Форматы адресов для различных типов транспортов приведены в таблице 3;
- *Пользователь* - имя/идентификатор пользователя удалённой станции, от имени которого выполнять подключение;
- *Пароль* - пароль пользователя удалённой станции;
- *Режим* – выбор из списка режимов:
  - пользовательский – предоставляет дерево конфигурации удаленного хоста;
  - системный – предоставляет возможность указывать подключение;
  - пользовательский и системный – совмещает два этих режима.

Таблица 3

Тип транспорта	Формат адреса
Последовательный, входной	[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'.
Последовательный, выходной	[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.
Сокеты, входной	TCP:[адрес]:[порт]:[режим] где: адрес – Адрес, на котором открывается сокет. Должен быть

Тип транспорта	Формат адреса
	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:[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:[имя]:[режим] где: имя – имя файла UNIX сокета; режим – тоже что в TCP. Пример: <UNIX:/tmp/scada:1> – UNIX-сокет доступен через файл /tmp/oscada и соединения не разрывает.
Сокеты, выходной	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:[имя] где: имя – имя файла UNIX сокета. Пример: <UNIX:/tmp/scada> – соединится с UNIX-сокетом через файл /tmp/scada.
SSL, входной	[адрес]:[порт]:[режим]

Тип транспорта	Формат адреса
	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
SSL, выходной	[адрес]:[порт]:[режим] 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

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

Каждый модуль подсистемы "Транспорты" предоставляет конфигурационную страницу с вкладками "Транспорты" и "Помощь". Вид вкладки "Транспорты" показан на рисунке 43.

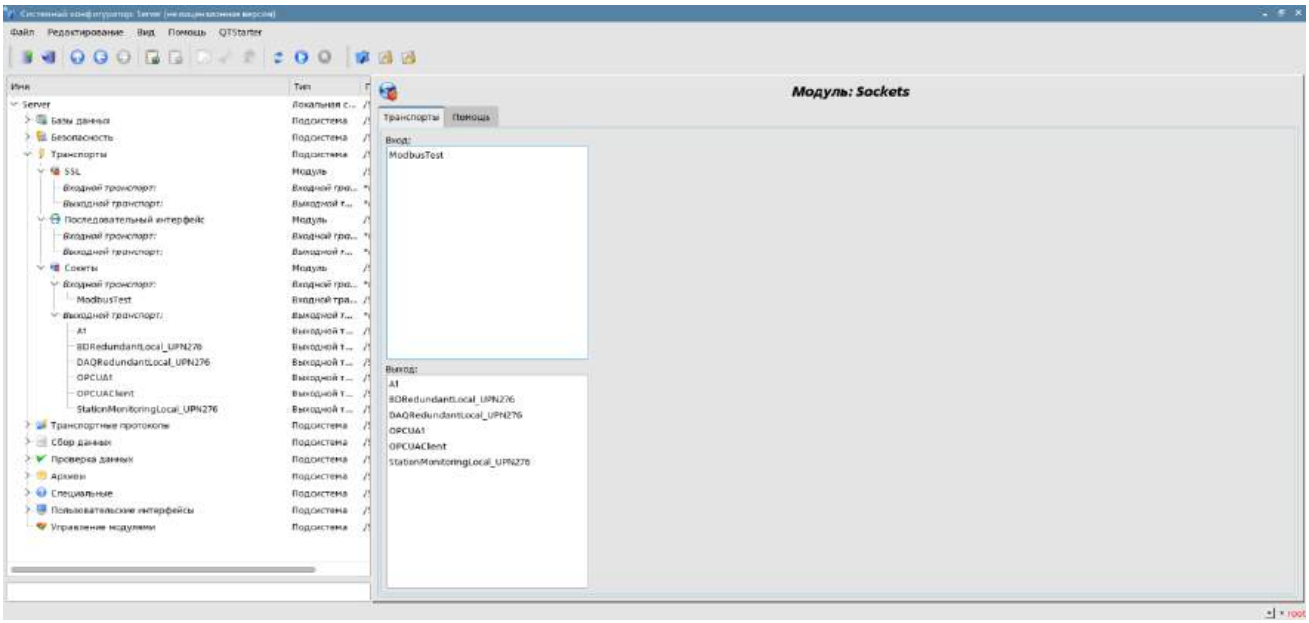


Рисунок 43

Вкладка "Транспорты" содержит список входных и выходных 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", состав которой идентичен для всех модулей.

Каждый транспорт содержит собственную страницу конфигурации с одной вкладкой "Транспорт". Эта вкладка содержит основные настройки транспорта. Вид страницы входного транспорта показан на рисунке 44.

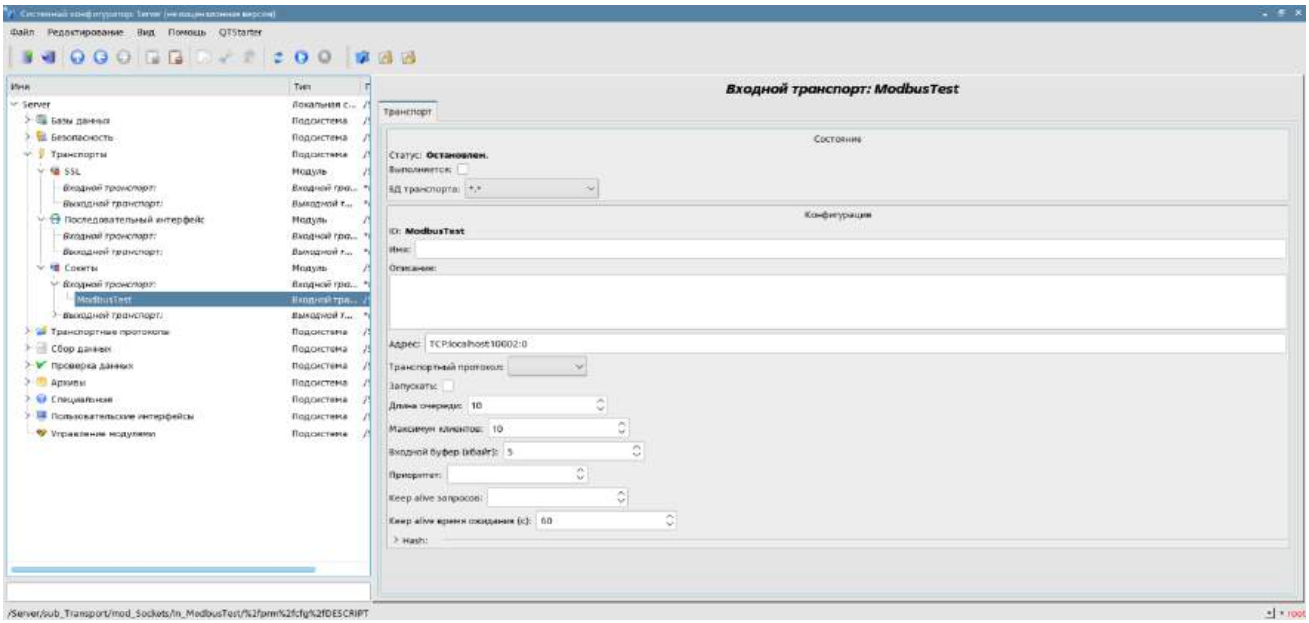


Рисунок 44

Входной транспорт 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, выбирается из списка щелчком мыши.
- 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 (см. таблицу 3);

- *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 следует перевести транспорт при загрузке;
- *Длина очереди* - размер очереди сокетов;
- *Максимум клиентов* - максимальное количество обслуживаемых клиентов;
- *Входной буфер* - размер входного буфера;
- *Приоритет* – уровень приоритета (0 – стандартный пользовательский приоритет);
- *Keep alive запросов* – закрытие подключения после указанного количества запросов (если задан 0, то никогда не закрывать);
- *Keep alive время ожидания* (с) - закрытие подключения после отсутствия запросов в течении указанного времени (если указан ноль, то никогда не закрывать).

Вид страницы выходного транспорта показан на рисунке 45.

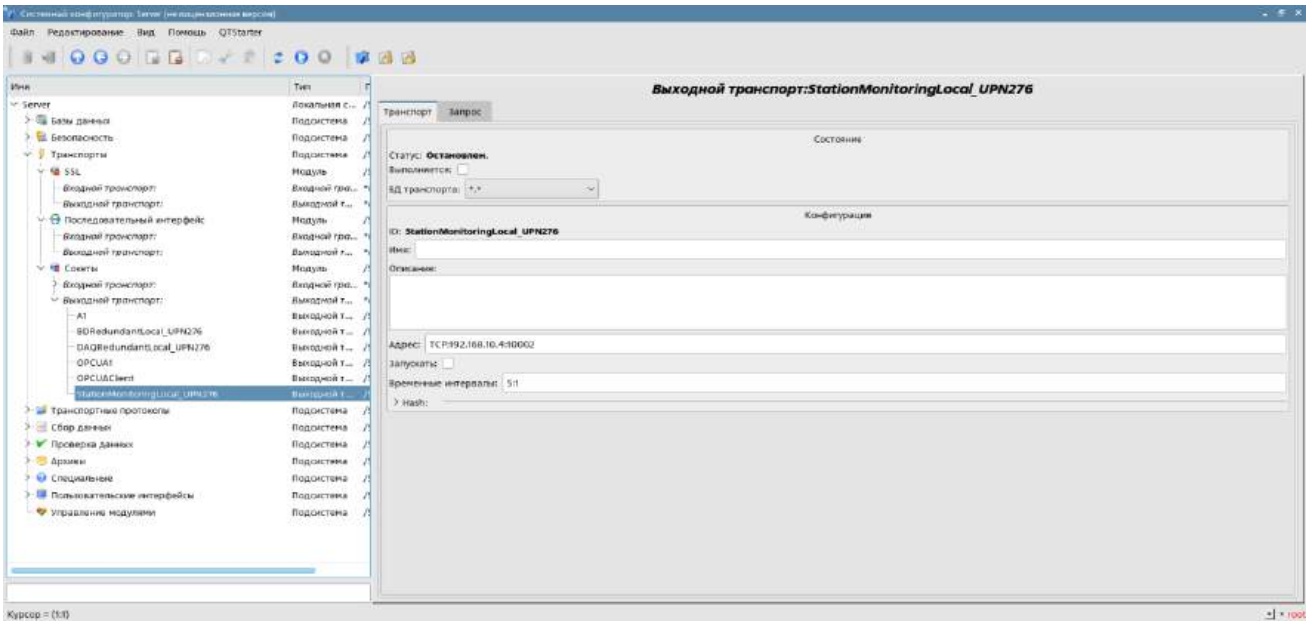


Рисунок 45

Выходной транспорт 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, выбирается из списка щелчком мыши.

- 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 (см. таблицу 3);

- *Запускать* - указывает на состояние "Выполняется", в которое следует перевести транспорт при загрузке;

- *Временные интервалы* - 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.

Выходной транспорт, в дополнение, предоставляет вкладку формирования пользовательского запроса через данный транспорт. Вид вкладки показан на рисунке 46.

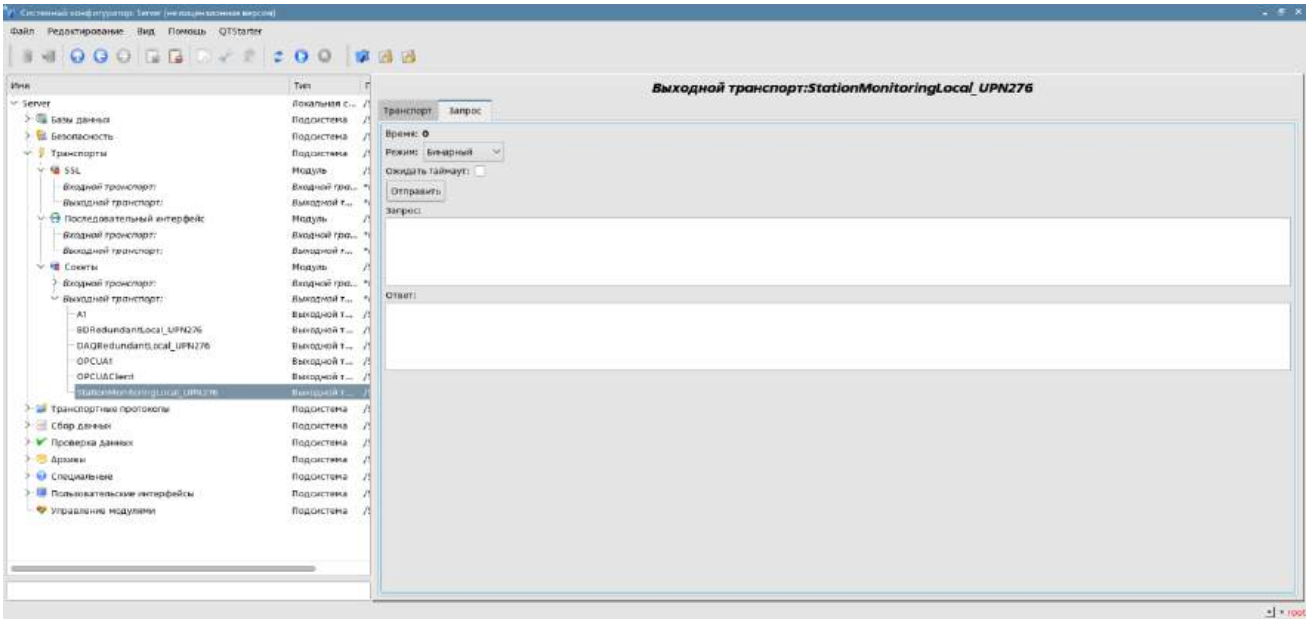


Рисунок 46

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;

- *Ожидать таймаут* – признак ожидания таймаута при получении ответа. При установке этого флага будет осуществляться ожидание всех кусков ответа, вплоть до отсутствия данных в течение таймаута дожидания ответа (например, протокол HTTP).

- *Отправить* - команда отправить запрос;

- *Запрос* - содержит запрос в выбранном режиме представления данных;

- *Ответ* - предоставляет ответ в выбранном режиме представления данных.

5.6 Подсистема "Транспортные протоколы"

5.6.1 Общие сведения

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:

OPC UA;

Собственный протокол системы СКАДА;

ModBUS;

HTTP;

Пользовательский протокол.

Полную цепочку связи можно описать 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.

5.6.2 Модуль «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 определённых OPC UA типов данных.

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.

В СКАДА модуль OPC UA содержит код реализации протокольной части OPC UA как для клиентского, так и для серверного сервисов. Для построения OPC UA сервера достаточно создать входной транспорт, обычно это TCP-транспорт модуля Sockets, и выбрать в нём модуль данного протокола (рисунок 48), а также сконфигурировать хотя бы один конечный узел модуля протокола.

Входной транспорт создается в соответствии с пунктом 5.5.2:

- в модуле «Транспорты» добавить входной транспорт «Сокет»;

- в качестве транспортного протокола выбрать OPC UA;

- указать адрес сервера и порт соединения;

- сохранить сделанные изменения в БД.

На рисунке 47 представлено окно настройки входного транспорта «opcServer».

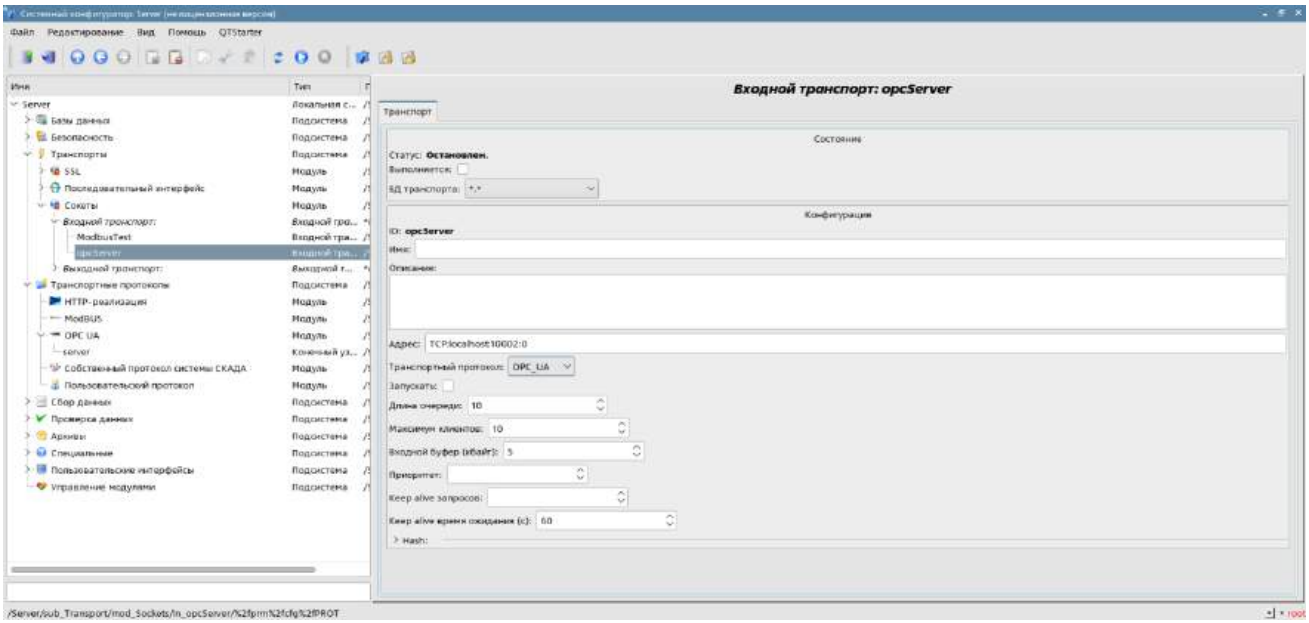


Рисунок 47

Входящие запросы к модулю-протоколу обрабатываются модулем в соответствии со сконфигурированными конечными узлами OPC UA.

Конечный узел протокола OPC UA - это фактически объект сервера OPC UA. В СКАДА реализованы только локальные конечные узлы. Локальный конечный узел предназначен для предоставления ресурсов СКАДА-станции по протоколу OPC UA.

Общая конфигурация конечного узла осуществляется на главной вкладке страницы «Транспортные протоколы» 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 ("Бинарный");

URL конечной точки – адрес сервера и порт соединения;

сертификат сервера в формате PEM - поле обязательное для заполнения;

приватный ключ в формате PEM - поле обязательное для заполнения;

политики безопасности сервера – выбирается значение None(нет).

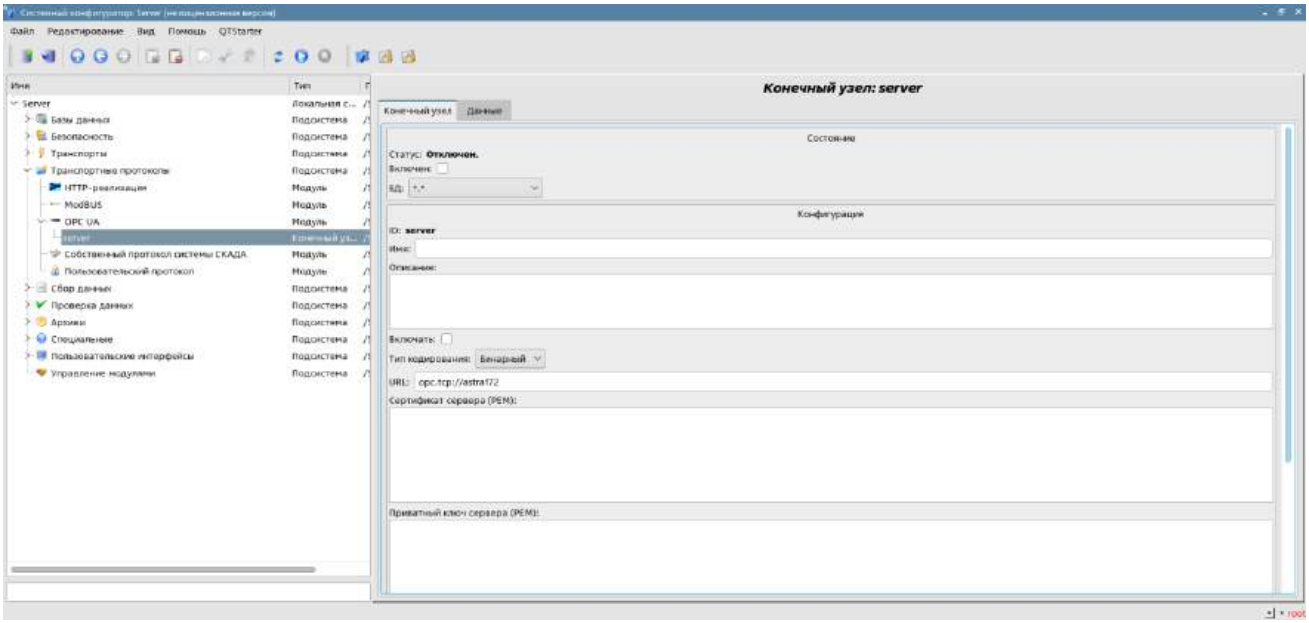


Рисунок 48

Настройка клиента OPC UA осуществляется в одноименном модуле подсистемы «Сбор данных».

5.6.3 Модуль «HTTP»

Модуль транспортного протокола HTTP предназначен для реализации поддержки
сетевого протокола HTTP (Hypertext Transfer Protocol) в системе СКАДА.

Протокол HTTP используется для передачи содержимого WWW. Так, через HTTP
передаются следующие типы документов: html, xhtml, png, java и другие. Добавление поддержки HTTP в СКАДА 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 используются модули подсистемы “Пользовательские интерфейсы” с дополнительным информационным полем “SubType”, имеющим значение “WWW”.

В запросах к Web ресурсам используется URL (Universal Resource Locator), который 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.

5.6.4 Модуль «ModBUS»

ModBUS — коммуникационный протокол, основань на клиент-серверной архитектуре для использования в контроллерах с программируемой логикой (PLC). 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.

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 «ModBUS» подсистемы «Сбор данных» предоставляет возможность собирать информацию у различных устройств по протоколу ModBUS во всех режимах, а 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".

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 определяется в зависимости от режима. В режиме RTU окончание посылки определяется по временному интервалу между окончанием приёма предыдущего байта и началом приёма следующего, время символа. Если этот интервал превысил время, необходимое для приёма полтора байта на заданной скорости передачи то приём фрейма ответа считается завершённым. В режиме ASCII критерием окончания посылки является символ '\r', а в

режиме TCP — ожидаемый размер посылки, информация о котором присутствует в заголовке пакета.

Все операции с данными привязаны к нулю, каждый вид данных (регистр, бит, регистр входа или бита входа) начинаются с адреса 0000 и заканчиваются 65535.

В протоколе ModBUS можно выделить несколько подмножеств команд:

- страндартные – коды 1 - 21;
- резерв для расширенных функций – коды 22-64;
- пользовательские – коды 65-119;
- резерв для внутренних нужд – коды 120-255.

Модулем сбора данных используются команды 0x03 и 0x06(0x10) для чтения и записи регистров, 0x01 и 0x05(0x0F) для чтения и записи битов, 0x02 и 0x04 для чтения бита и регистра входа соответственно.

Модуль протокола обрабатывает запросы командами 0x03 и 0x06(0x10) для чтения и записи регистров, 0x01 и 0x05(0x0F) для чтения и записи битов.

Входная часть обслуживания запросов к модулю протокола осуществляет проверку и обработку запросов посредством объектов узлов. Узел протокола эквивалентен физическому узлу устройства сети ModBUS. Узел протокола может работать в трёх режимах:

"Данные" — режим отражения данных системы СКАДА 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 СКАДА-станция может прозрачно представлять несколько узлов сети ModBUS с различными данными.

5.6.5 Модуль «Собственный протокол системы СКАДА» (SelfSystem)

Модуль транспортного протокола SelfSystem предназначен для отражения интерфейса управления СКАДА-системы в сеть с целью предоставления возможности внешним системам взаимодействовать с системой СКАДА, а также для взаимодействия станций, построенных на основе СКАДА между собой. Модуль используется для обеспечения удалённого конфигурирования одной СКАДА станции из другой через сеть посредством модуля конфигурирования QTCfg.

5.6.6 Модуль «Пользовательский протокол» (UserProtocol)

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

Основная цель модуля - упростить задачу подключения к системе СКАДА 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 СКАДА. Для реализации этого предоставляется механизм формирования протокола исходящего запроса.

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

Модуль предоставляет возможность создания реализаций множества различных протоколов в объекте "Пользовательский протокол" (рисунок 49).

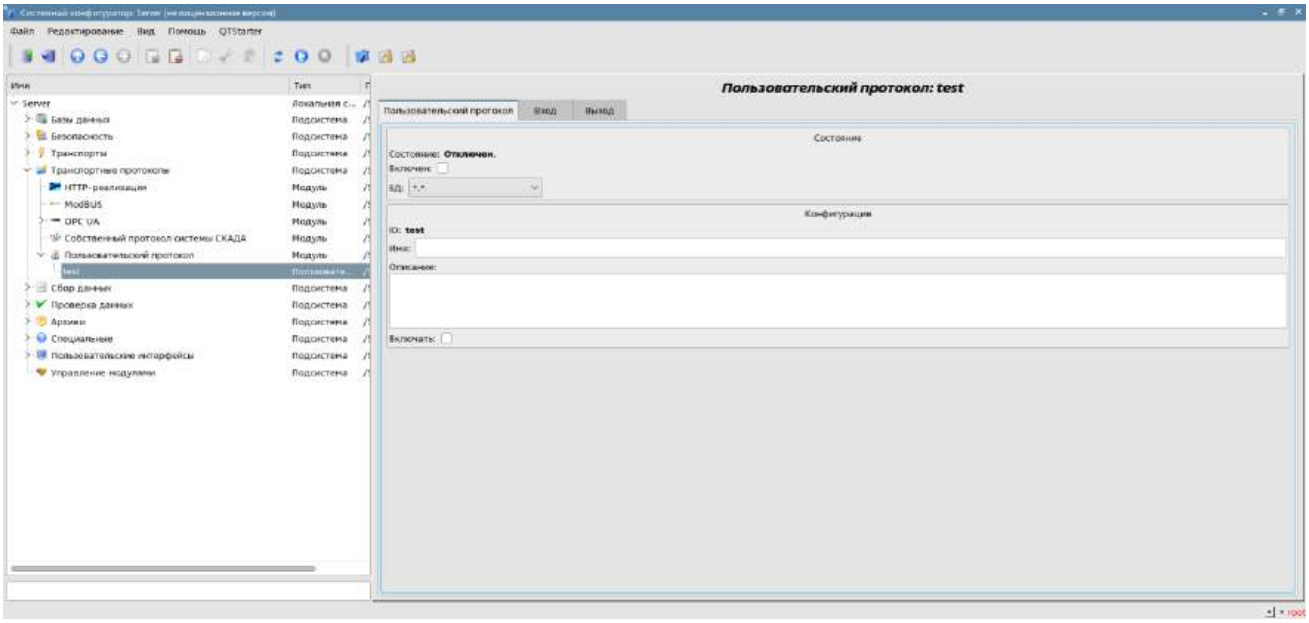


Рисунок 49

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.

5.6.7 Конфигурирование подсистемы «Транспортные протоколы»

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Транспортные протоколы", содержащая вкладки "Модули" и "Помощь".

Вкладка "Модули" (рисунок 50) содержит список модулей подсистемы "Транспортные протоколы".

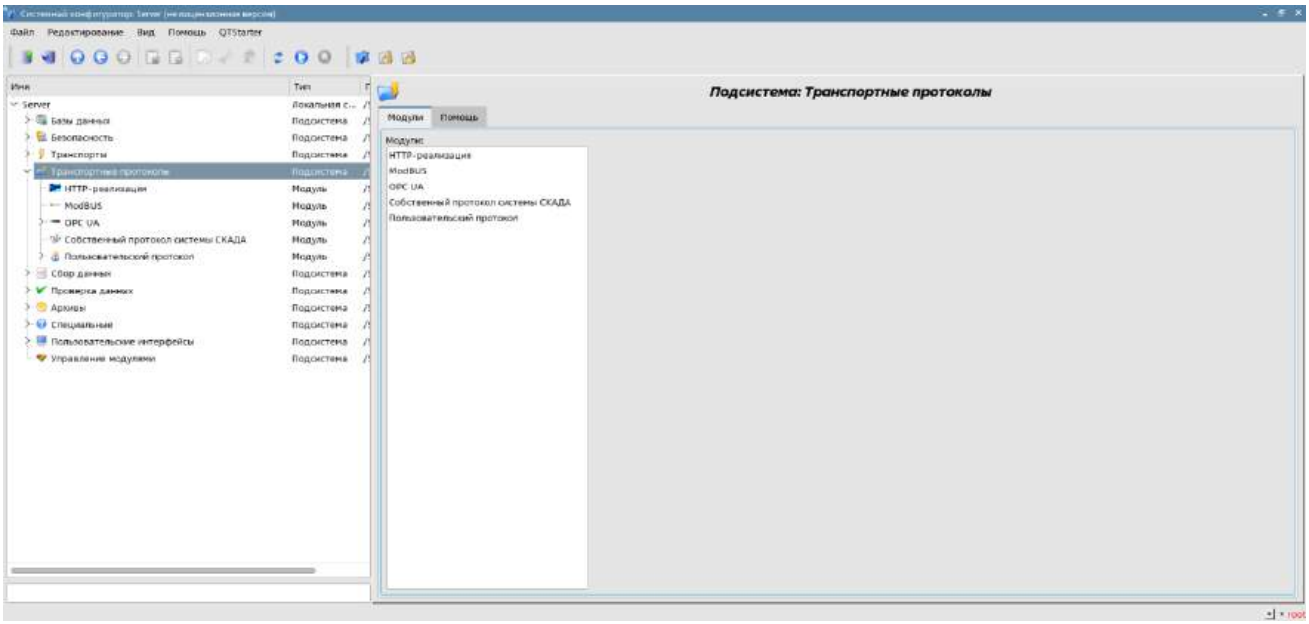


Рисунок 50

Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждый модуль подсистемы "Транспортные протоколы" предоставляет конфигурационную страницу с индивидуальными настройками в соответствии с выбранным протоколом, а также содержит вкладку "Помощь" с информацией о модуле подсистемы (рисунок 51).

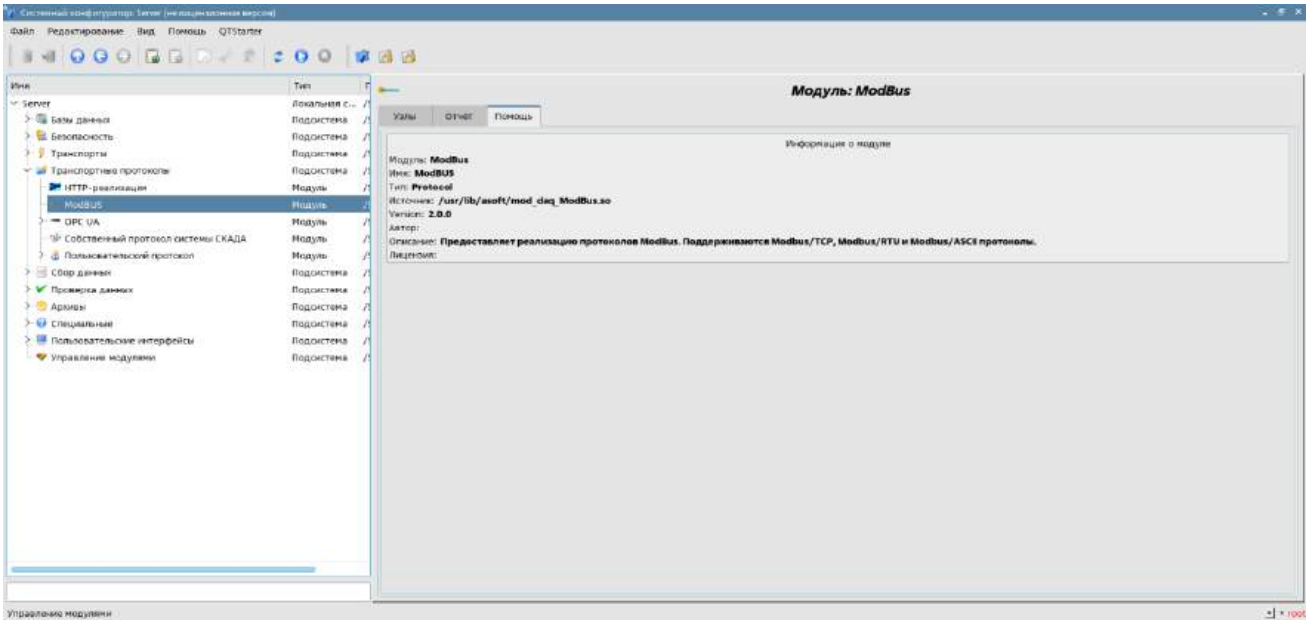


Рисунок 51

5.7 Подсистема "Сбор данных"

5.7.1 Общие сведения

Подсистема «Сбор данных» предназначена для обеспечения поддержки источников динамических данных, будь то PLC-контроллеры, платы УСО, виртуальные источники и т.д., а также различных механизмов взаимодействия в СКАДА. В функции этой подсистемы входит предоставление полученных данных в структурированном виде и обеспечение управления этими данными, например, модификация данных.

Подсистема 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" СКАДА.

В настоящее время поддерживаются следующие типы источников данных:

сбор данных операционной системы (ОС);

блочный вычислитель;

вычислитель на Java-подобном языке;

шлюз приема данных по изменению от удаленных СКАДА станций в локальную;

шлюз источников данных подсистемы "Сбор данных" от одной СКАДА станции к другой;

протокол HART;

протокол ModBUS;

протокол OPC UA;

IEC 60870-5-104 клиент;

сбор данных сетевых устройств посредством протокола SNMP;

источник данных логического уровня СКАДА.

Каждый тип источника выполнен в виде отдельного модуля, который может быть подключен/отключен.

В терминах системы СКАДА 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.

Отдельно взятый контроллер может содержать параметры определённых модулем типов. Например, параметры аналогового типа: основной информацией, которую они предоставляют, 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.

Источник динамических данных может быть удалённым, т.е. быть подключен на удалённой СКАДА-системе.

5.7.2 Конфигурирование подсистемы «Сбор данных»

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Сбор данных", содержащая вкладки "Резервирование", "Задачи событий по изменению", "Библиотеки шаблонов", "Модули" и "Помощь". Вкладка "Задачи событий по изменению" в настоящее время не используется.

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

На рисунке 52 представлен вид вкладки "Резервирование" подсистемы "Сбор данных".

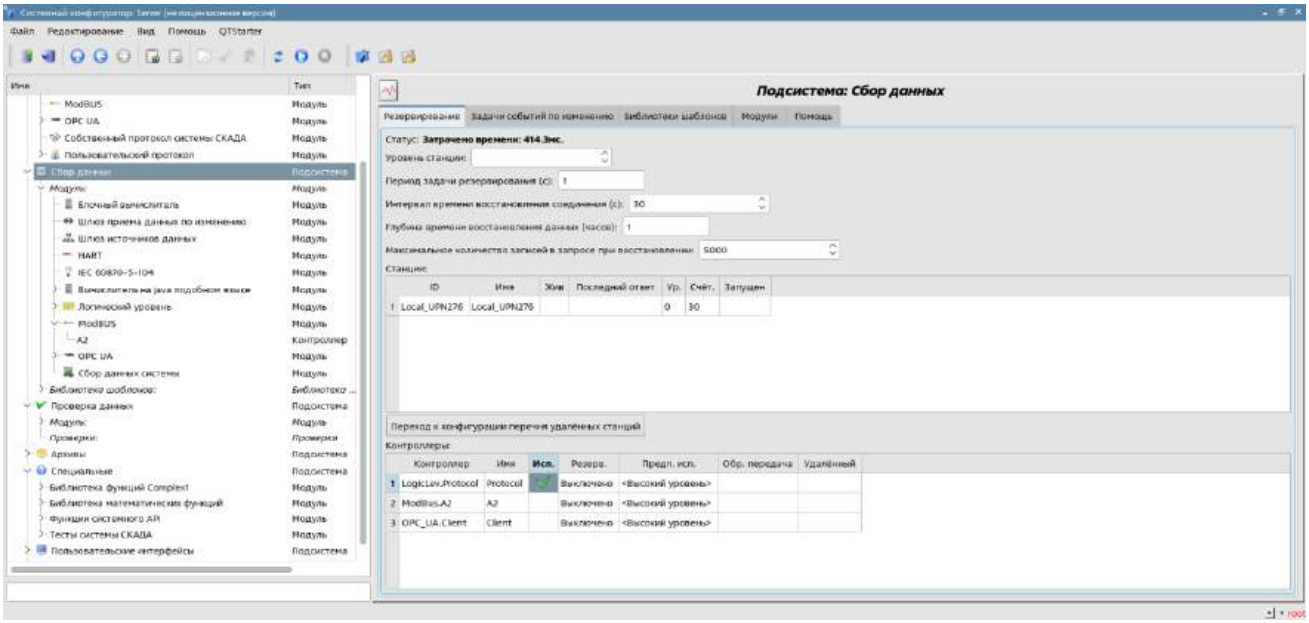


Рисунок 52

Вкладка "Резервирование" 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 интервал времени, через который возможно осуществить 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 СКАДА. Таблица предоставляет следующую информацию о станции:

ID - идентификатор системной станции СКАДА, должен быть изменён после добавления путём выбора из перечня доступных идентификаторов;

Имя - имя системной станции СКАДА;

Жив - признак наличия связи с резервной станцией;

Last time – время последнего соединения 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 СКАДА 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 выбран из перечня: "Выключен", "Асимметричное" и "Архивное";

- *Предпочтение исполнения* - конфигурация предпочтительного исполнения на указанной станции, может иметь следующие значения:

<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.

Вкладка "Библиотеки шаблонов" содержит список библиотек шаблонов для параметров этой подсистемы. Вид вкладки показан на рисунке 53.

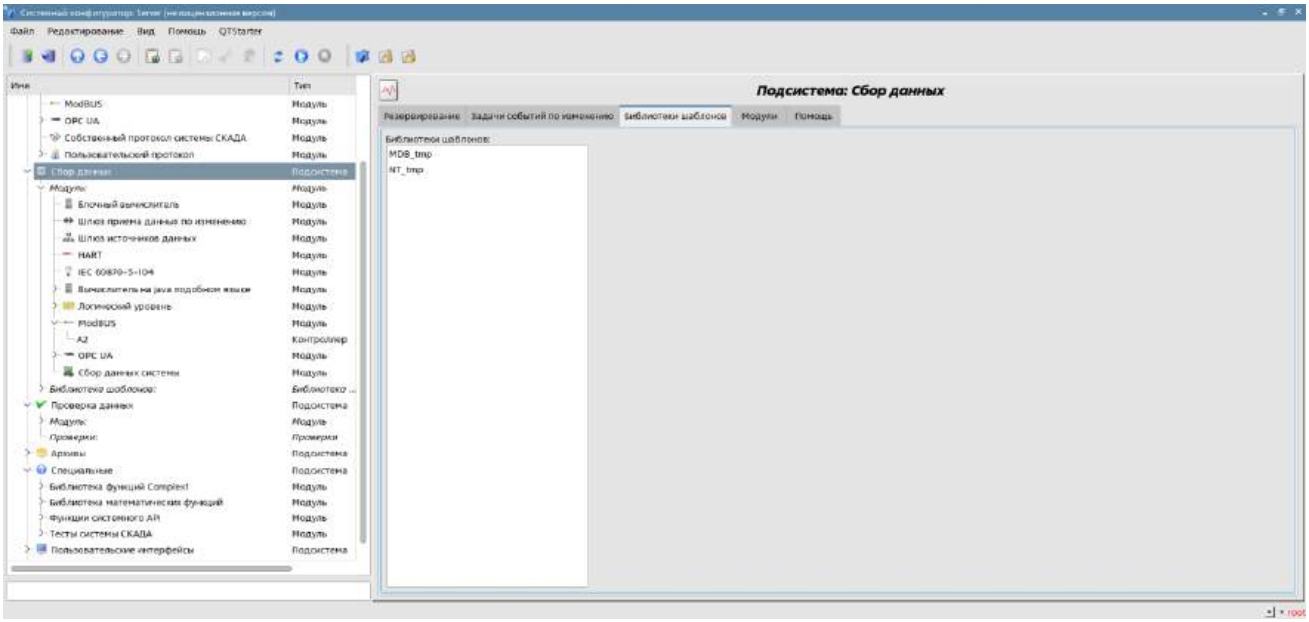


Рисунок 53

В контекстном меню списка библиотек шаблонов пользователю предоставляется возможность добавления, удаления и перехода к нужной библиотеке.

Вкладка "Модули" содержит список модулей подсистемы "Сбор данных" и идентична для всех модульных подсистем.

Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждая библиотека шаблонов подсистемы "Сбор данных" предоставляет конфигурационную страницу с вкладками "Библиотека" и "Шаблоны параметров". Вкладка "Библиотека" (рисунок 54) 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, выбирается из списка щелчком мыши.

- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:

- *ID* - информация об идентификаторе библиотеки;

- *Имя* - указывает имя библиотеки;

- *Описание* - краткое описание библиотеки и её назначения.

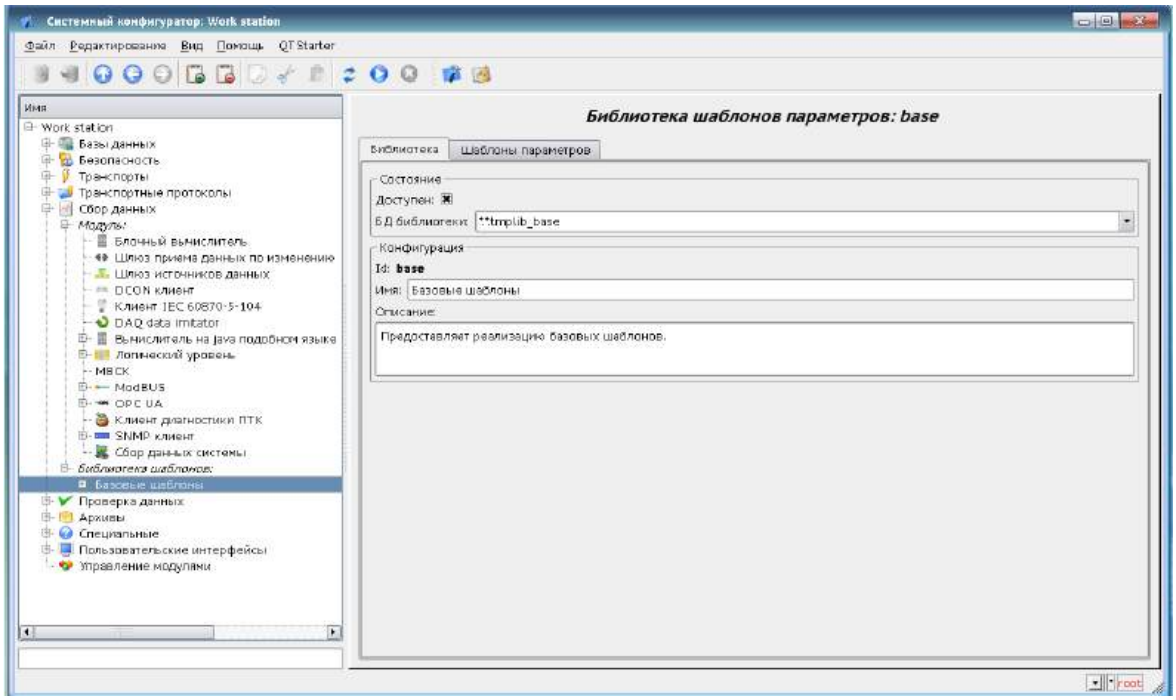


Рисунок 54

Вкладка "Шаблоны параметров" (рисунок 55) содержит список шаблонов в библиотеке. В контекстном меню списка пользователю предоставляется возможность добавления, удаления и перехода к нужному шаблону.

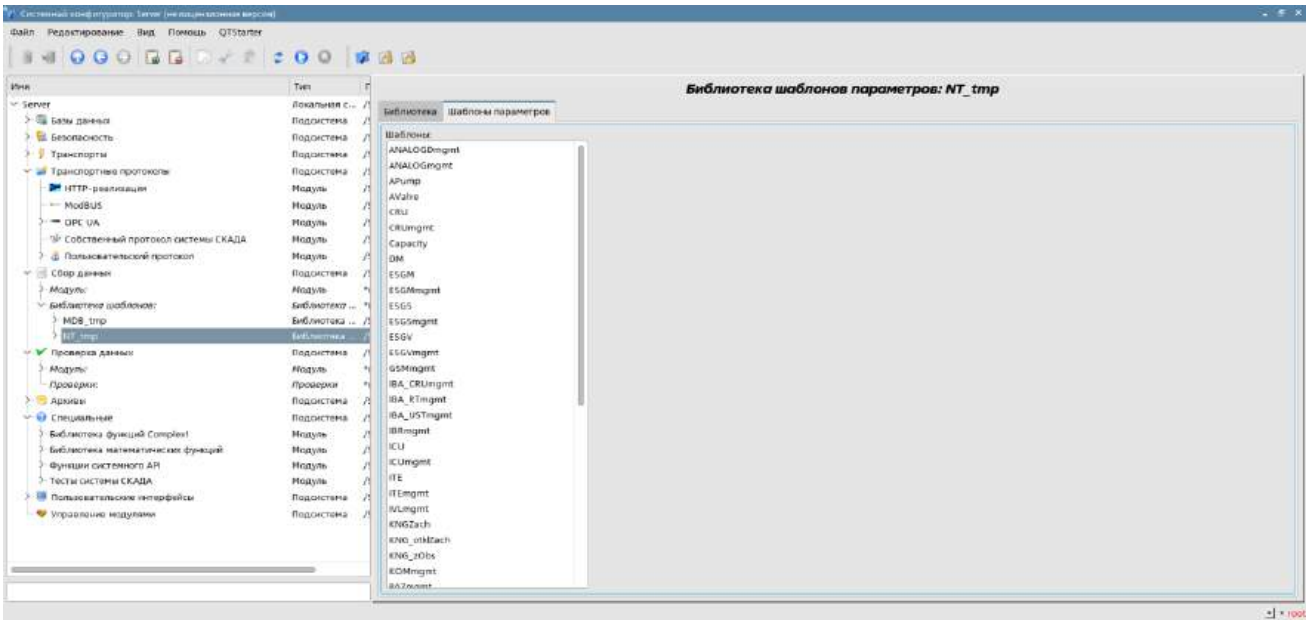


Рисунок 55

Каждый шаблон библиотеки шаблонов предоставляет конфигурационную страницу с вкладками "Шаблон" и "IO". Вид вкладки "Шаблон" показан на рисунке 56.

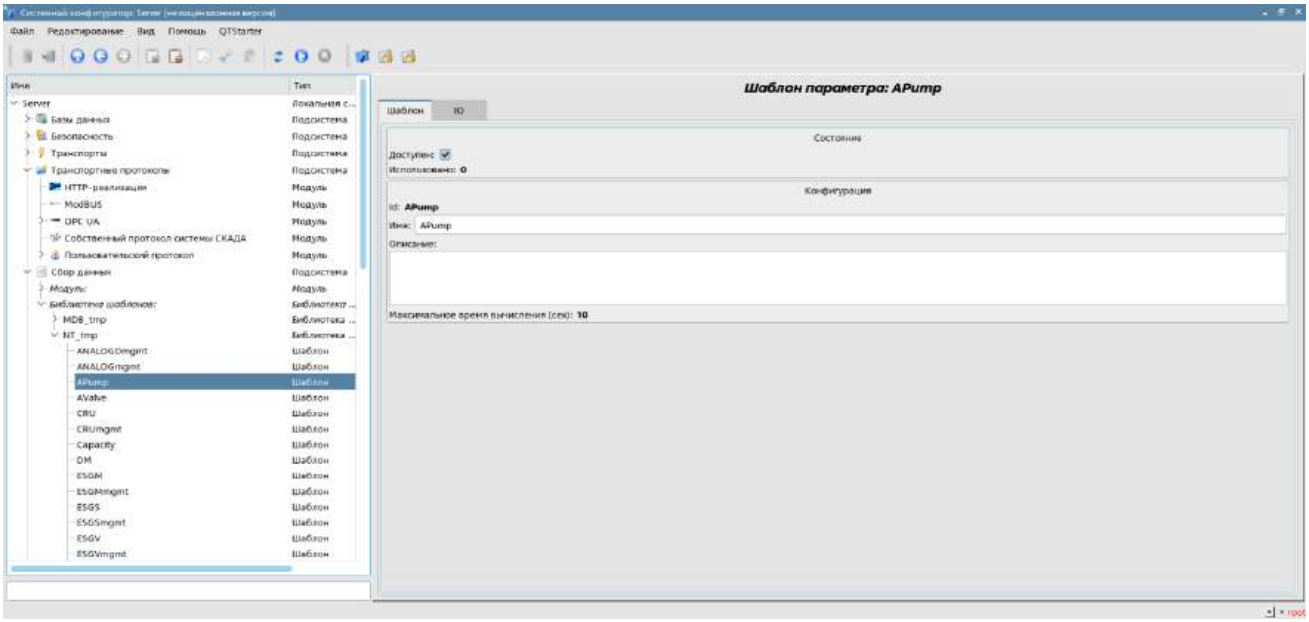


Рисунок 56

Вкладка "Шаблон" содержит основные настройки шаблона в составе:

- Раздел "Состояние" - содержит свойства, характеризующие состояние шаблона:

  - *Доступен* - состояние шаблона "Доступен";

  - *Использовано* - указывает, сколько раз шаблон использован;

- Раздел "Конфигурация" - непосредственно содержит поля конфигурации:

- *ID* - информация об идентификаторе шаблона;

- *Имя* - указывает имя шаблона;

- *Описание* - краткое описание шаблона и его назначения.

Вид вкладки "IO" показан на рисунке 57.

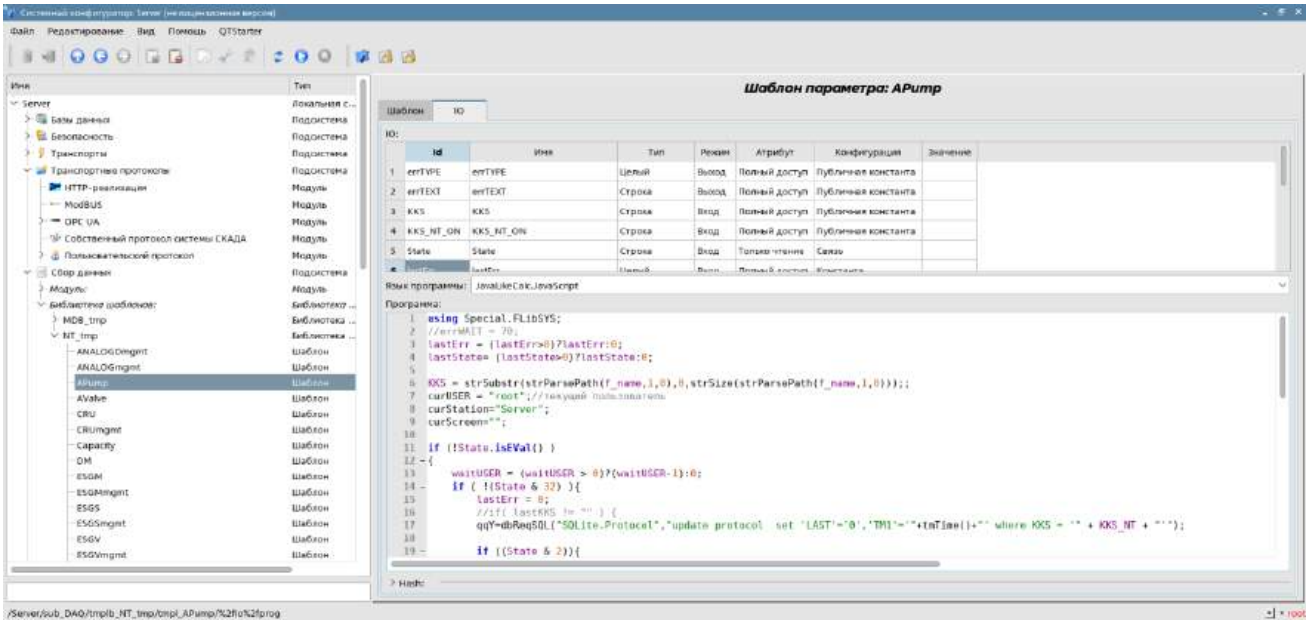


Рисунок 57

Вкладка "IO" содержит конфигурацию атрибутов (IO) шаблонов и программу шаблона на встроенном языке DAQ.JavaLikeCalc.JavaScript.

В таблице 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 чтение", "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}.

Каждый модуль подсистемы "Сбор данных" предоставляет конфигурационную страницу с вкладками "Контроллеры" и "Помощь". Вид вкладки "Контроллеры" показан на рисунке 58.

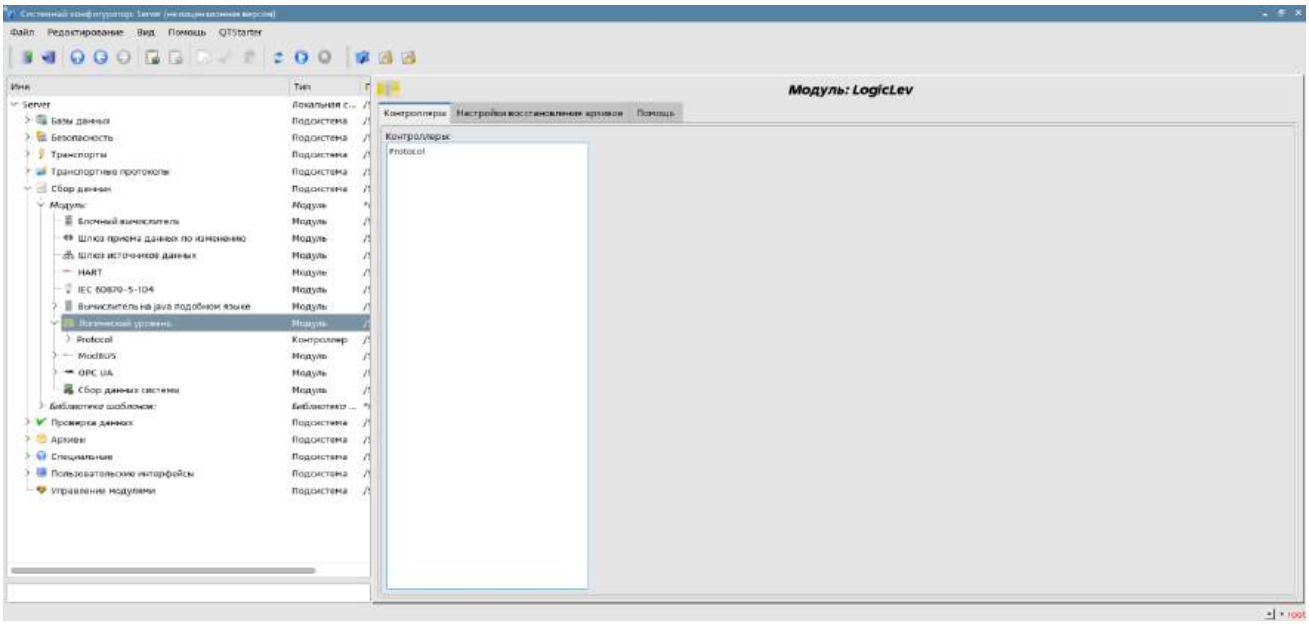


Рисунок 58

Вкладка "Контроллеры" 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", состав которой идентичен для всех модулей.

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

Вид вкладки "Контроллер" показан на рисунке 59.

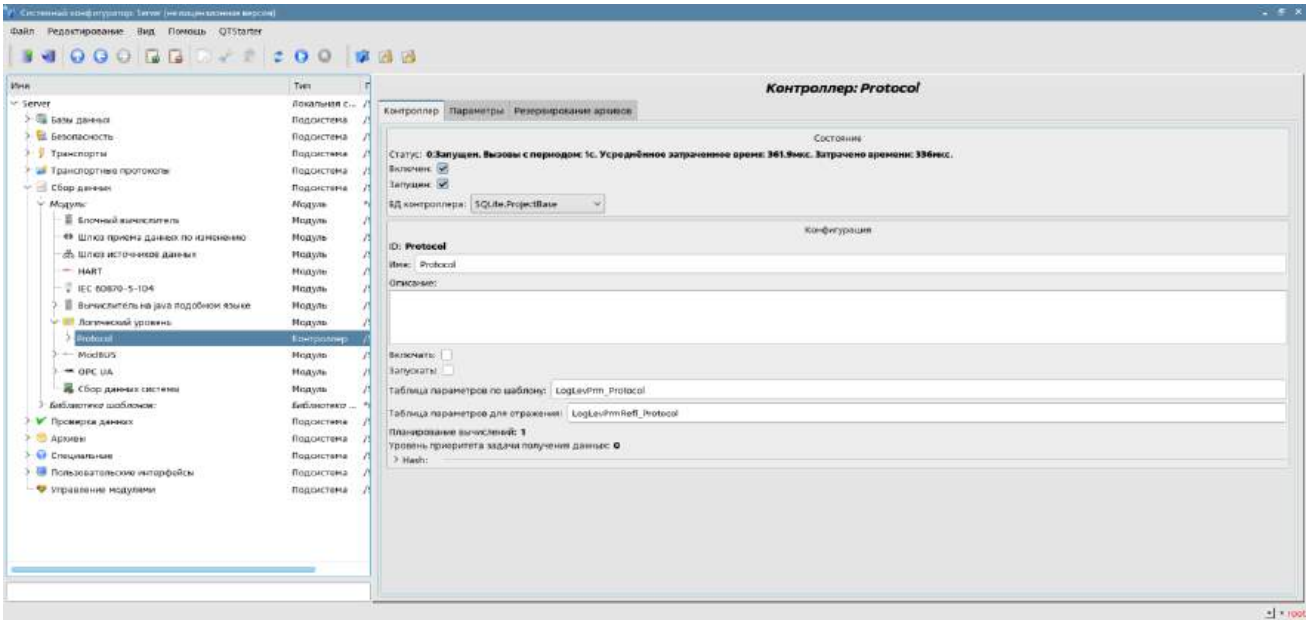


Рисунок 59

Вкладка "Контроллер" 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 модуля «Логический уровень» DAQ.LogicLev:

- Раздел "Состояние" - содержит свойства, характеризующие состояние контроллера:

- *Статус* - указывает статус контроллера и время его вычисления;

- *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, выбирается из списка щелчком мыши.

- 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 следует перевести контроллер при загрузке;

- *Запускать* - указывает на состояние "Запущен", в которое следует перевести контроллер при загрузке;

- *Таблица параметров по шаблону* - имя таблицы, в которой сохранять параметры (имеются в виду объекты параметров сбора данных) контроллера;

- *Планирование вычислений (с)* - периодичность выполнения задачи;

- *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.

Вкладка "Параметры" 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 (рисунок 60).

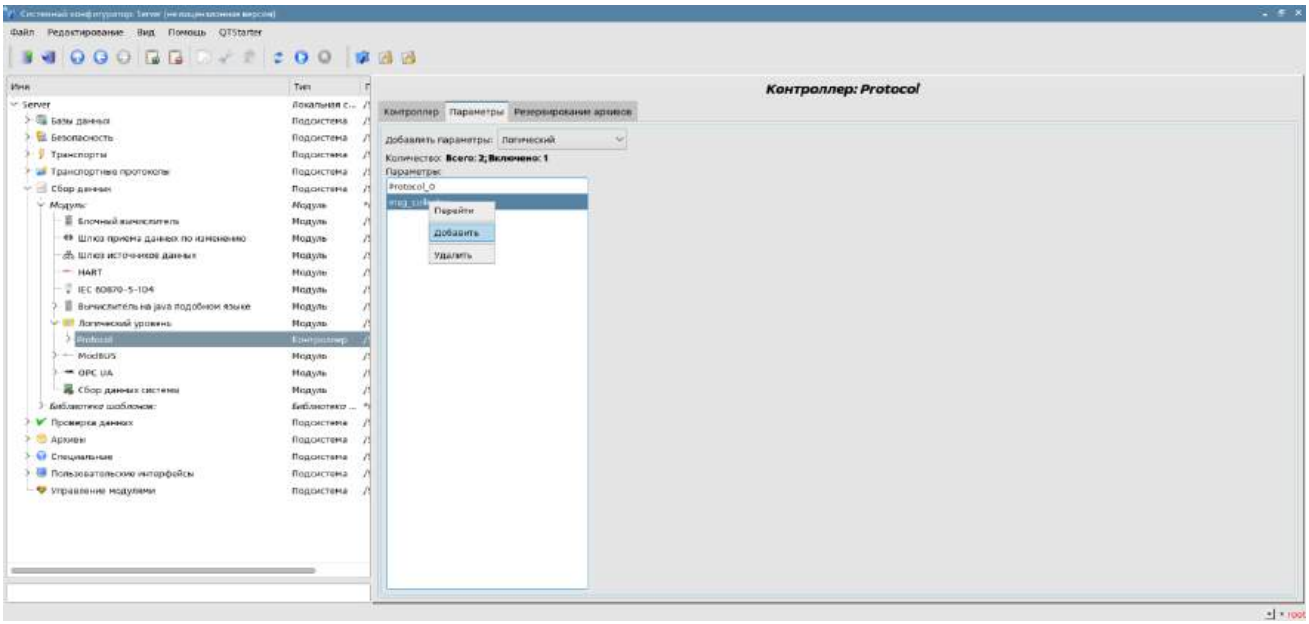


Рисунок 60

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.

Вкладка "Параметр" (рисунок 61) 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;

- *Шаблон параметра* – выбор из выпадающего списка шаблонов, настроенных в «Библиотеке шаблонов».

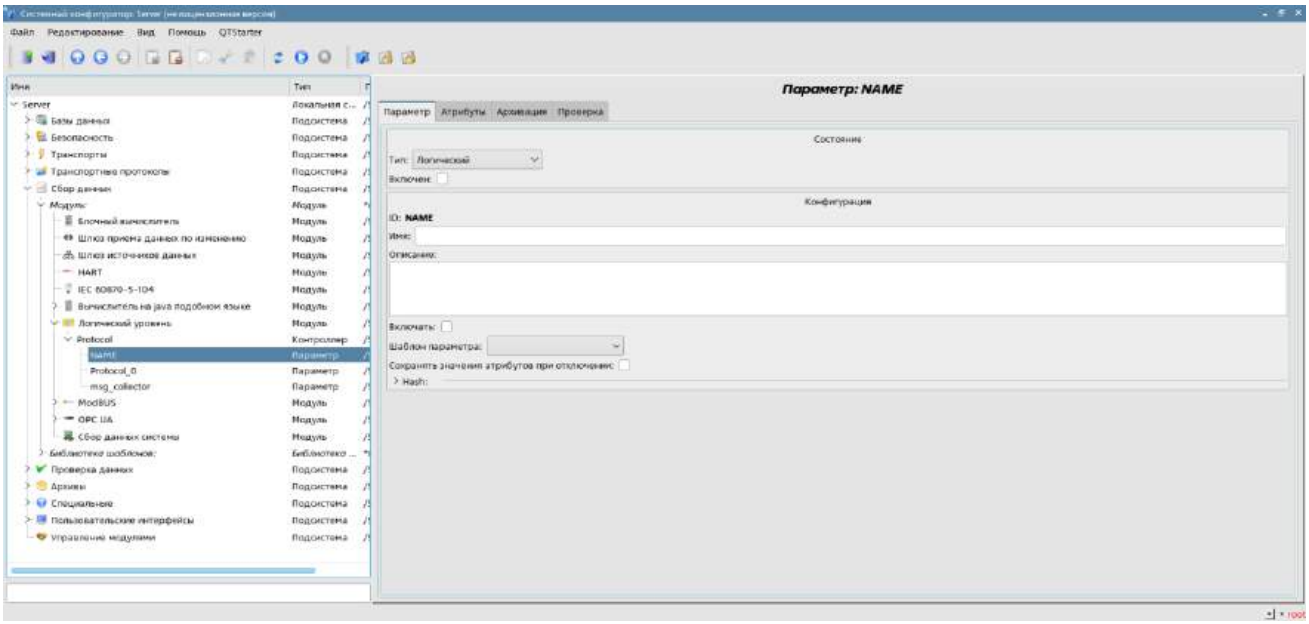


Рисунок 61

Вкладка "Атрибуты" (рисунок 62) содержит атрибуты параметра и их значения в соответствии с конфигурацией используемого шаблона и выполнением его программы.

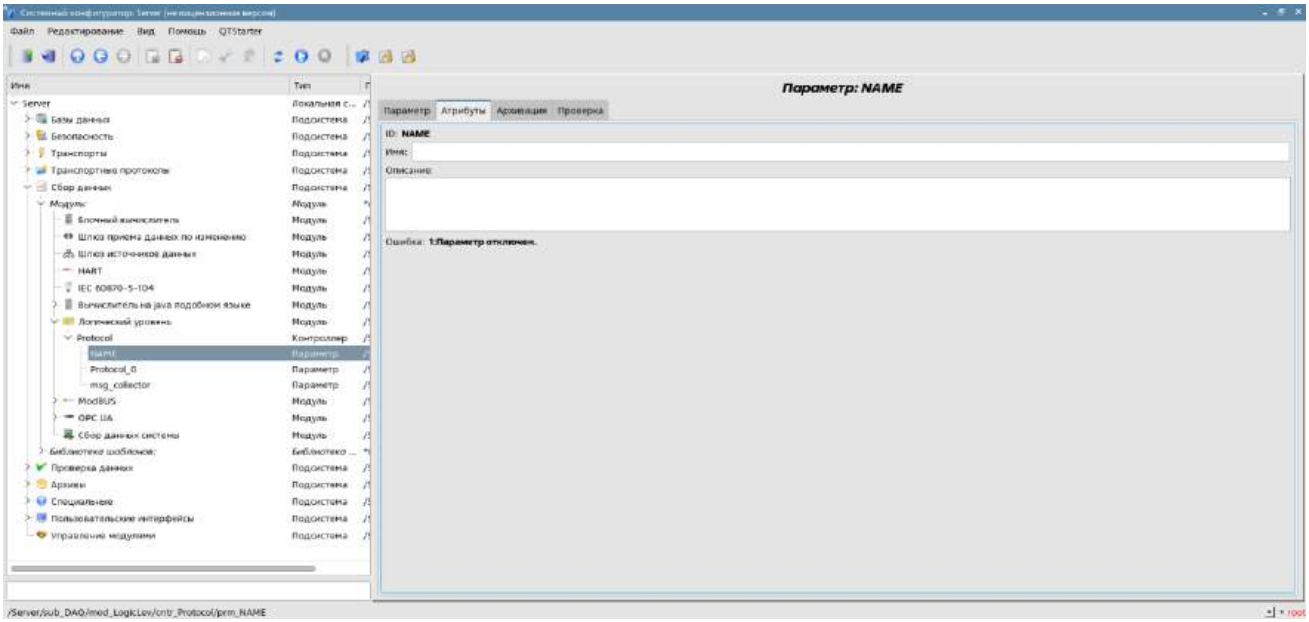


Рисунок 62

Вкладка "Архивация" (рисунок 63) содержит таблицу с атрибутами параметра в строках, и архиваторами в колонках. Пользователь имеет возможность установить архивацию нужного атрибута требуемым архиватором, просто изменив ячейку на

пересечении. Например, чтобы добавить архивацию атрибута «NAME» архиватором «DBArch.ArchVal» необходимо произвести двойной щелчок мышью в ячейке на пересечении соответствующей колонки и строки (рисунок 64). При этом архиватор «DBArch.ArchVal» предварительно должен быть создан в модуле подситемы «Архивы» (порядок действий в 5.9.4). Затем щелкнуть мышью на появившемся списке и выбрать «Да» (рисунок 65). В результате для атрибута «NAME» будет установлена необходимая архивация (рисунок 66).

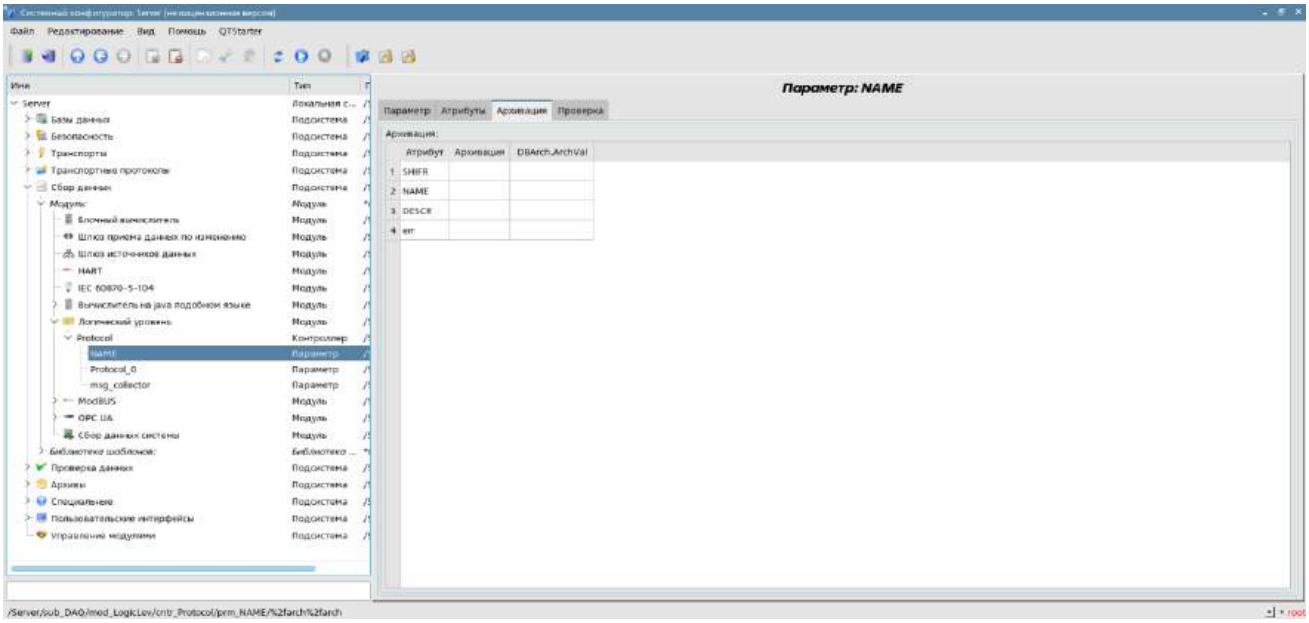


Рисунок 63

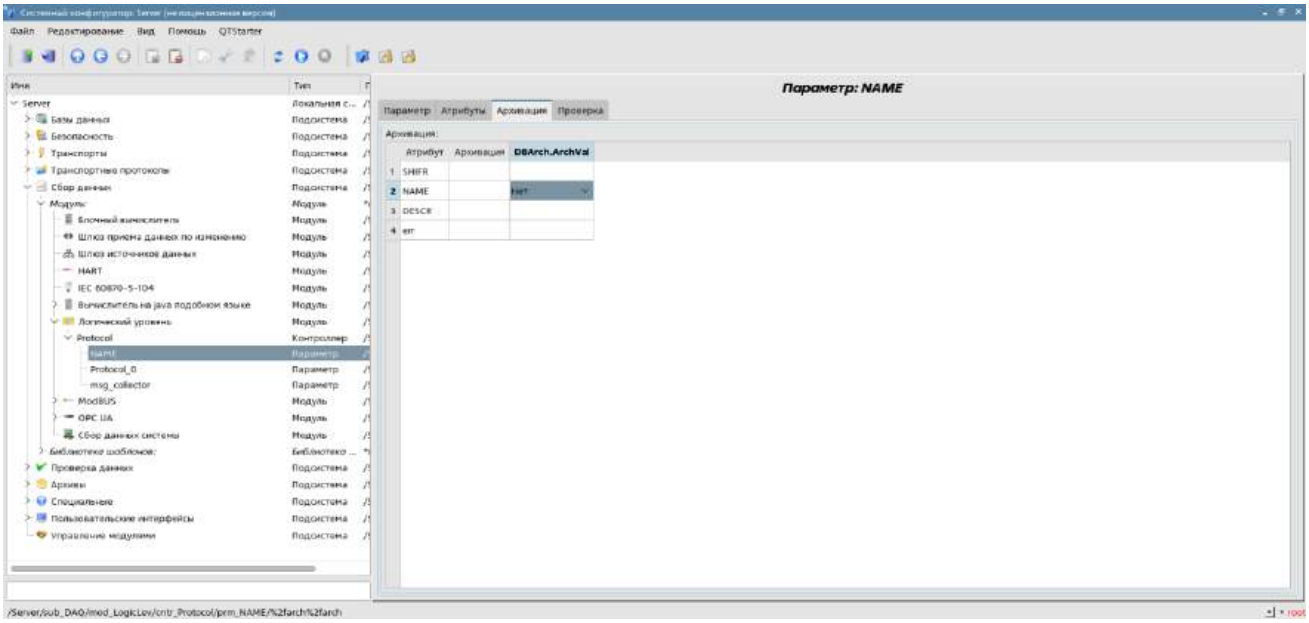


Рисунок 64

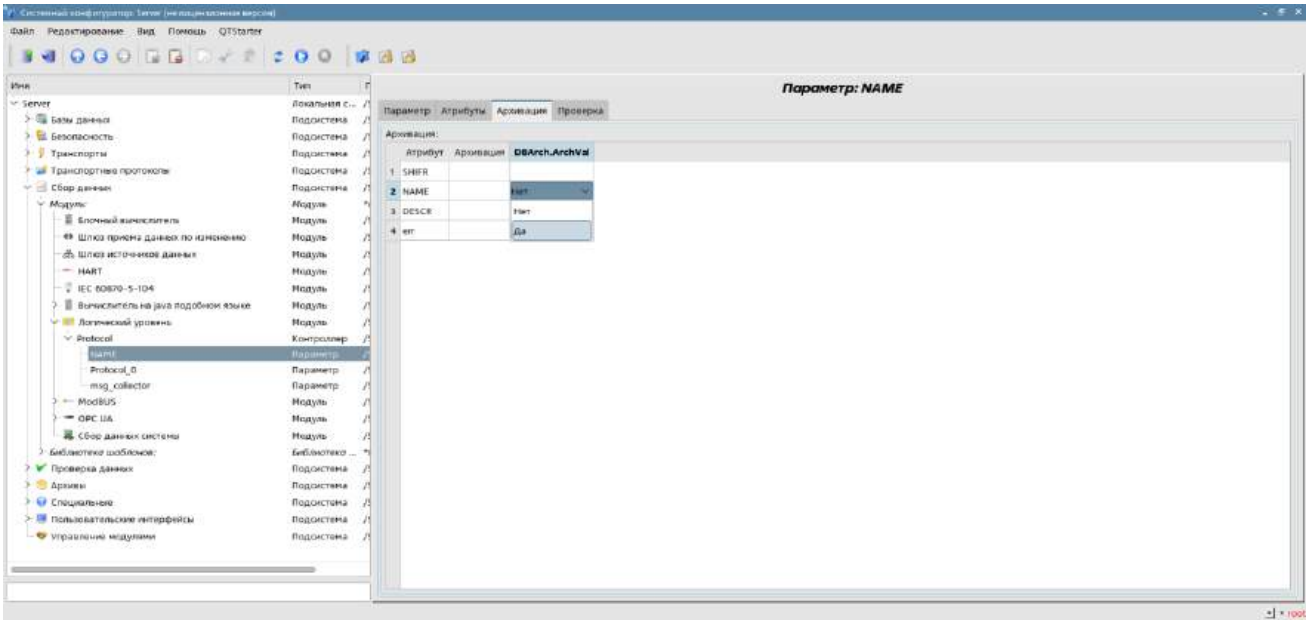


Рисунок 65

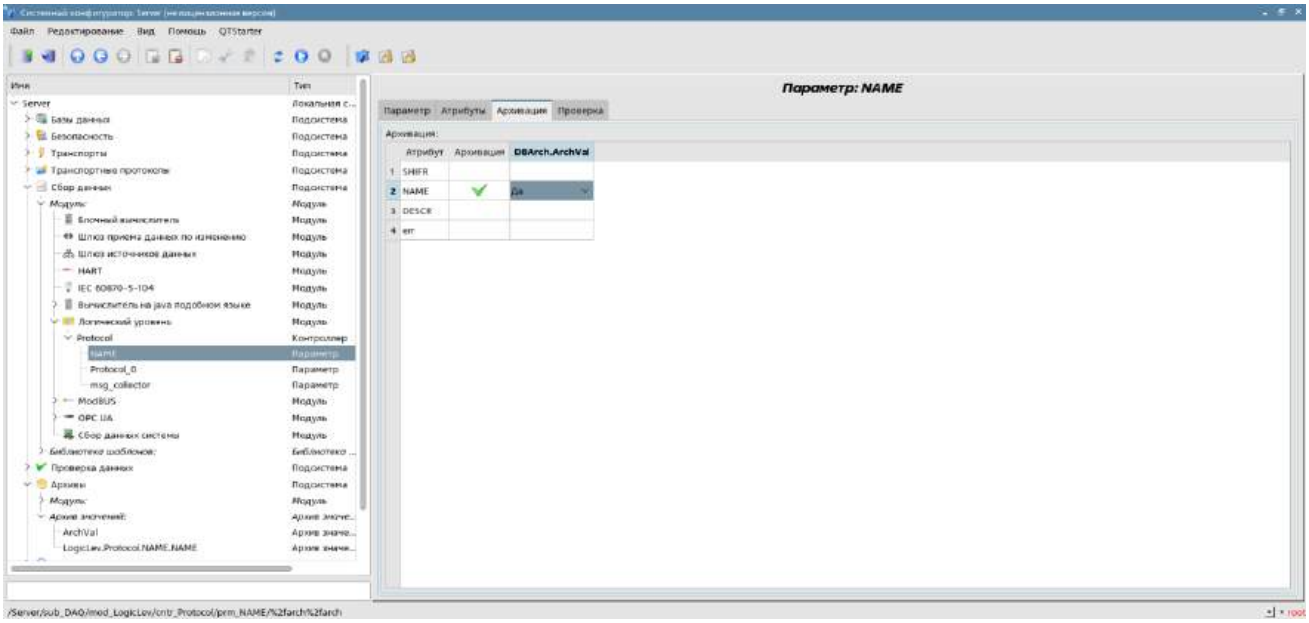


Рисунок 66

Вкладка «Проверка» (рисунок 67) содержит настройки для следующих проверок значений атрибутов:

- выход за границы диапазона;
- коды ошибок от устройства;
- скорость изменения атрибута.

Раздел «Атрибуты» содержит перечень атрибутов, для которых возможно настроить проверки и задать соответствующие сообщения о тревогах, которые будут поступать в журнал тревог и событий.

Раздел «Параметры проверки» предоставляет возможность выбора типа проверки:

VldBorder – выход за границы диапазона, т.е. задание верхних и нижних границ диапазона и предупредительных и аварийных уставок (рисунок 67);

VldFromDevice – коды ошибок от устройства (рисунок 68);

VldSpeed – задание верхней и нижней границ скорости изменения атрибута (рисунок 70).

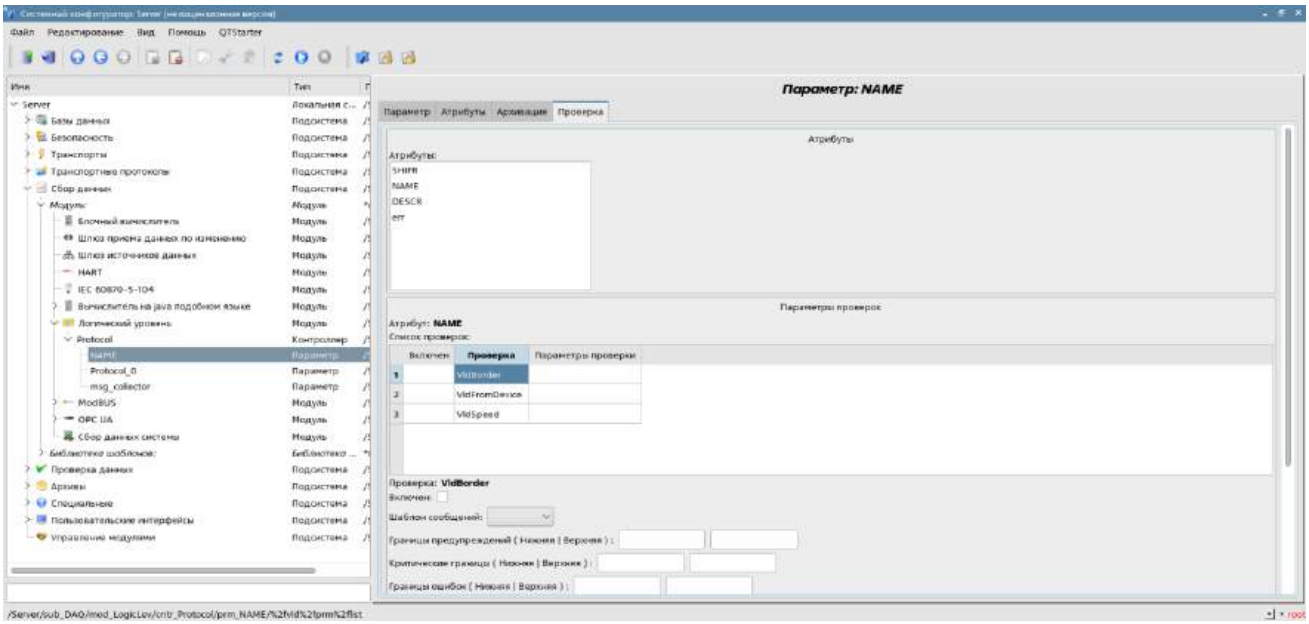


Рисунок 67

При выборе атрибута и типа проверки становятся доступными соответствующие поля настройки для выбранной проверки. Всплывающая подсказка предоставляет формат заполнения выбранного поля (рисунки 68 и 69).

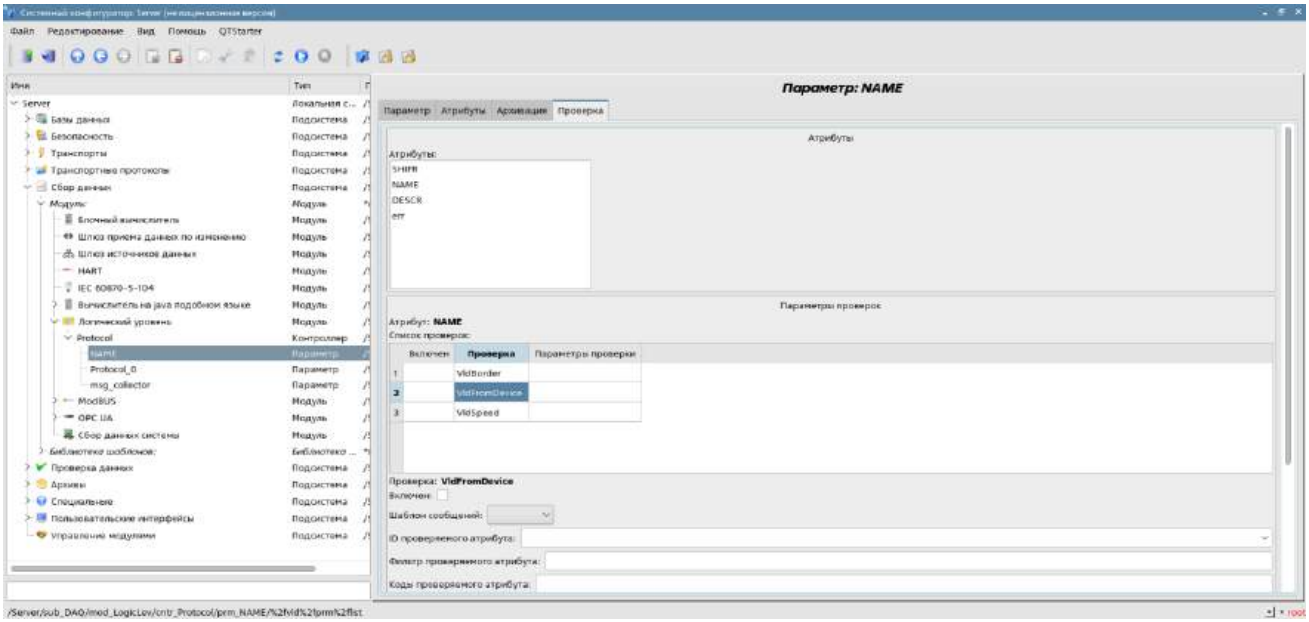


Рисунок 68

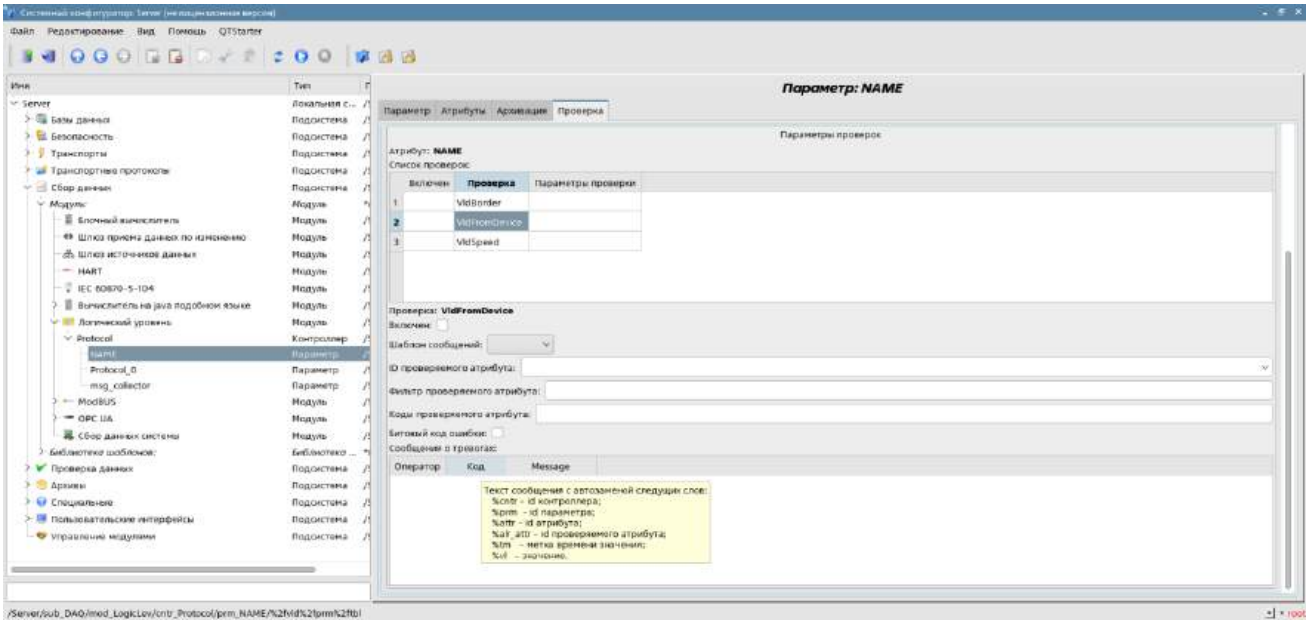


Рисунок 69

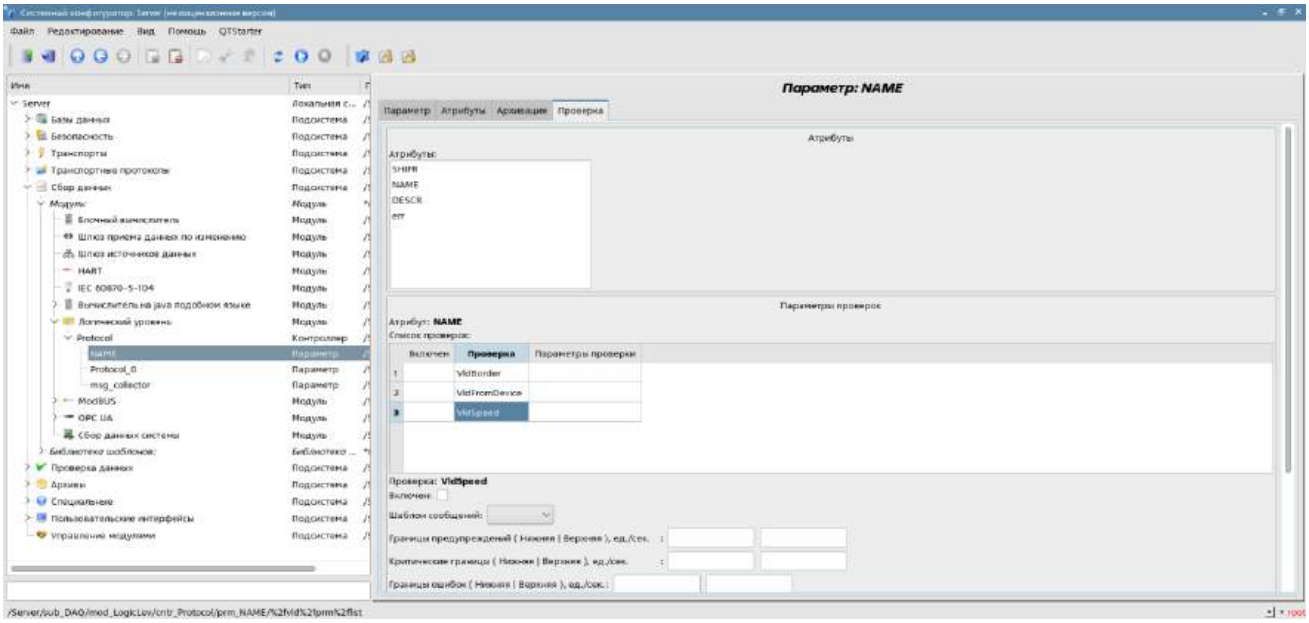


Рисунок 70

Для настройки проверки атрибута на выход за границы диапазона (VldBorder) необходимо задать верхние и нижние значения границ диапазона для предупредительных, критических и ошибочных уставок. Выбор опции «Проверка на EVAL» позволяет системе выдавать сообщение об ошибке при значении атрибута равном <EVAL>. Для настройки сообщений об ошибках необходимо заполнить поле «Сообщение» в таблице «Сообщения о тревогах». Для запуска проверки необходимо установить флаг «Включен» (рисунок 71).

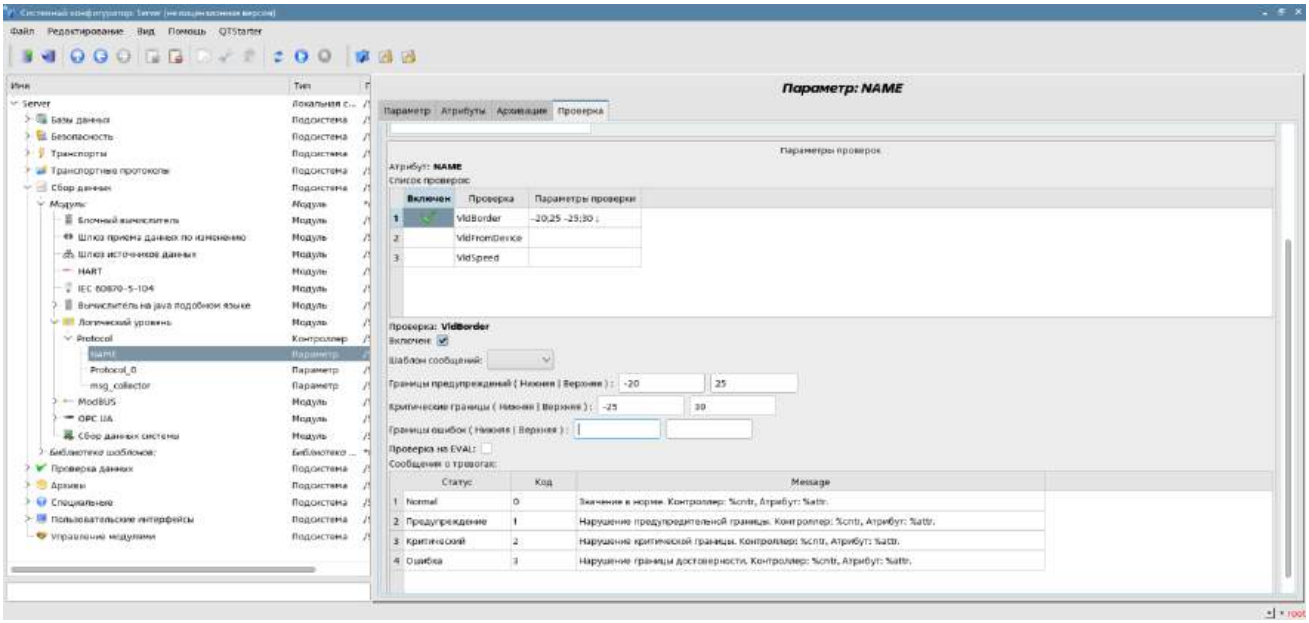


Рисунок 71

Для настройки проверки кодов ошибок от устройства (VldFromDevice) необходимо указать ID проверяемого атрибута. Для преобразования приходящих от устройств значений при необходимости настроить фильтр проверяемого атрибута:

Формат поля: <оператор><фильтр>

Операторы: AND: &, OR: |, NOT: ~, XOR: ^

Далее, указать коды проверяемого атрибута:

Формат поля: [опер1]([опер2][код1;][опер3][код2;][опер4][код-3-код6;])

Операторы: AND: &, OR: |, NOT: ~, XOR: ^, <EVAL>: E

Для проверки атрибута на EVAL необходимо указывать код с оператором *E*. Если проверяемый атрибут содержит несколько битов ошибки, необходимо выбрать опцию «Битовый код ошибки». Для настройки сообщений об ошибках необходимо в таблице «Сообщения о тревогах» добавить запись (правой кнопкой), выбрать соответствующий оператор, указать код ошибки и сообщение. Для запуска проверки установить флаг «Включен» (рисунок 72).

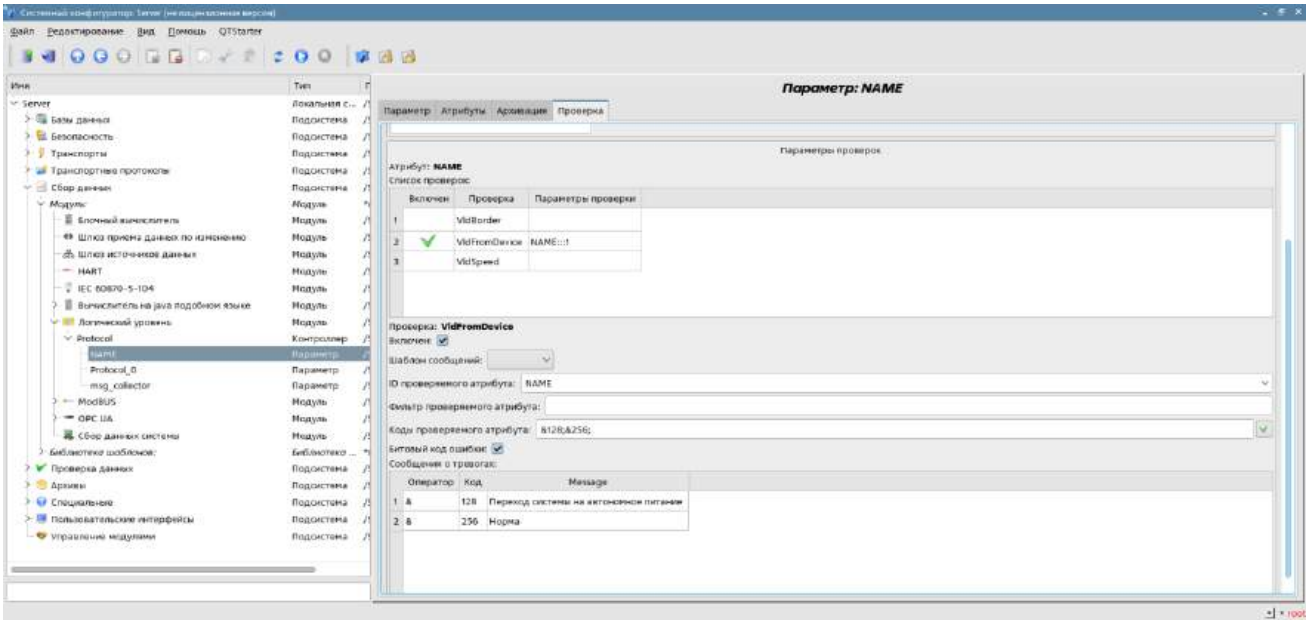


Рисунок 72

Для настройки проверки скорости изменения атрибута (VldSpeed) необходимо задать верхние и нижние границы диапазона для предупредительных, критических и ошибочных уставок. Для настройки сообщений об ошибках необходимо заполнить поле «Сообщение» в таблице «Сообщения о тревогах». Для запуска проверки выбрать опцию «Включен» (рисунок 73).

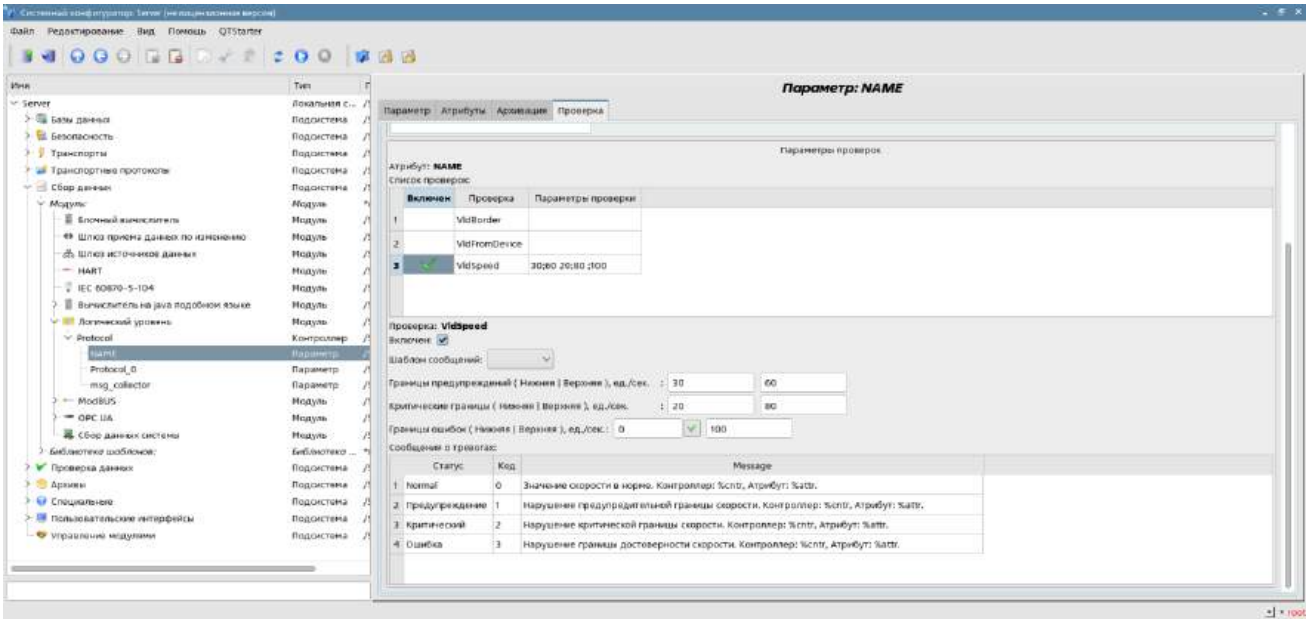


Рисунок 73

Вкладка "Конфигурация шаблона" (рисунок 74) содержит конфигурационные поля в соответствии с шаблоном. Связь можно установить, указав путь к параметру (выбирается мышью из списка), (рисунок 75). Параметры и атрибуты шаблона настраиваются с помощью Библиотеки шаблонов на вкладке «IO» (см. рисунок 57).

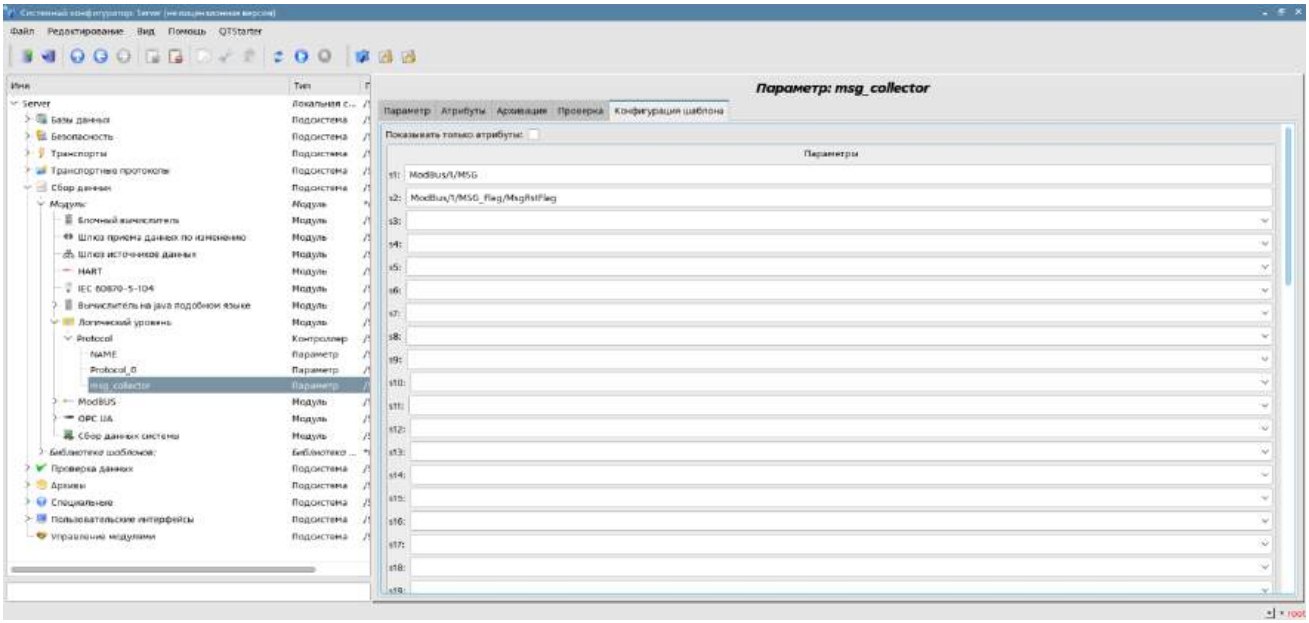


Рисунок 74

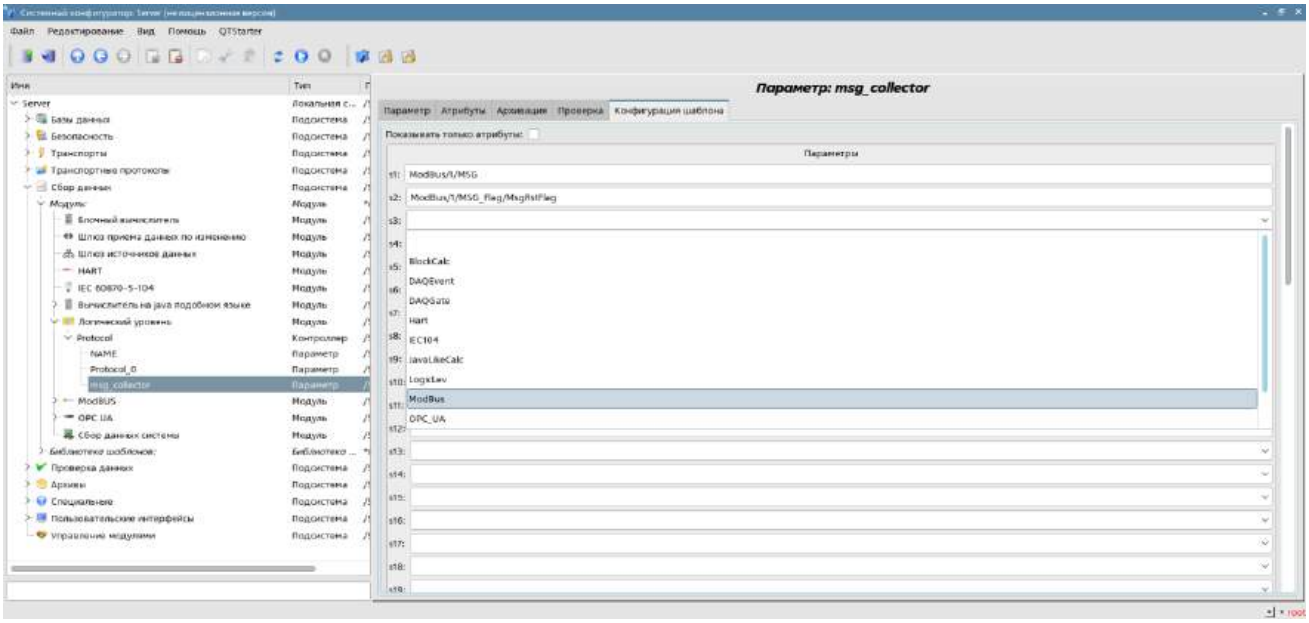


Рисунок 75

5.8 Подсистема «Проверка данных»

Подсистема «Проверка данных» предназначена для создания шаблонов для проверки значений атрибутов модулей подсистемы «Сбор данных» на:

- выход за границы диапазона;
- коды ошибок от устройства;
- скорость изменения атрибута.

Шаблоны проверок можно применять для каждого источника данных подсистемы "Сбор данных" выбрав необходимый в строке "Шаблон сообщений".

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Проверка данных", содержащая вкладки "Проверки", "Модули" и "Помощь".

Вкладка «Проверки» содержит список ранее созданных проверок, поля для настройки периода проверки данных и установки приоритета задачи проверки данных. В контекстном меню списка проверок предоставляется возможность перехода к нужной проверке. Вид вкладки "Проверки" показан на рисунке 76.

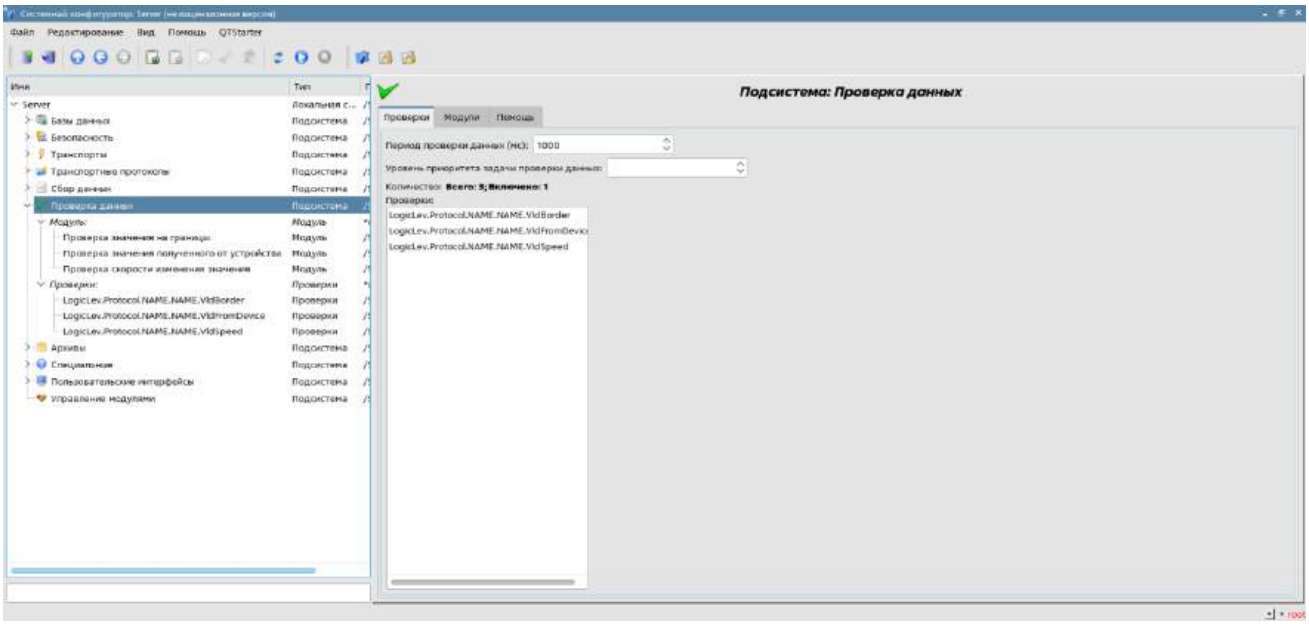


Рисунок 76

Вкладка "Модули" содержит список модулей проверок: проверка значения на границы, проверка значения, полученного от устройства, проверка скорости изменения значения (рисунок 77). Для перехода к контекстному меню списка шаблонов необходимо дважды щелкнуть левой кнопкой мыши по модулю.

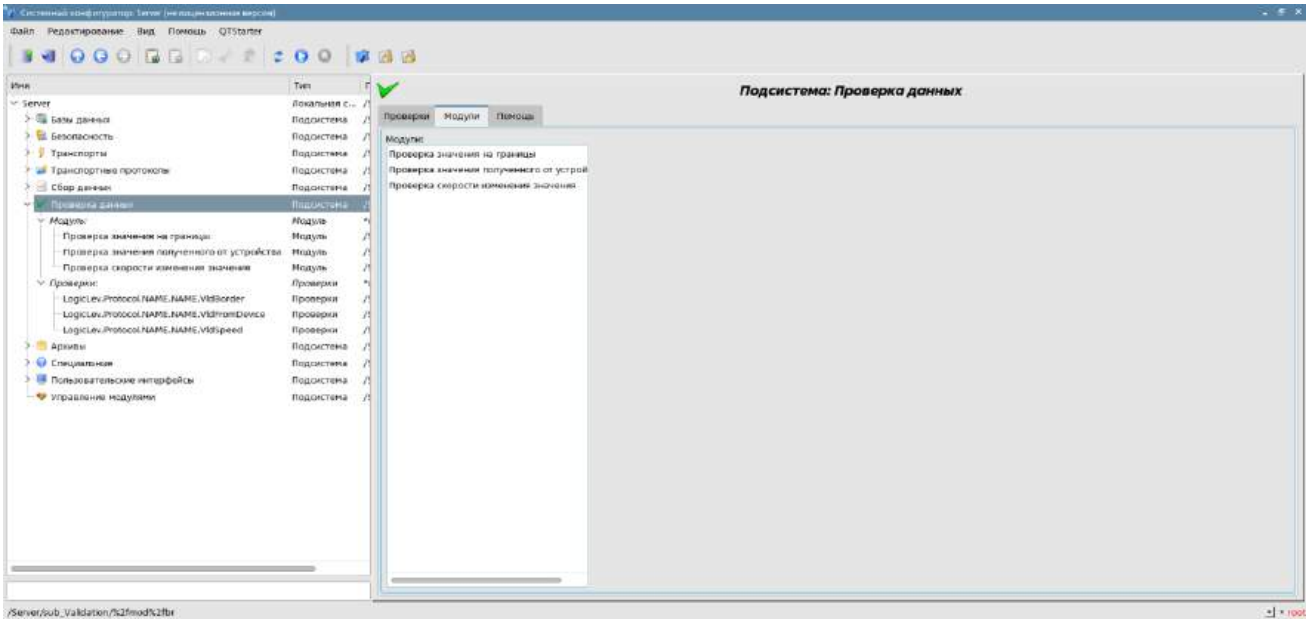


Рисунок 77

В контекстном меню списка шаблонов пользователю предоставляется возможность добавления, удаления и перехода к нужному шаблону сообщений

проверок (рисунок 78). Для добавления шаблона нужно щелкнуть правой клавишей мыши в любом месте списка и в появившемся списке выбрать пункт «Добавить». В окне «Установка имени элемента» заполнить поля «ID» и «Имя» (рисунок 79). В результате новый шаблон появится в списке (рисунок 80).

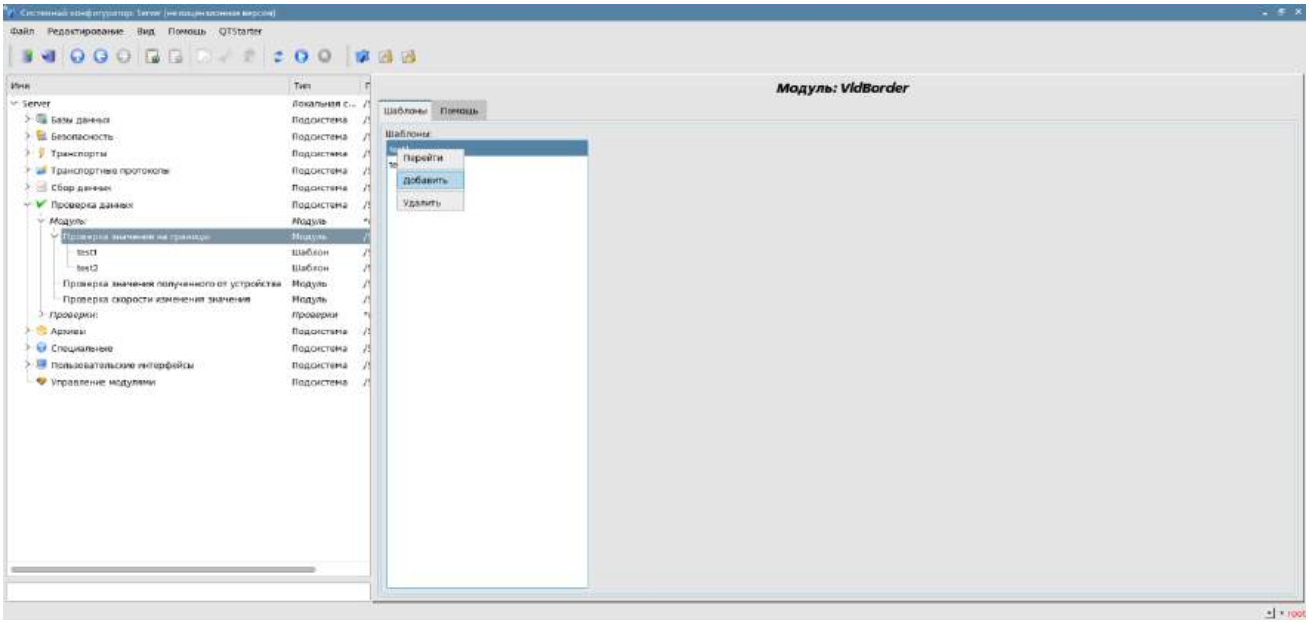


Рисунок 78

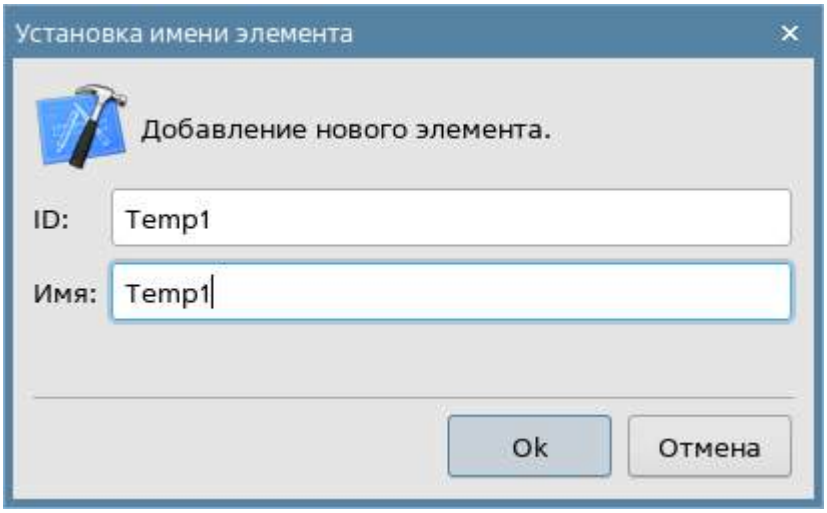


Рисунок 79

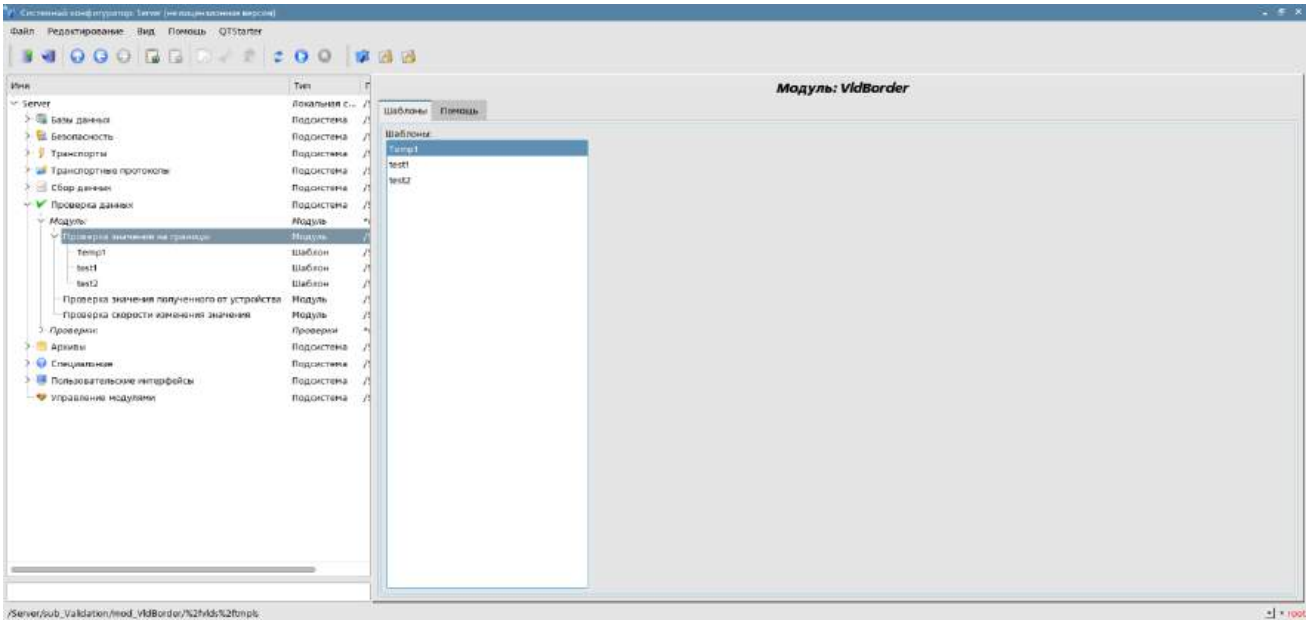


Рисунок 80

При переходе на созданный шаблон «Temp1» (двойной клик левой кнопкой мыши) откроется вкладка шаблон для настройки шаблона сообщения проверки (рисунок 81).

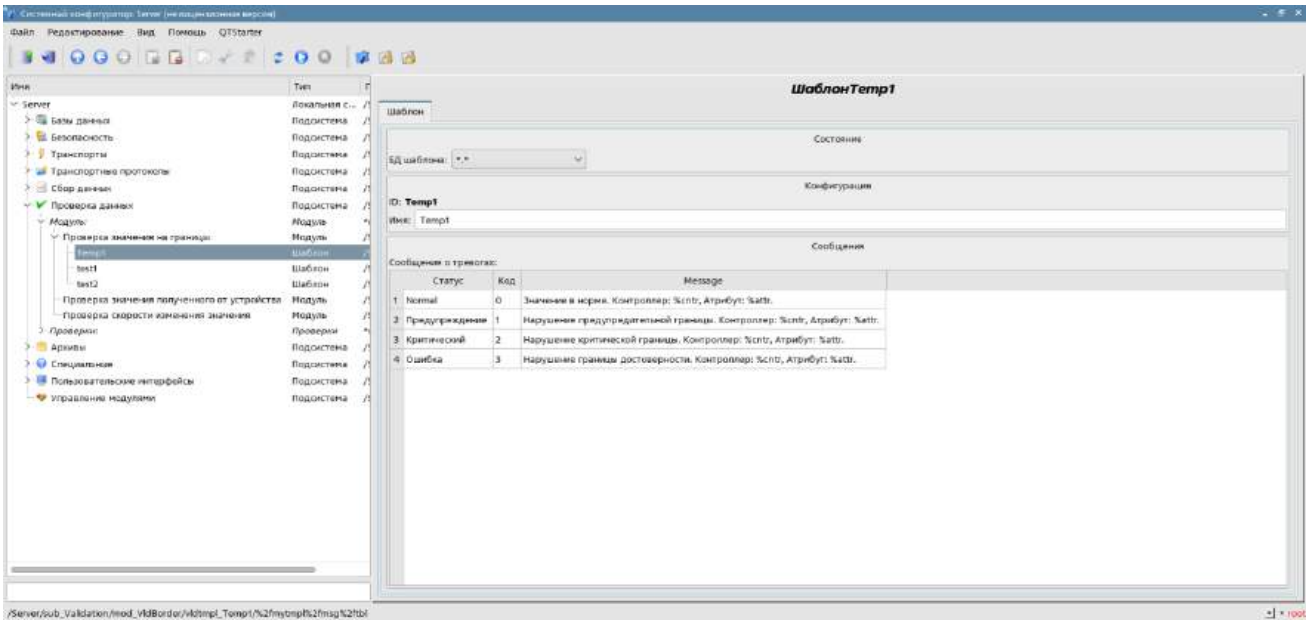


Рисунок 81

Для использования шаблона сообщений проверки на вкладке конфигурации проверки модуля подсистемы «Сбор данных» в выпадающем списке «Шаблон сообщений» необходимо выбрать соответствующий идентификатор шаблона (рисунок 82).

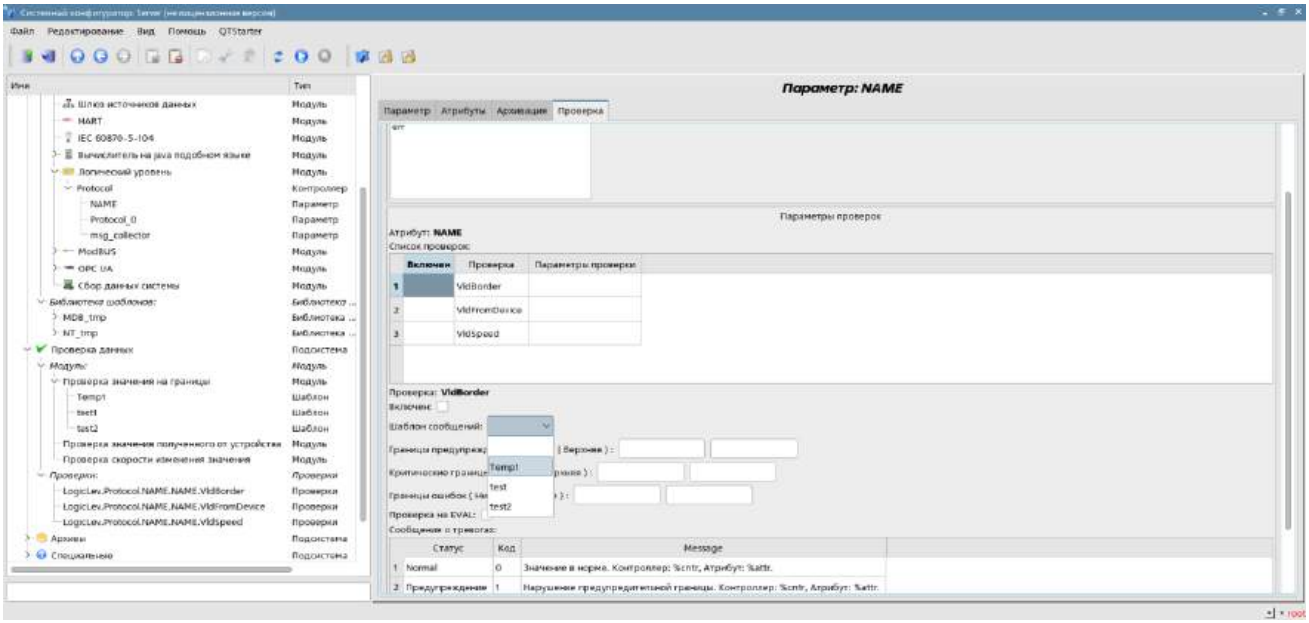


Рисунок 82

5.9 Подсистема «Архивы»

5.9.1 Общие сведения

Для решения задач архивирования потоков данных в СКАДА 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). 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 СКАДА 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 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 или регулярные выражения, которые определяют какие сообщения пропускать. Каждый архиватор содержит собственные настройки фильтра.

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

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

5.9.2 Архиватор на БД

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

Архивы условно можно разделить на два типа: архивы сообщений и архивы значений.

Особенностью архивов сообщений является то, что архивируются события. Характерным признаком события является его время возникновения. Архивы сообщений обычно используются для архивирования сообщений в системе, т.е. ведения 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.

Подсистема в целом является модульной, что позволяет создавать архивы, основанные на различной природе и способах хранения данных. Данный модуль предоставляет механизм архивирования в БД как для потока сообщений, так и для потока значений.

5.9.2.1 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.

5.9.2.2 Архиватор значений

Архивы значений формируются архиваторами значений индивидуально для каждого источника. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделить архивы по различным характеристикам, 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 СКАДА, а также другие внешние источники данных (пассивный режим). Источниками данных также могут быть: сетевые архиваторы удалённых СКАДА систем, среда программирования системы СКАДА и др. Не менее важными параметрами архива являются параметры его буфера. Буфер создается свой для каждого источника данных. От параметров буфера зависит возможность работы архиваторов. Так, периодичность опроса буфера архиватором не должна быть менее

периодичности поступления данных в буфер от источника. А размер должен выбираться в зависимости от периодичности опроса архиватором, для исключения переполнения буфера. В противном случае возможны потери данных!

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.

Для хранения начала, конца архива и иной информации об архиве 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.

5.9.3 Архиватор на ФС

Данный модуль предоставляет механизм архивирования на файловую систему, как для потока сообщений, так и для потока значений.

5.9.3.1 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.

В число этих параметров входят:

выбор 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 размещении в директории архиватора файлов архивов другой станции.

5.9.3.2 Формат файлов архива сообщений

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

Таблица 4

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).

5.9.3.3 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.

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.

5.9.3.4 Формат файлов архива значений

Для реализации архивирования на файловую систему предъявляются 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 рисунке 83.

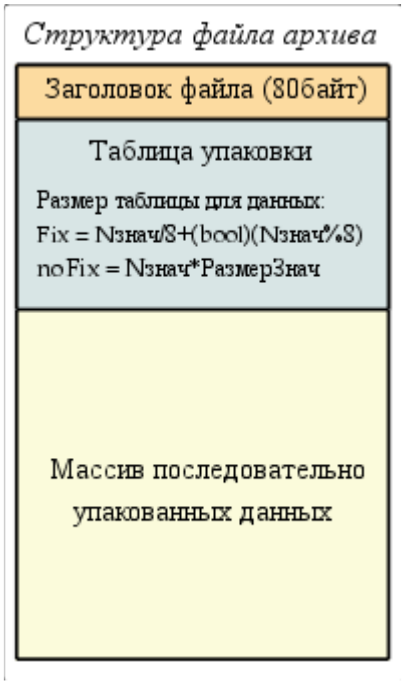


Рисунок 83

При создании нового файла архива формируется: заголовок (структура заголовка в таблице 5), 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 5

Поле	Описание	Размер, байт (бит)
f_tp	Системное имя архива («SCADA 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

Поле	Описание	Размер, байт (бит)
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 рисунке 84. 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 трудоёмкой 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).

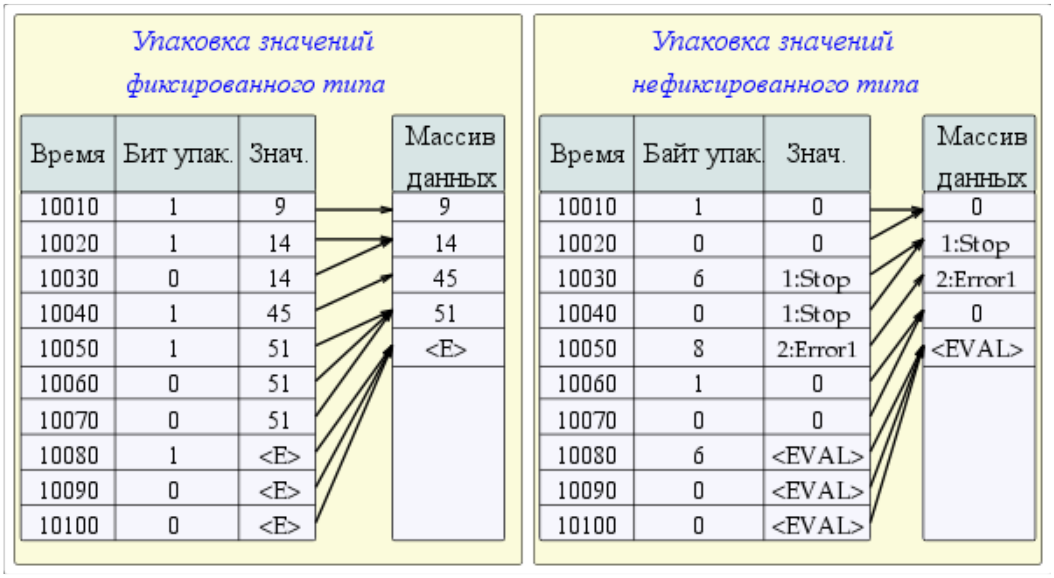


Рисунок 84

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.

5.9.4 Конфигурирование подсистемы «Архивы»

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.

Вид вкладки "Архив сообщений" показан на рисунке 85.

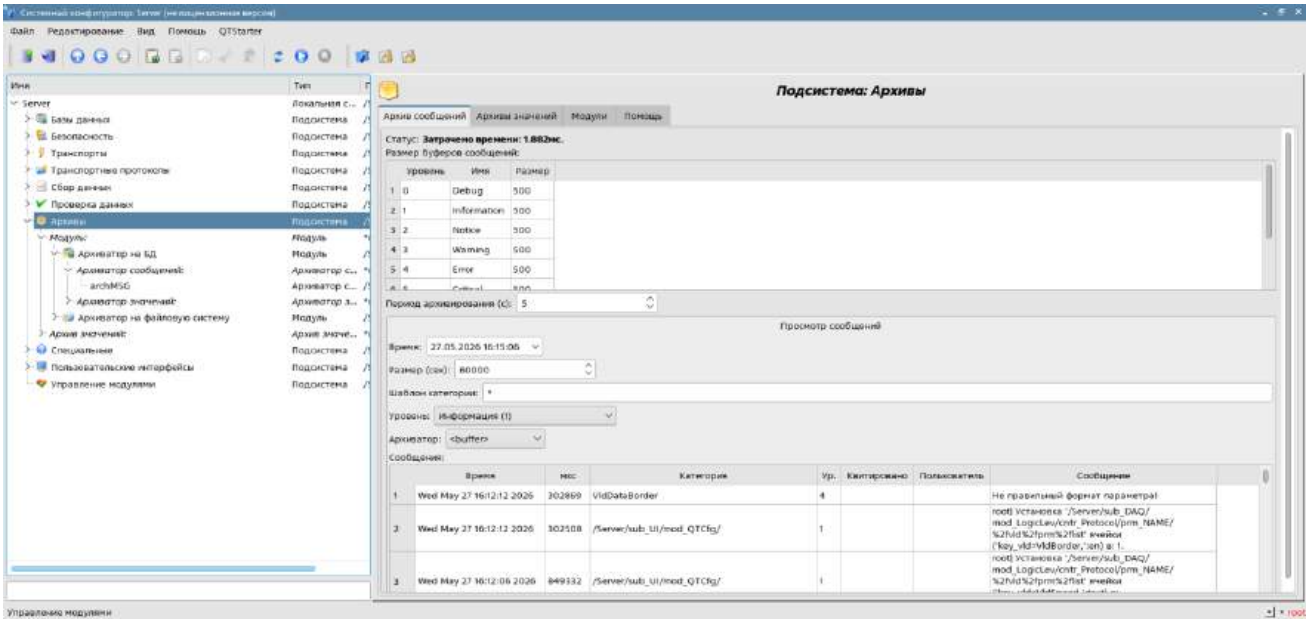


Рисунок 85

Вкладка "Архив сообщений" содержит конфигурацию архива сообщений и форму запроса сообщений из архива.

Конфигурация архива сообщений представлена полями:

- *Размер буфера сообщений* - указывает на размерность области оперативной памяти, зарезервированной под 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.

Вид вкладки "Архивы значений" показан на рисунке 86.

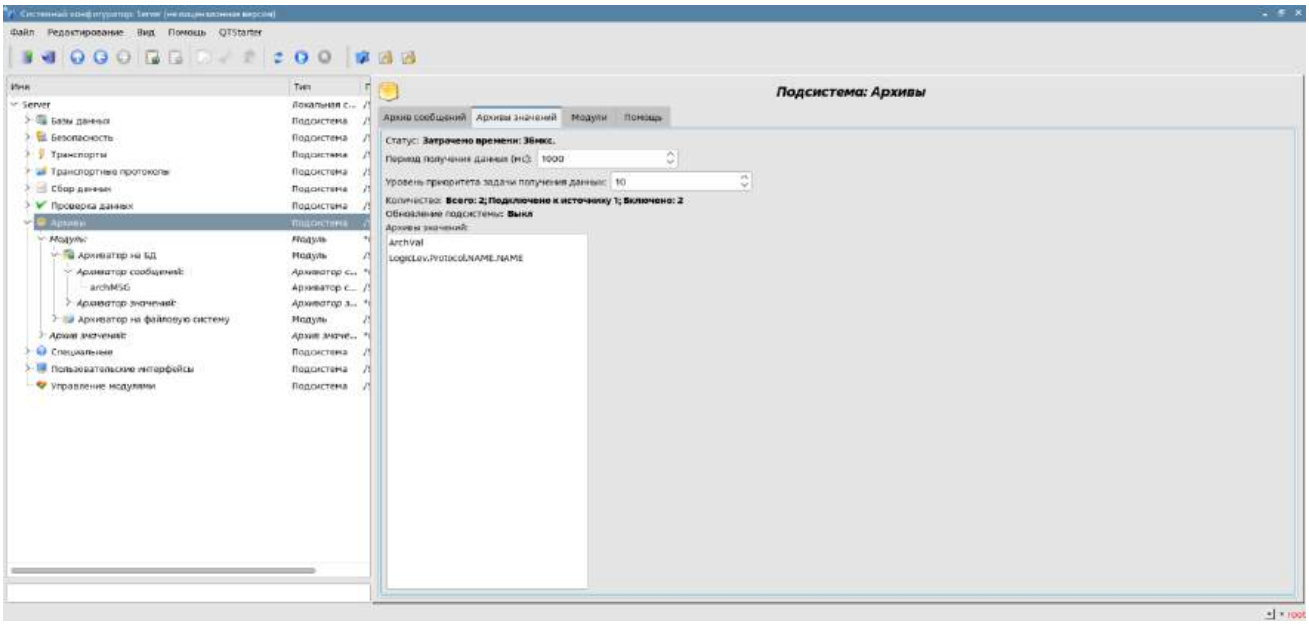


Рисунок 86

Вкладка "Архивы значений" 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.

- *Архивы значений* - список имен архивов значений, которые используются в системе.

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

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

Вид вкладки "Архив " показан на рисунке 87.

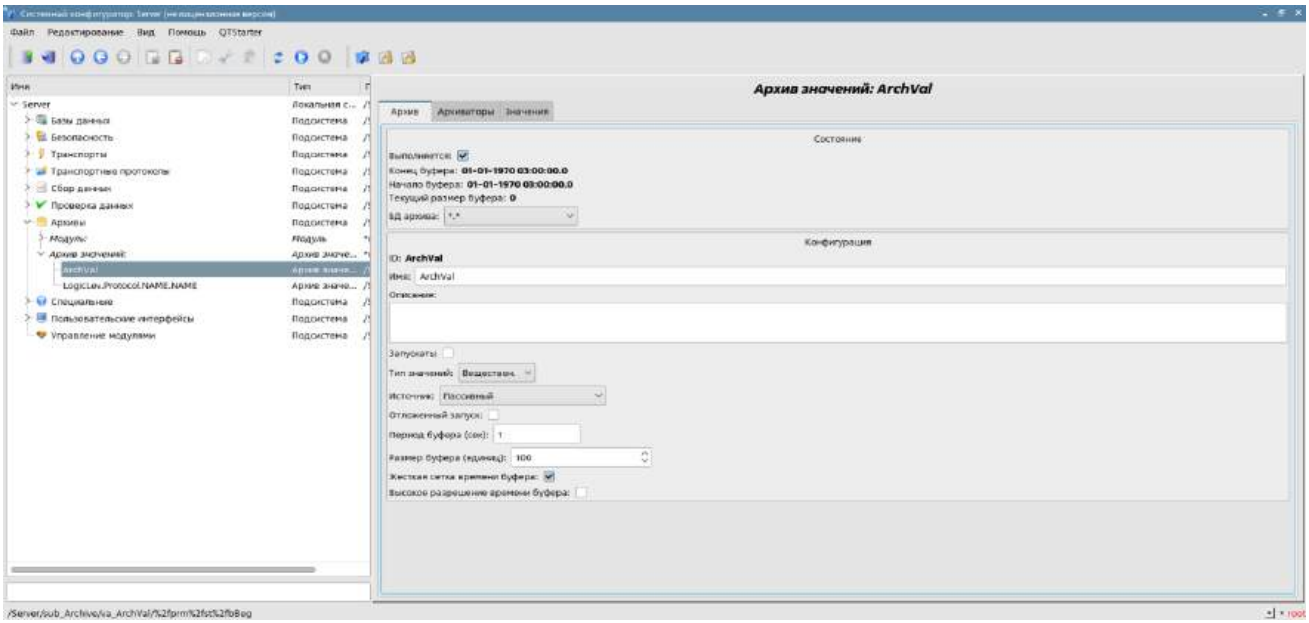


Рисунок 87

Вкладка "Архив" 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, выбирается из списка щелчком мыши.

- 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 на источник. Пассивный атрибут параметра направляет данные в архив самостоятельно с собственным периодом сбора данных. Активный атрибут параметра опрашивается задачей архивирования этой подсистемы;

- *Отложенный запуск* – позволяет при запуске системы не выводить предупредительное сообщение, если источник отсутствует;

- *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.

Вид вкладки "Архиваторы" показан на рисунке 88.

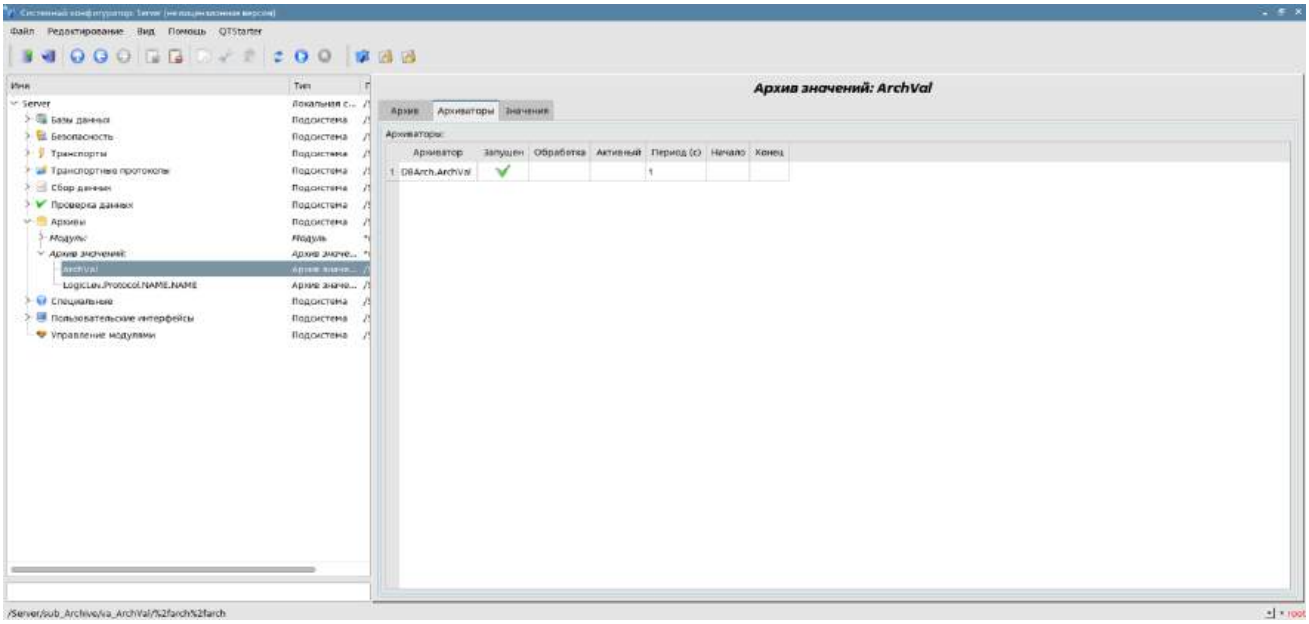


Рисунок 88

Вкладка "Архиваторы" 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.

Вид вкладки "Значения" показан на рисунке 89.

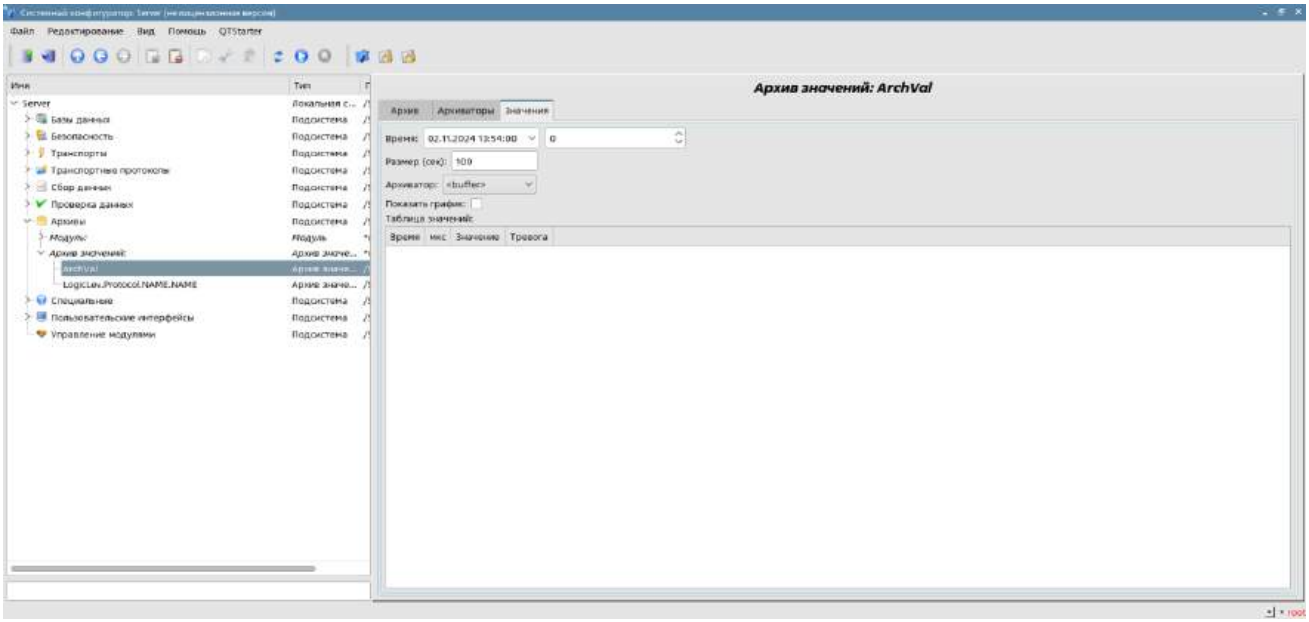


Рисунок 89

Вкладка "Значения" 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 (рисунок 90), 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.

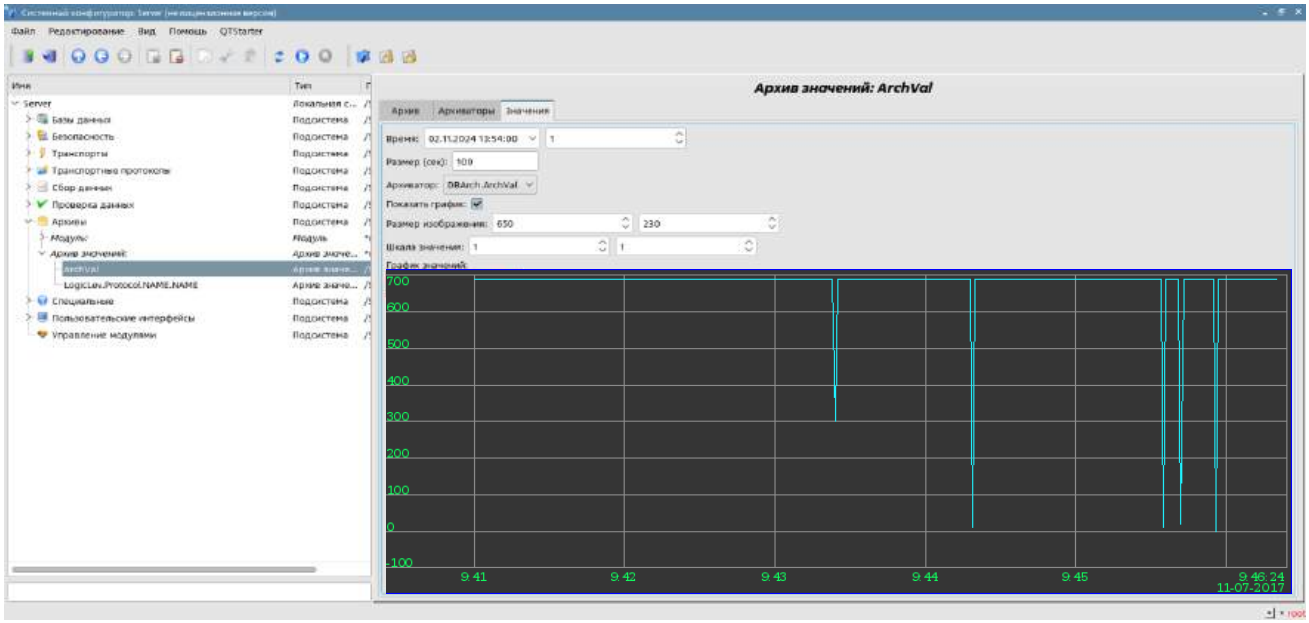


Рисунок 90

Каждый модуль подсистемы "Архивы" предоставляет конфигурационную страницу с вкладками "Архиваторы" и "Помощь". Вкладка "Архиваторы" содержит списки архиваторов сообщений и значений, зарегистрированных в модуле (рисунок 91). В контекстном меню списков пользователю предоставляется возможность добавления, удаления и перехода к нужному архиватору (рисунок 92). Во вкладке

"Помощь" содержится информация о модуле подсистемы "Архивы", состав которой идентичен для всех модулей (рисунок 93).

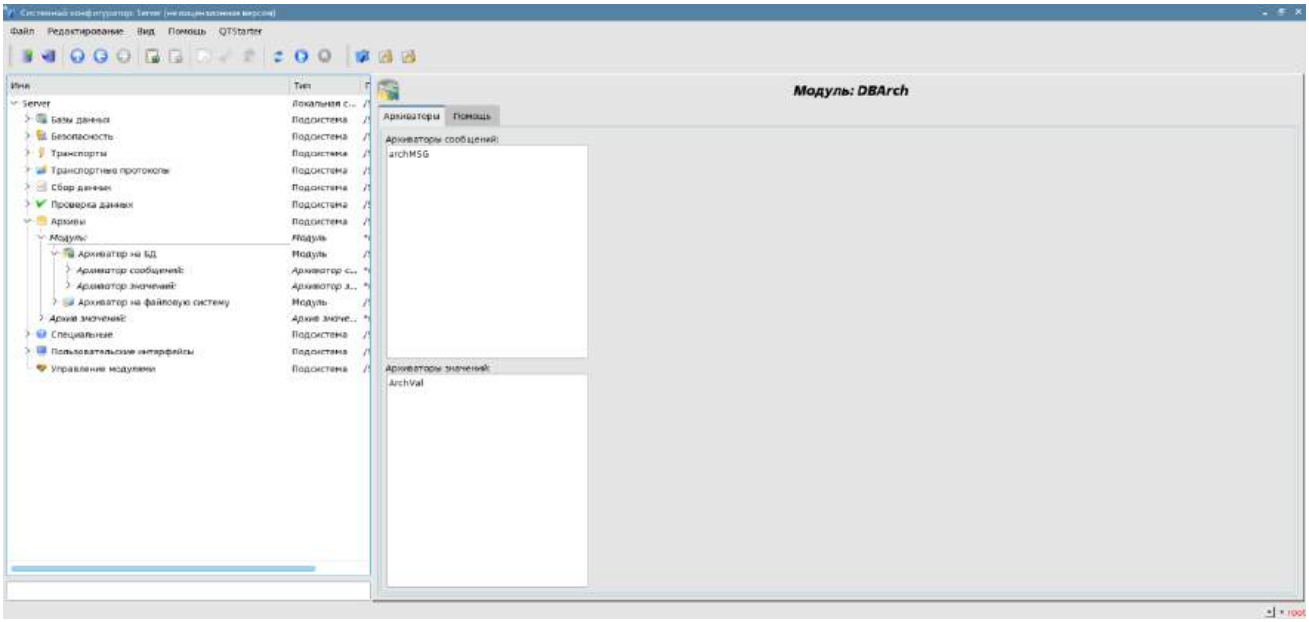


Рисунок 91

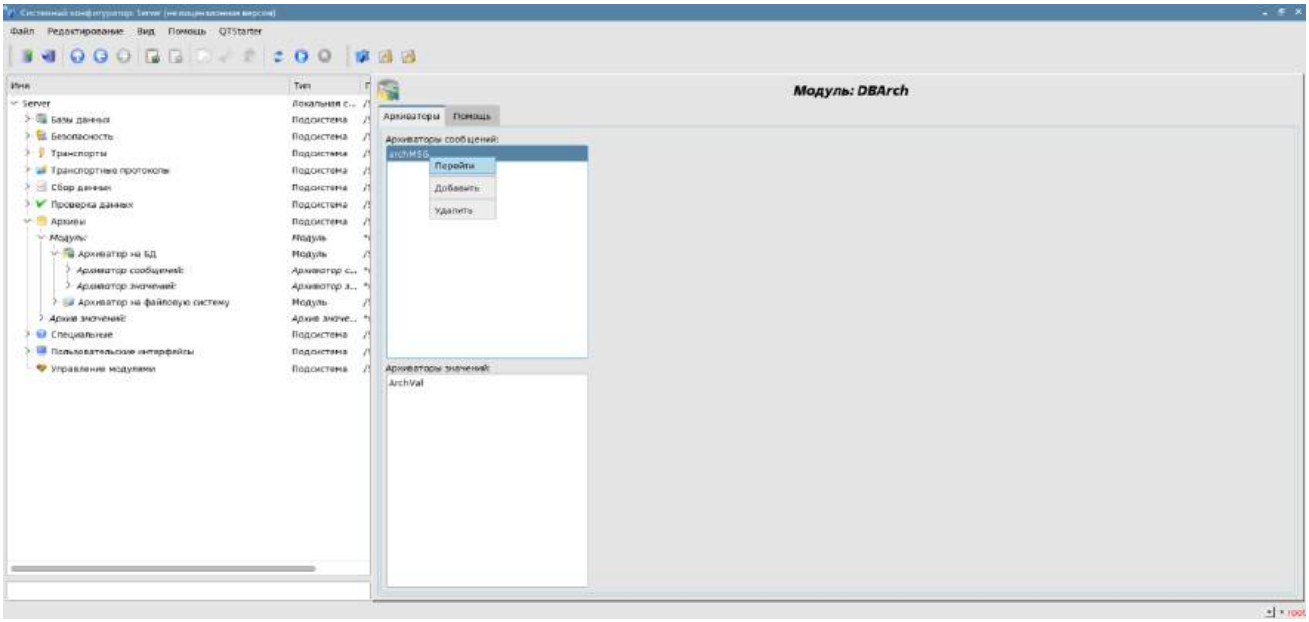


Рисунок 92

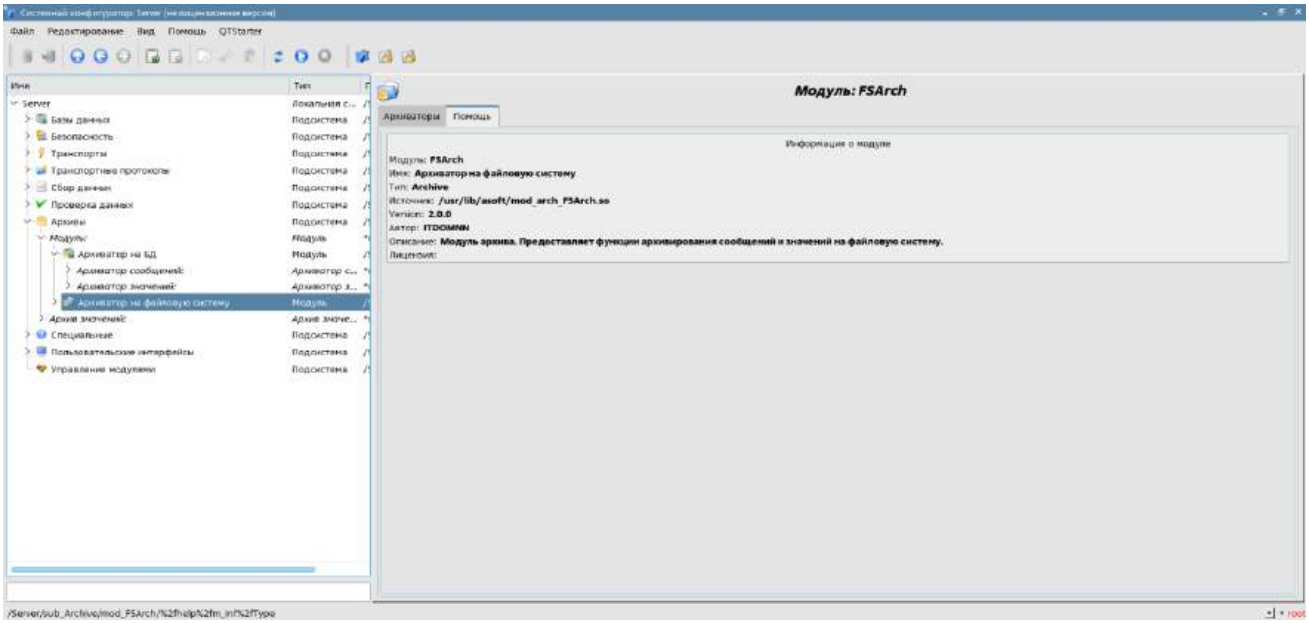


Рисунок 93

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

Вид вкладки "Архиватор" показан на рисунке 94.

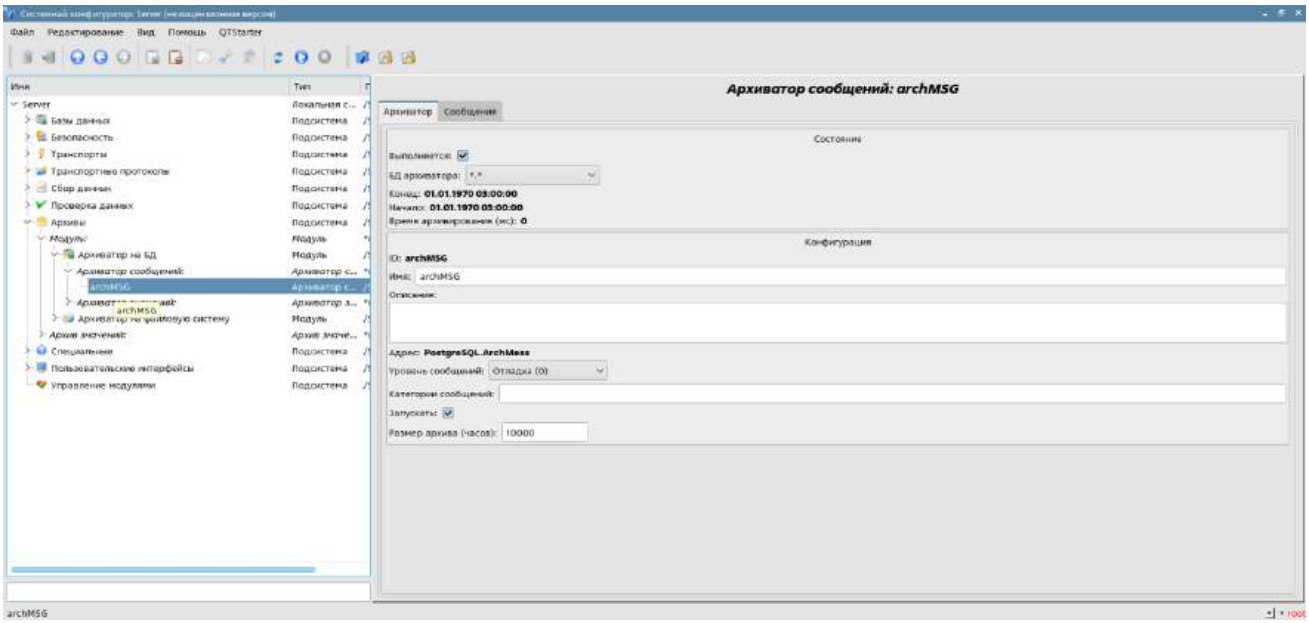


Рисунок 94

Вкладка "Архиватор" 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 рассматриваются настройки архиватора сообщений у модуля архива на БД. 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, выбирается из списка щелчком мыши;

- *Конец* – отображается дата+время последних данных в хранилище архиватора;

- *Начало* – отображается дата+время первых данных в хранилище архиватора;

- *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 сообщений, разделённый символом ';'. 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 следует перевести архиватор при загрузке;

- *Размер архива (часов)* – указывается максимальный размер архива в часах.

Вид вкладки "Сообщения" показан на рисунке 95.

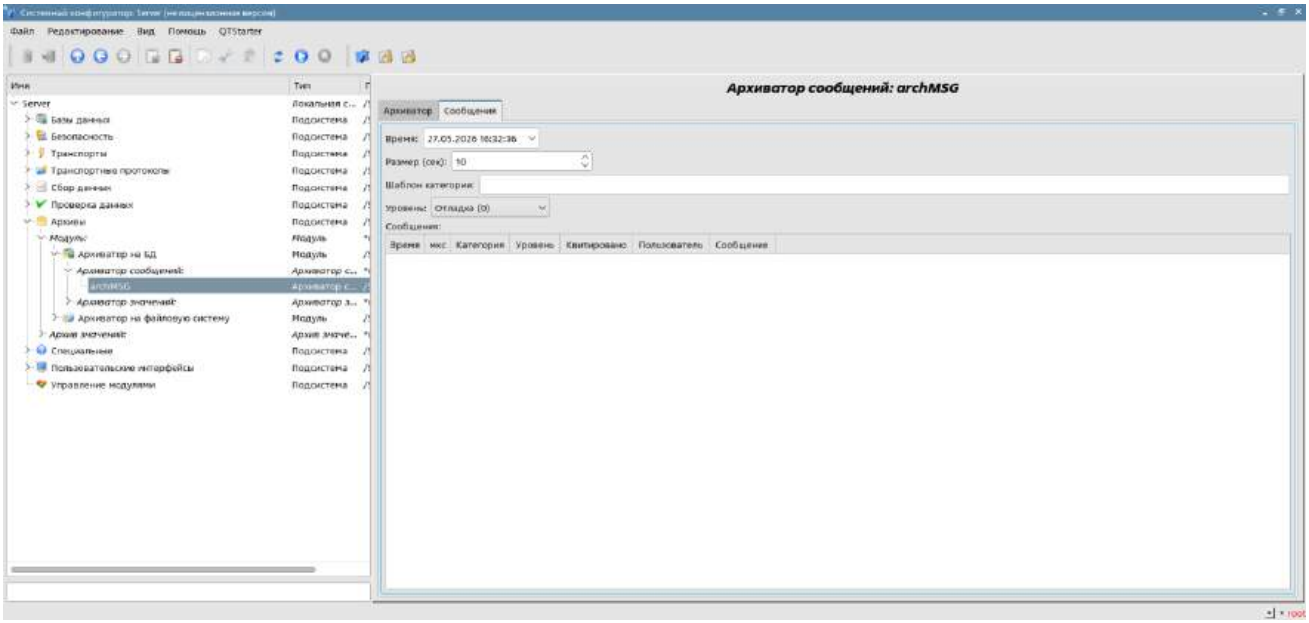


Рисунок 95

Вкладка "Сообщения" 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 сообщения 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.

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

Вид вкладки "Архиватор" показан на рисунке 96.

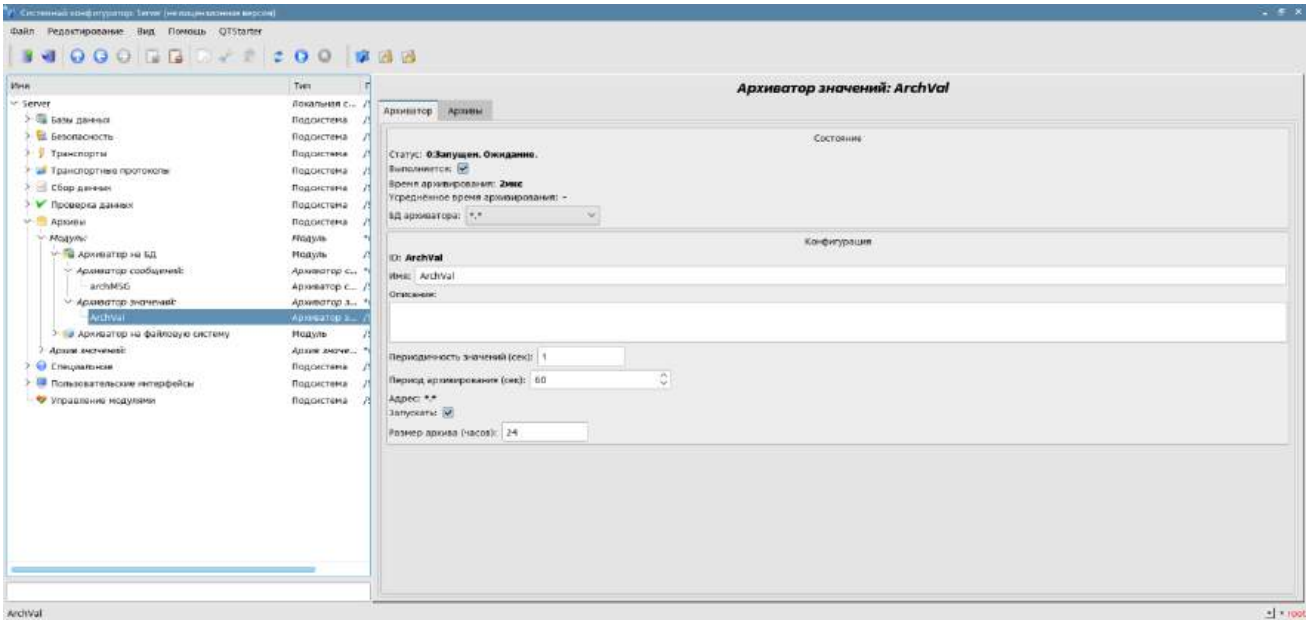


Рисунок 96

Вкладка "Архиватор" 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 рассматриваются настройки архиватора значений у модуля архива на БД.

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, выбирается из списка щелчком мыши.

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;

- *Размер архива (часов)* – указывается максимальный размер архива в часах;

- *Использование сегментирования для очистки архива* – чек-бокс для выбора использования сегментирования.

Внимание! При отключении флага «Выполняется» во время работы СКАДА удаляет все архивы, привязанные к этому архиватору!

Вид вкладки "Архивы" для модуля «Архиватор на БД» показан на рисунке 97.

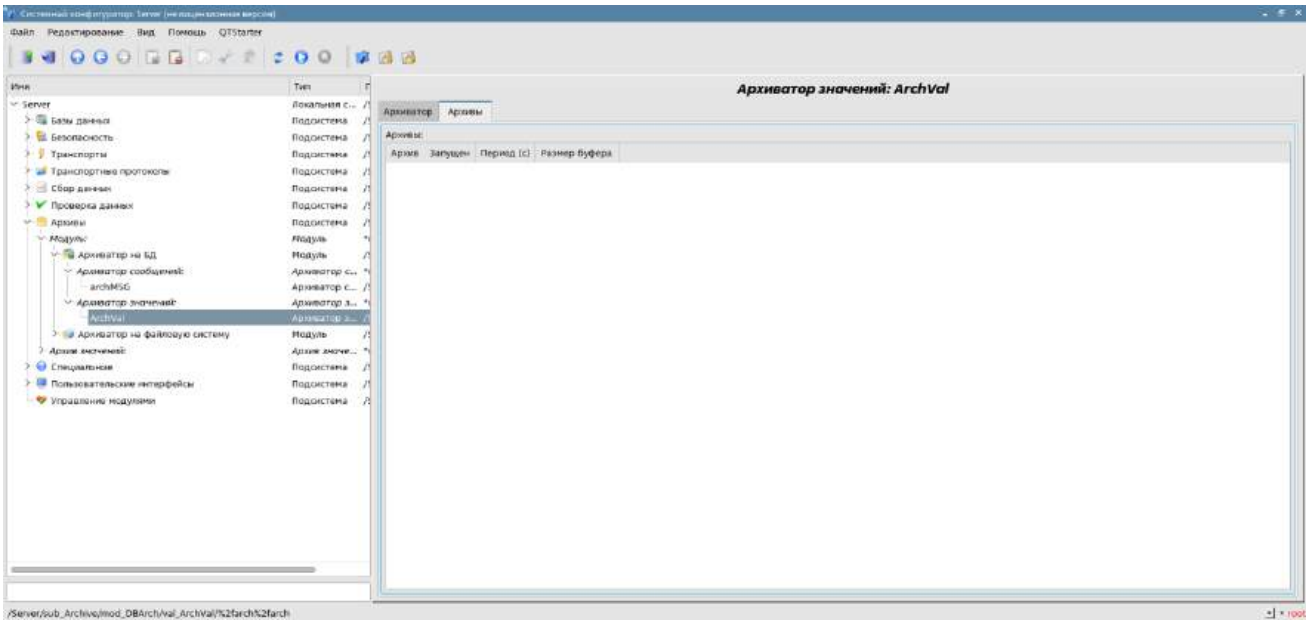


Рисунок 97

Вкладка "Архивы" 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 «Архиватор на файловую систему» поле с информацией о суммарной размерности файлов хранилища архиватора для архива.

В случае с модулем «Архиватор на файловую систему» в этой вкладке ещё содержится форма экспорта данных архиватора, содержащая следующие параметры:

Архив - имя архива для экспорта, выбирается мышью из списка;

Начало - дата и время начала данных для экспорта;

Конец - дата и время конца данных для экспорта;

Тип - тип экспортируемого файла (ascii или wav), выбирается мышью из списка;

В файл - имя файла для экспорта.

После настройки всех вышеуказанных параметров можно осуществить экспорт данных нажатием мыши на кнопку «Экспорт».

5.10 Подсистема "Пользовательские интерфейсы"

5.10.1 Общие сведения

Подсистема является модульной. Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Пользовательские интерфейсы", содержащая вкладки "Модули" и "Помощь".

Вкладка "Модули" 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".

Вид вкладки "Пользовательский интерфейс" показан на рисунке 98.

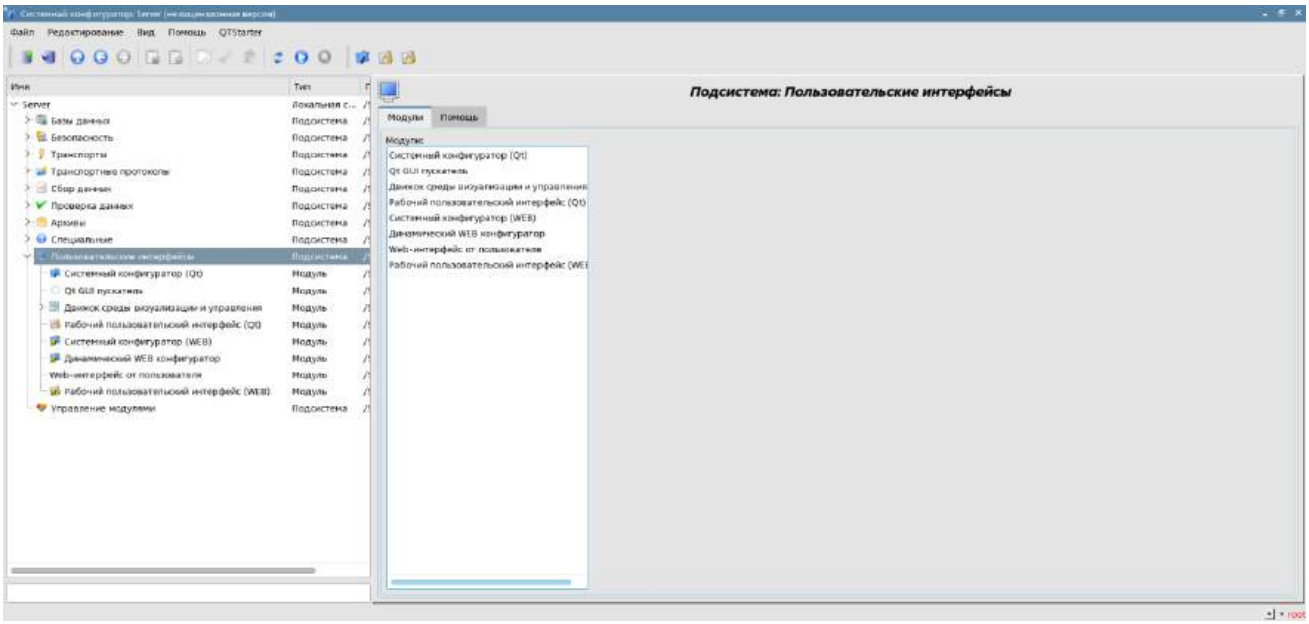


Рисунок 98

Вкладка "Пользовательский интерфейс" предоставляет параметр для контроля состояния "Выполняется" модуля, а также разделы конфигурации, специализированные для модулей этой подсистемы. Во вкладке "Помощь"

содержится информация о модуле подсистемы "Пользовательские интерфейсы", состав которой идентичен для всех модулей.

5.10.2 Рабочий пользовательский интерфейс (QT)

Модуль «Vision» предоставляет механизм конечной визуализации среды визуализации и управления (СВУ) в СКАДА. Модуль основан на многоплатформенной библиотеке графического пользовательского интерфейса (GUI) QT. В своей работе модуль использует данные движка СВУ (модуль VCAEngine).

Среда визуализации и управления (СВУ) является неотъемлемой составляющей СКАДА. Она применяется на клиентских станциях с целью доступного предоставления информации об объекте управления и выдачи управляющих воздействий на объект.

Данный модуль непосредственной визуализации СВУ предназначен для формирования и исполнения интерфейсов СВУ в среде графической библиотеки QT.

Модуль обеспечивает:

- три уровня сложности в формировании интерфейса визуализации, позволяющие органично осваивать и применять инструментарий 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);
- построение интерфейсов визуализации различной сложности, начиная от простых плоских интерфейсов мониторинга и заканчивая полноценными иерархическими интерфейсами, используемыми в СКАДА системах;
- 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 СКАДА, 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, 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 последующим централизованным исполнением в фоне (визуальное построение и исполнение математических моделей, логических схем, релейных схем и иных процедур);
- предоставление функций объектного API в систему СКАДА, 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;
- организацию клиентских станций на различной основе 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;
- поддержку пользовательского формирования палитры и шрифтовых предпочтений для интерфейса визуализации;
- поддержку пользовательского формирования карт событий под различное оборудование и пользовательские предпочтения;
- гибкое хранение и распространение библиотек виджетов, кадров и проектов интерфейсов визуализации в БД, поддерживаемых СКАДА. Практически пользователю нужно только зарегистрировать полученную БД с данными.

5.10.3 Конфигурирование модуля «Рабочий пользовательский интерфейс»

Вкладка «Пользовательский интерфейс» (рисунок 99) содержит следующие опции:

Станция движка СВУ - выбирается мышью из списка «Local, Loop, Loop SSL». Определяет станцию, на которой будет запускаться движок СВУ;

Стартовый пользователь - выбирается мышью из списка всех пользователей;

Время жизни страниц в кэше - время в часах, определяет интервал неактивности для закрытия страниц в кеше. Нулевое значение времени исключает закрытие страниц в кеше;

Перечень запускаемых проектов - перечень автоматически запускаемых проектов, разделённый символом ';'. Для открытия окна проекта на нужном дисплее (1) используется имя проекта в формате «PrjName-1»;

Переход к конфигурации перечня удалённых станций - переход к настройкам подсистемы «Транспорты».

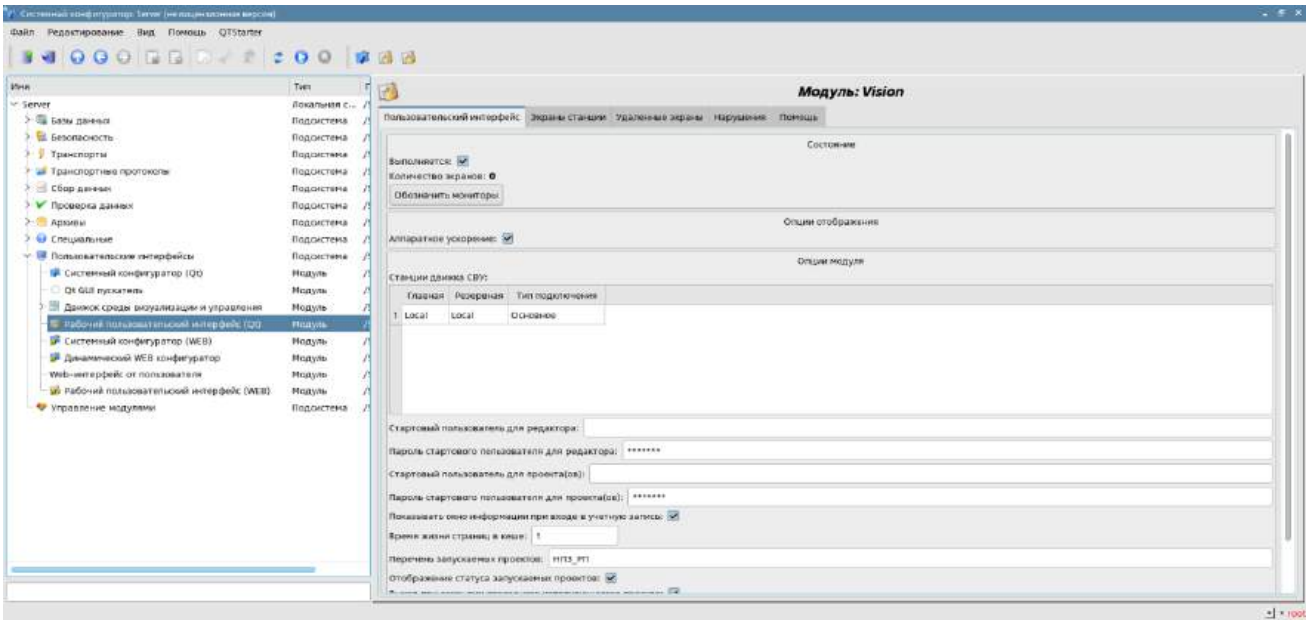


Рисунок 99

Вкладка «Удаленные экраны» предоставляет возможность конфигурирования вывода видеокадров на экраны табло коллективного пользования (рисунок 100). Для этого выбрать в списке станций для удаленного открытия видеокадров на экранах «Local». Задать для каждого экрана группу пользователей, для которых разрешено управление отображением видеокадров на этом экране. Также можно изменить имена экранов, отредактировав их в таблице.

Чтобы настроить вывод видеокадров на экраны табло для удаленного рабочего места, необходимо выбрать название этого рабочего места из списка станций для удаленного открытия видеокадров. Затем задать количество экранов и их имена для выбранного рабочего места.

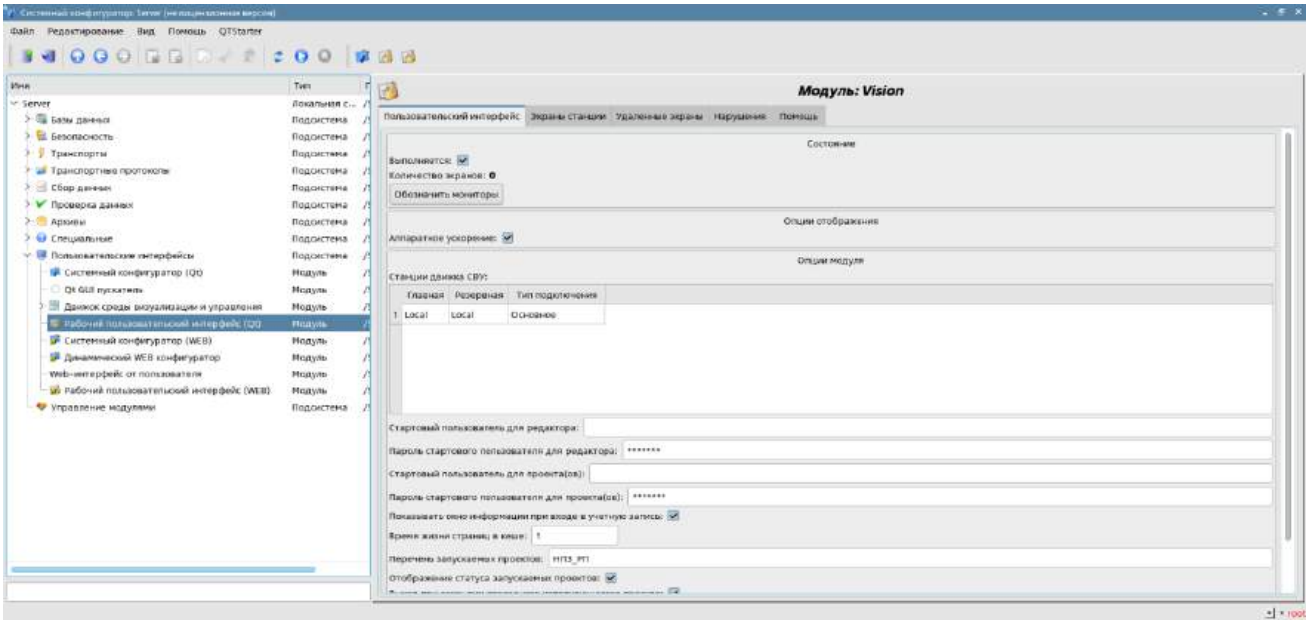


Рисунок 100

Вкладка «Нарушения» содержит настройку «Команда проигрывания» формата «play -q %f» - командная строка для вызова проигрывания звука. Для вставки имени файла источника используется «%f». Если файл источника не указан, то данные направляются в канал.

5.10.4 Системный конфигуратор (QT)

Модуль "QTCfg" предоставляет конфигуратор системы СКАДА. Конфигуратор реализован на основе многоплатформенной библиотеки графического пользовательского интерфейса (GUI) QT.

В основе модуля лежит язык интерфейса управления системой СКАДА, предоставляющий единый интерфейс конфигурации. Обновление модуля может потребоваться только в случае обновления спецификации языка интерфейса управления. Для запроса контекста страницы используется групповой запрос интерфейса управления, что позволяет оптимизировать время удалённого доступа по высоколатентным и медленным каналам связи.

Вкладка «Пользовательский интерфейс» (рисунок 101) предоставляет следующие опции:

Стартовый путь конфигуратора - директория в формате «/path», где хранятся настройки запуска конфигуратора;

Стартовый пользователь конфигуратора - пользователь, с настройками которого запускается конфигуратор при запуске, выбирается мышью из списка пользователей;

Перейти к конфигурации списка удалённых станций - переход к настройкам подсистемы «Транспорты».

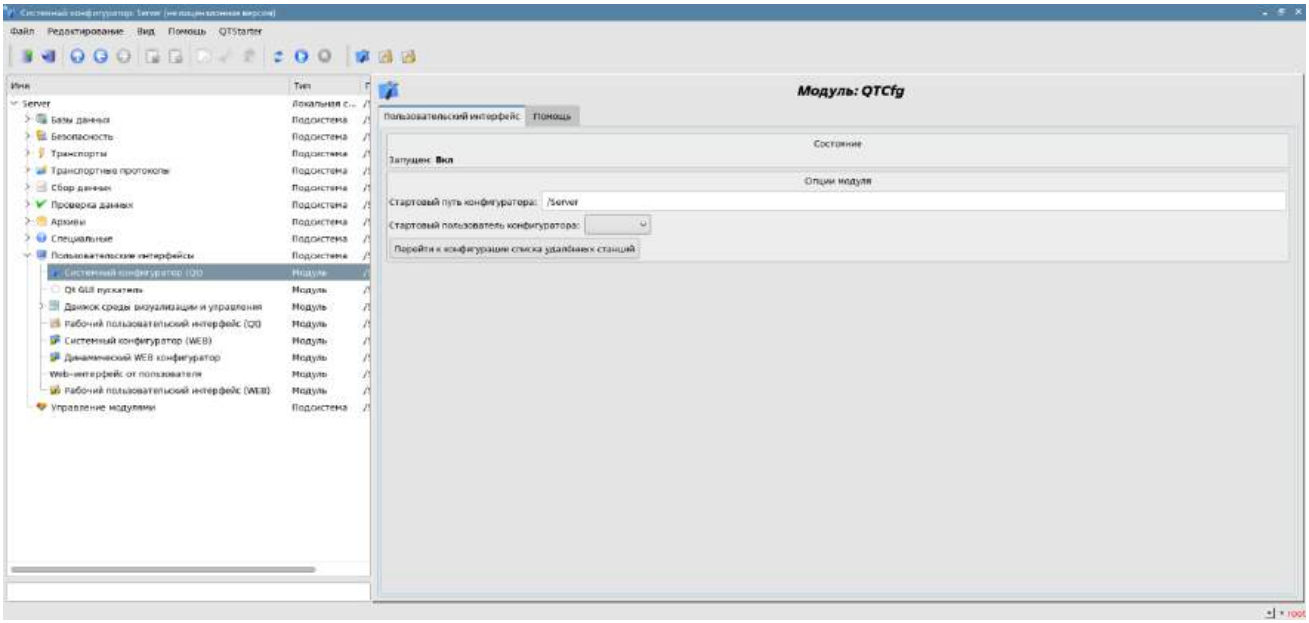


Рисунок 101

5.10.5 QT GUI пускатель

Модуль «QT GUI пускатель» предоставляет в СКАДА 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, пускатель с ним работает.

После получения объекта главного окна пускатель добавляет свою панель управления и пункт меню в это окно и запускает его. Панель управления пускателя 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 «QT GUI пускатель» 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 (рисунок 102).

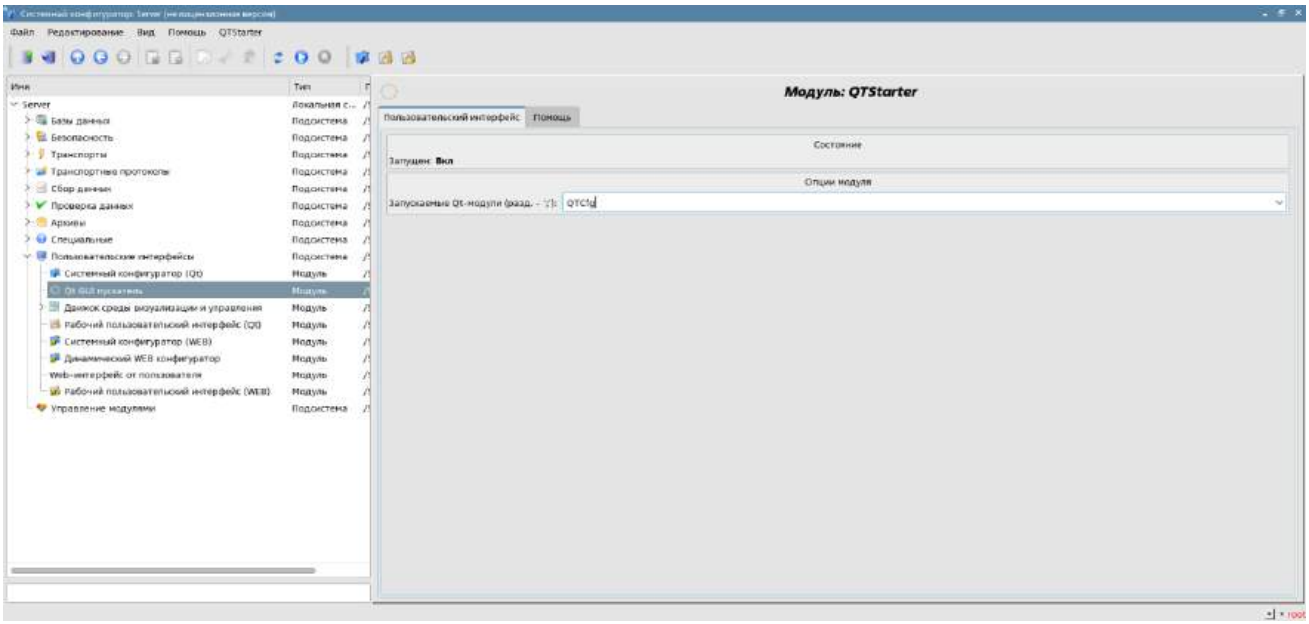


Рисунок 102

Опции модуля:

Запускаемые QT-модули (разд. - ;) - список модулей, разделённый ';', которые будут доступны для запуска с помощью QT-пускателя.

В случае закрытия окон всех QT GUI модулей QT-пускатель создаёт своё диалоговое окно, которое предлагает выбрать доступные QT GUI модули или завершить работу СКАДА. Вид диалогового окна приведен на рисунке 103.

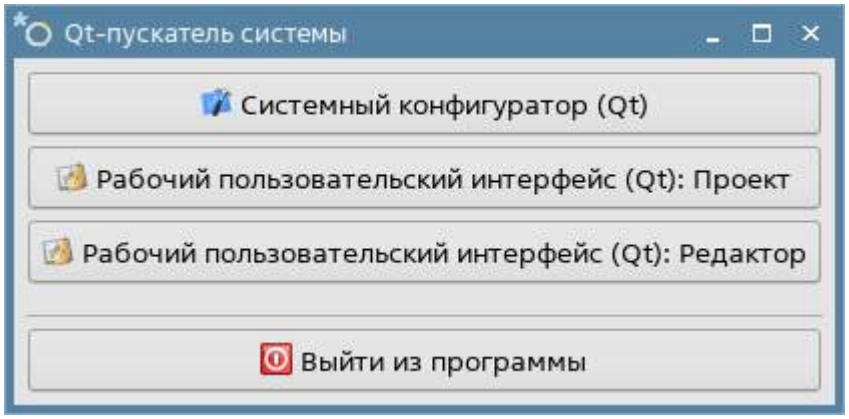


Рисунок 103

5.10.6 Движок среды визуализации и управления

Модуль VCAEngine предоставляет движок среды визуализации и управления (СВУ) в СКАДА. Данный модуль движка 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 СКАДА, так и посредством интерфейса управления СКАДА для удалённого доступа.

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.

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 использование 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

Данная архитектура СВУ позволяет реализовать поддержку трёх уровней сложности процесса разработки интерфейсов управления:

1) формирование интерфейса ВУ (визуализации и управления) с помощью библиотеки шаблонных кадров путём помещения шаблонов кадров в проект и назначения динамики.

2) 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.

3) 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 СКАДА.

Конфигурация модуля движка СВУ (рисунок 104):

Проект - список доступных в системе проектов;

Библиотеки виджетов - список библиотек виджетов, доступных в системе.

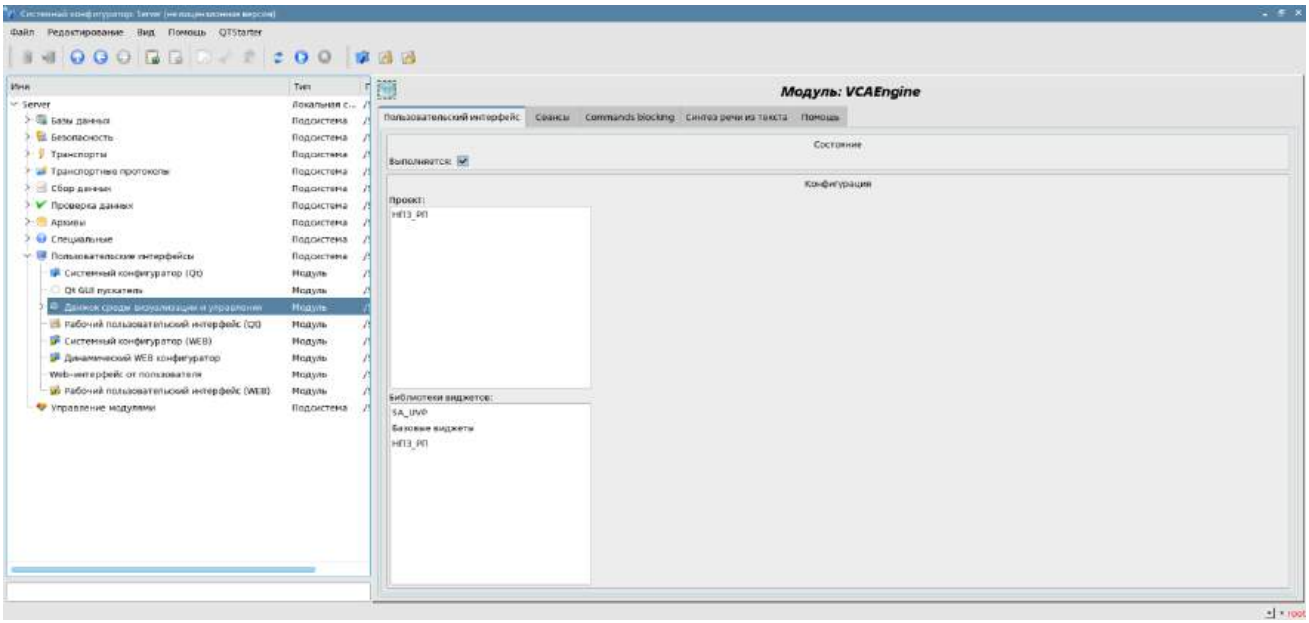


Рисунок 104

Вкладка «Сеансы» (рисунок 105) содержит следующие настройки:

Сеансы - содержит список идентификаторов доступных в системе сеансов;

Автоматическое создание и запуск - содержит настройки сеансов, которые будут автоматически создаваться и запускаться при старте системы.

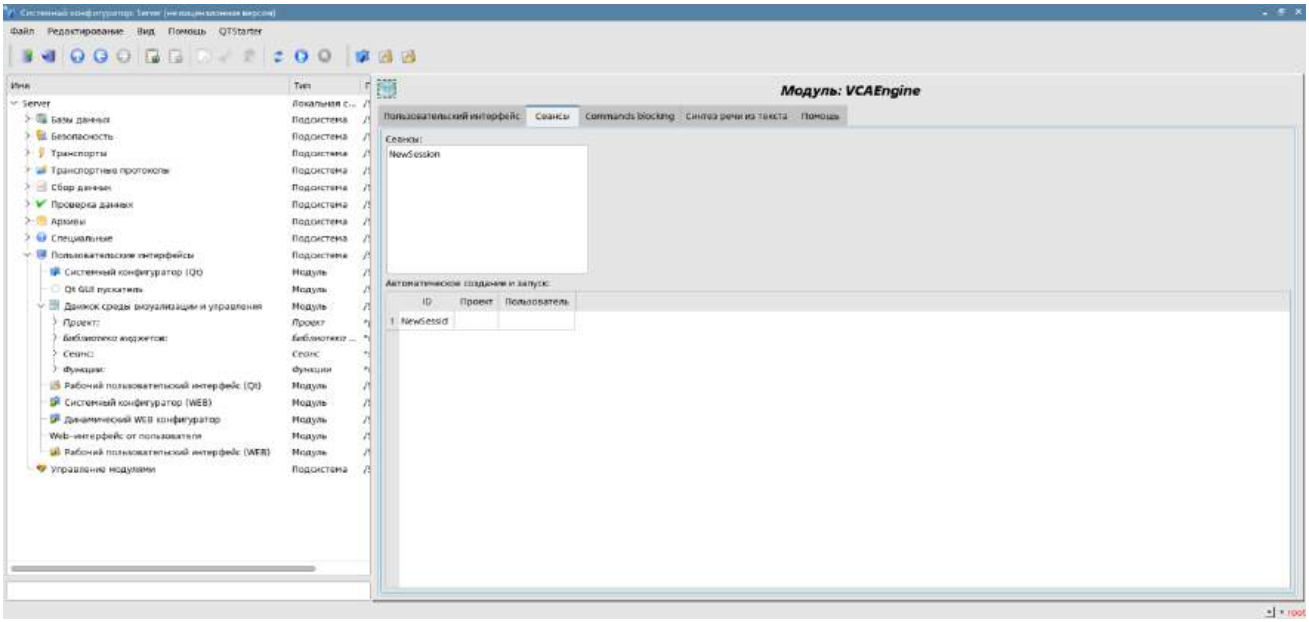


Рисунок 105

Вкладка «Блокировка команд» предоставляет механизм для блокировки управления при двухуровневой системе управления.

Сервера верхнего уровня являются основными подключениями, сервера нижнего уровня – дополнительными подключениями. Соответственно, при выборе

блокировки основных подключений управление возможно с АРМ нижнего уровня, при выборе блокировки дополнительных подключений – управление с АРМ нижнего уровня блокируется.

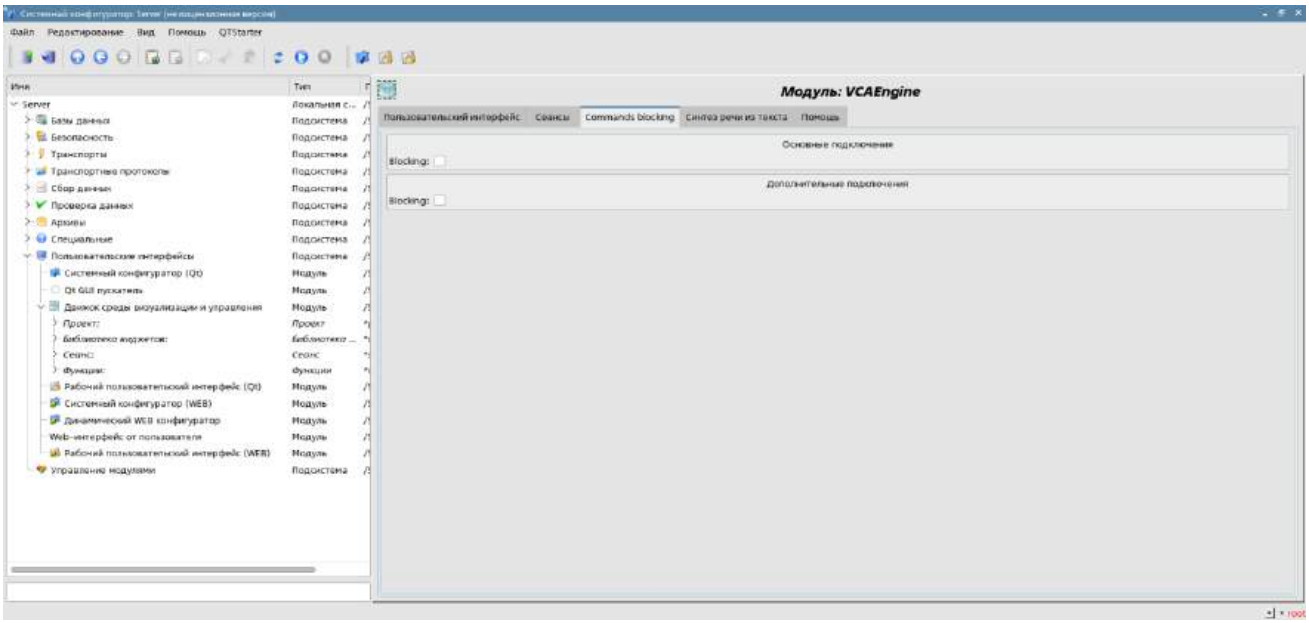


Рисунок 106

Вкладка «Синтез речи из текста» (рисунок 107) содержит следующие поля:

Команда - командная строка для вызова движка синтеза речи из текста, где «%t» - синтезируемый текст, «%f» - имя результирующего файла. Сама команда выбирается мышью из списка. Если файл результата не используется, то результат читается из канала. Если используется файл результата, но не используется «%t», то синтезируемый текст отправляется в канал.

Кодировка текста - кодировка текста движка, в которую будет перекодироваться текст.

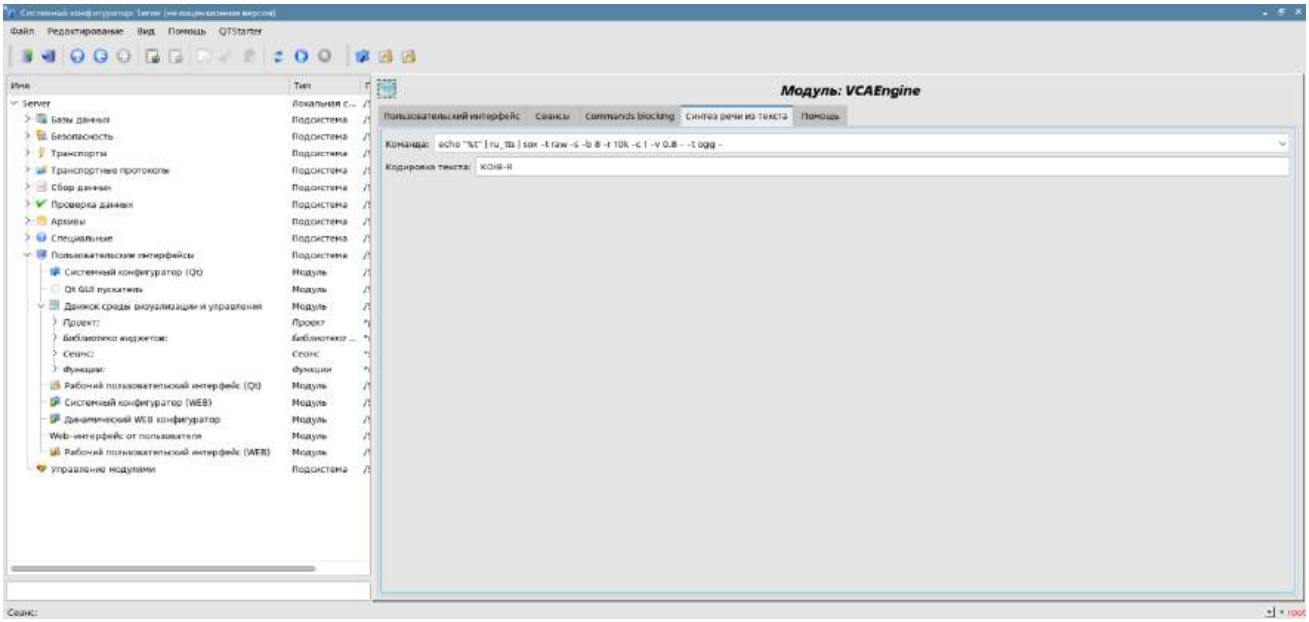


Рисунок 107

5.10.7 Модули, реализованные на основе WEB-технологий

Подсистема «Пользовательские интерфейсы» включает в себя следующие модули, реализованные на основе WEB технологий:

- системный конфигуратор (WEB);
- динамический WEB конфигуратор;
- WEB-интерфейс от пользователя;
- рабочий пользовательский интерфейс (WEB).

Системный конфигуратор (WebCfg) предоставляет конфигуратор системы, для работы которого достаточно обычного WEB-браузера (Opera, Mozilla и др.). Он также является модулем подсистемы транспортного протокола "HTTP": вызов "WebCfg" производится из "HTTP" посредством расширенного механизма коммуникации через функции: HttpGet() и HttpSet().

Динамический WEB конфигуратор (WebCfgD) предоставляет конфигуратор системы «СКАДА А-СОФТ», реализованный на основе следующих 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 роли WEB-сервера выступает СКАДА система, которая поддерживает стандартные коммуникационные механизмы TCP-сетей (модуль Сокеты подсистема «Транспорты»), протокол передачи гипертекста (модуль HTTP подсистема «Транспортные протоколы»), а также шифрование трафика между браузером и сервером (модуль SSL подсистема «Транспорты»). Для получения доступа к интерфейсу конфигурирования необходимо настроить транспорт сокет или SSL в связке с протоколом HTTP. По умолчанию интерфейс модуля доступен по URL: http://localhost:10002 или http://localhost:10004.

После получения XHTML-документа запускается программа на JavaScript для формирования динамического интерфейса конфигуратора.

WEB-интерфейс от пользователя (WebUser) предоставляет пользователю механизм создания Web-страниц, а также позволяет обрабатывать иные Web-

запросы на одном из внутренних языков СКАДА (JavaLikeCalc), не прибегая к низкоуровневому программированию. Он также является модулем подсистемы «Транспортного протокола» HTTP. Вызов WebUser производится из модуля HTTP подсистемы «Транспортные протоколы» посредством расширенного механизма коммуникации через функции: HttpGet() и HttpSet().

Рабочий пользовательский интерфейс (WEB) (WebVision) предоставляет механизм конечной визуализации среды визуализации и управления в систему «СКАДА А-СОФТ». Модуль основан на WEB технологиях: XHTML, JavaScript, CSS, AJAX. В своей работе модуль использует данные движка СВУ. 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 роли WEB-сервера выступает СКАДА система, которая поддерживает стандартные коммуникационные механизмы TCP-сетей (модуль Сокеты подсистема «Транспорты»), протокол передачи гипертекста (модуль HTTP подсистема «Транспортные протоколы»), а также шифрование трафика между браузером и сервером (модуль SSL подсистема «Транспорты»). Для получения доступа к интерфейсу конфигурирования необходимо настроить транспорт сокет или SSL в связке с протоколом HTTP. По умолчанию интерфейс модуля доступен по URL: http://localhost:10002 или http://localhost:10004.

5.11 Подсистема "Специальные"

5.11.1 Общее описание

Подсистема "Специальные" является модульной и предназначена для добавления в систему СКАДА непредусмотренных функций путём модульного расширения. Например, тесты системы и её модулей или библиотеки функций пользовательского программирования.

СКАДА содержит среду программирования, позволяющую на уровне пользователя 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 СКАДА представляет собой комплекс средств, организующих вычислительное окружение пользователя. В состав комплекса средств входят:

- объектная модель системы;
- модули библиотек функций;
- вычислительные контроллеры подсистемы “Сбор данных” и другие вычислители.

Модули библиотек функций предоставляют множество функций 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 и использовать их путём связывания.

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

Для конфигурации подсистемы предусмотрена корневая страница подсистемы "Специальные", содержащая вкладки "Модули" и "Помощь". Вкладка "Модули" содержит список модулей подсистемы:

- Библиотека функций Complex 1;
- Библиотека математических функций;

- Функции системного API;
- Тесты системы СКАДА.

Вкладка "Помощь" содержит краткую помощь для данной страницы.

Каждый модуль подсистемы "Специальные" предоставляет конфигурационную страницу с вкладками "Специальный модуль" и "Помощь". Вкладка "Специальный модуль" (рисунок 108) 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".

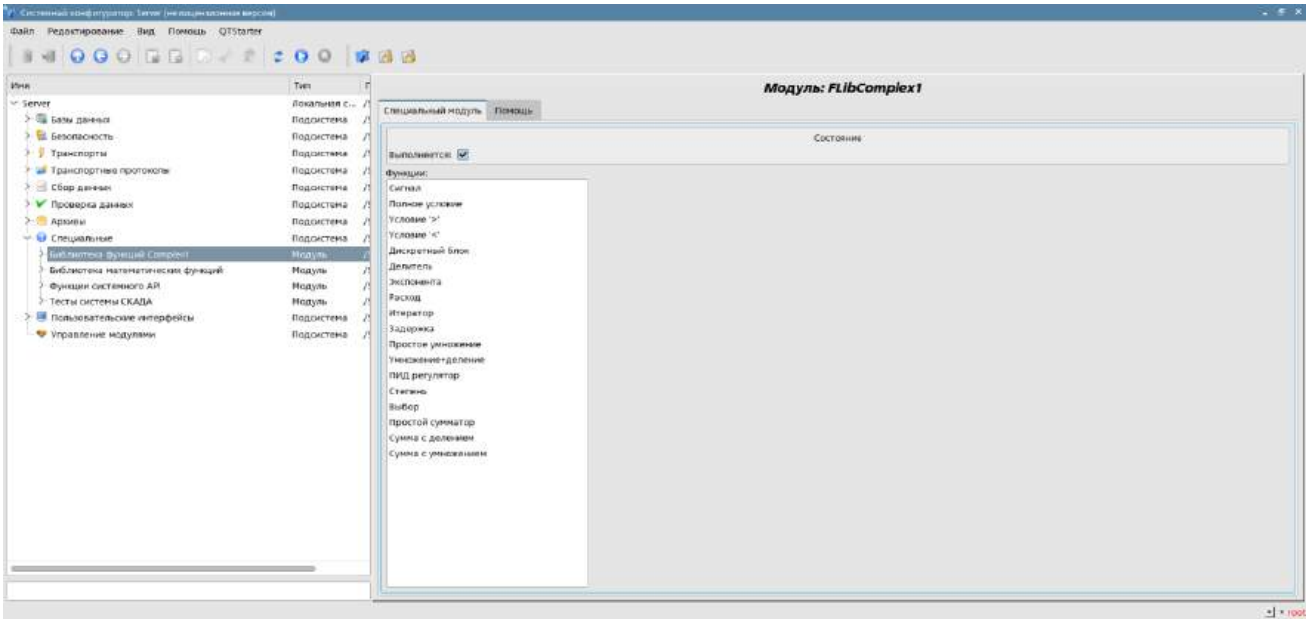


Рисунок 108

5.11.2 Библиотека стандартных математических функций

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.

Вкладка «Специальный модуль» содержит список стандартных математических функций, доступных для использования (рисунок 109).

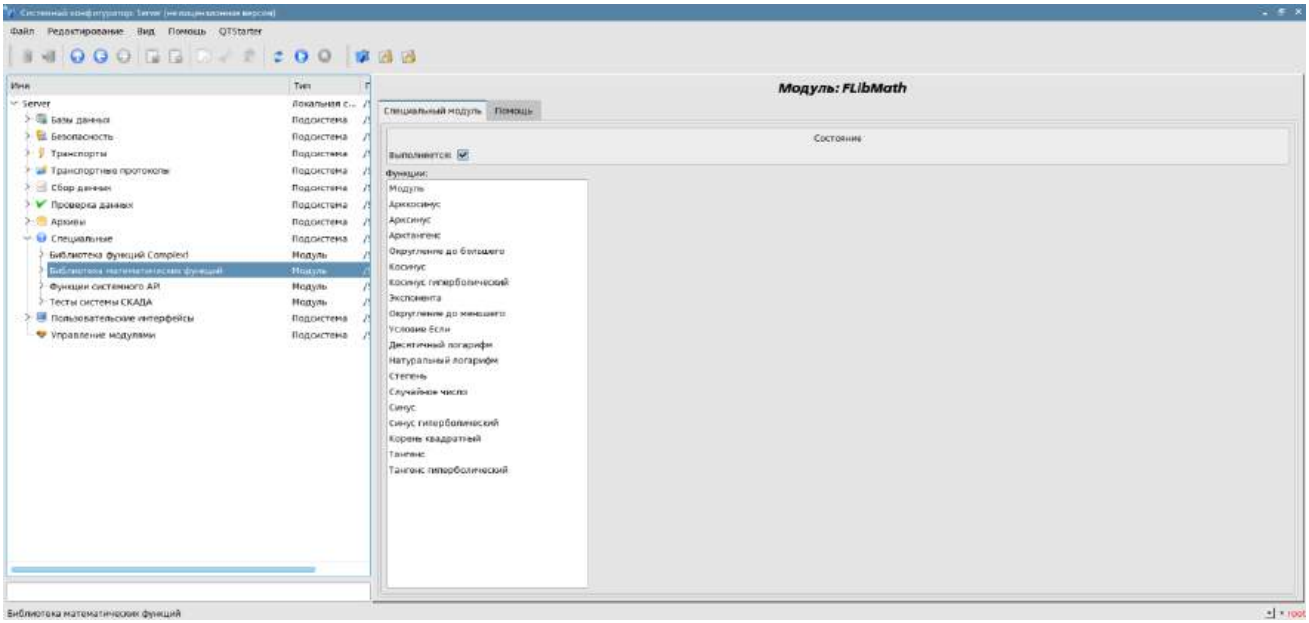


Рисунок 109

5.11.3 Библиотека функций системного API

Специальный модуль FLibSYS предоставляет в СКАДА статическую библиотеку функций для работы со СКАДА-ситемой, на уровне её системного API. Эти функции могут использоваться в среде пользовательского программирования СКАДА для организации нестандартных алгоритмов взаимодействия.

Для адресации к функциям этой библиотеки необходимо использовать путь: <Special.FLibSYS.*>. Где '*' - идентификатор функции в библиотеке.

На рисунке 110 представлен вид вкладки «Специальный модуль», которая содержит список функций системного API СКАДА.

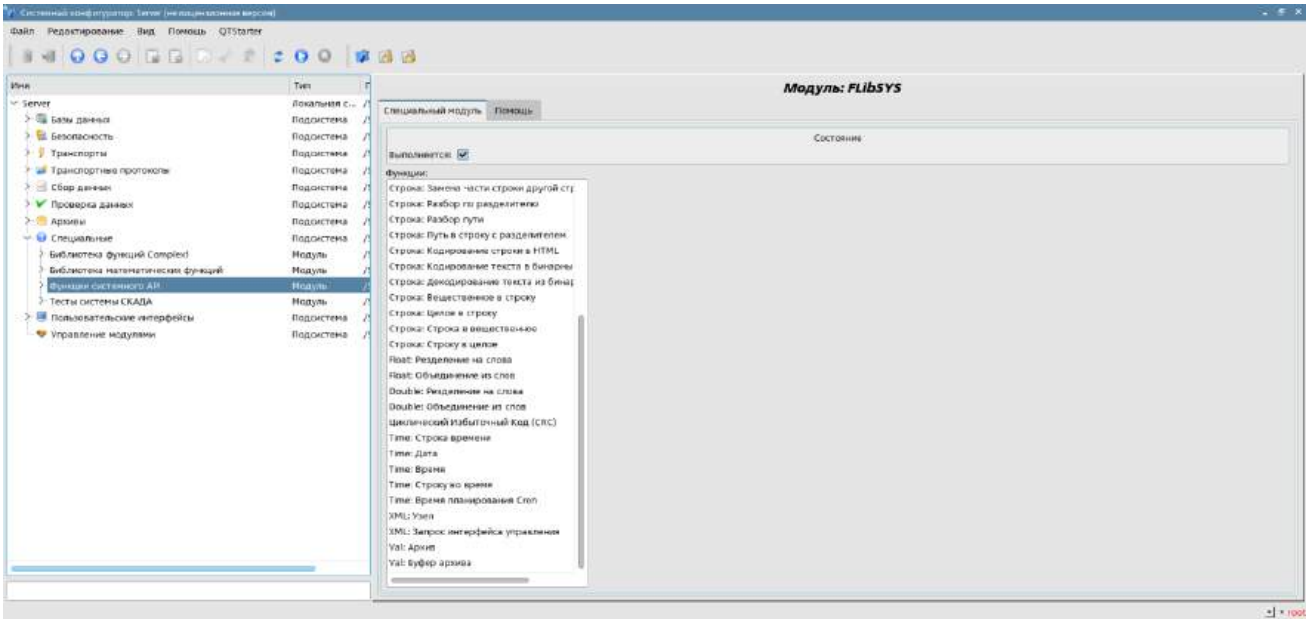


Рисунок 110

5.11.4 Тесты системы СКАДА

Специальный модуль SystemTests содержит набор тестов, предназначенных для тестирования различных подсистем и узлов СКАДА. 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.

Вид вкладки «Тесты», содержащей список доступных в системе тестов, представлен на рисунке 111.

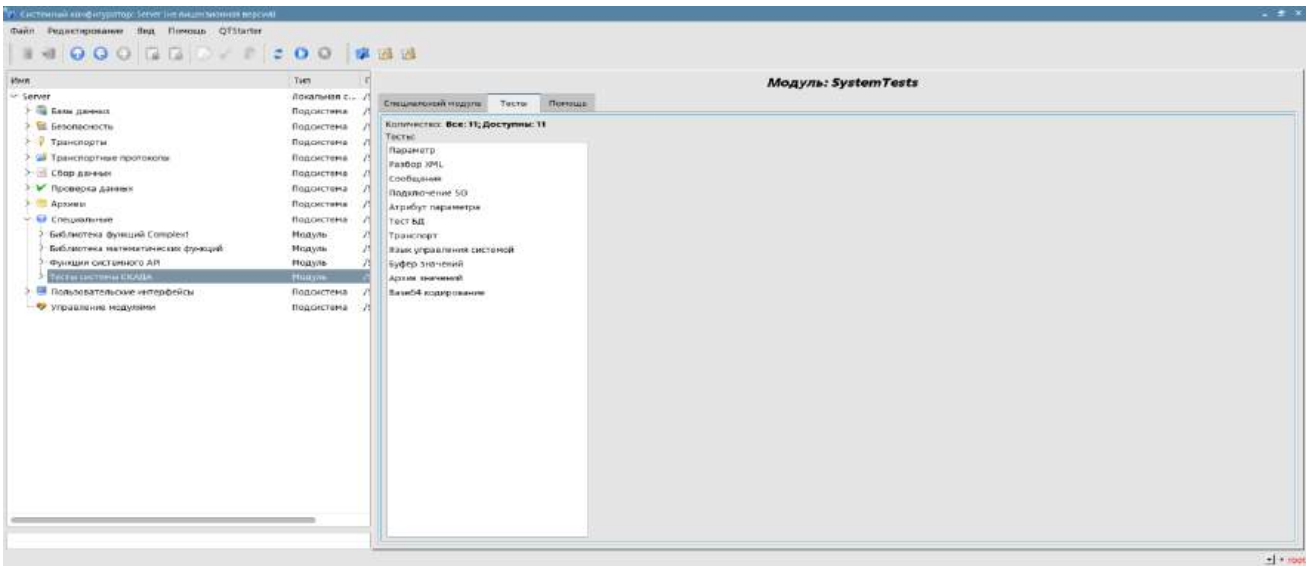


Рисунок 111

5.12 Подсистема "Управление модулями"

Вид страницы подсистемы "Управление модулями" показан на рисунке 112.

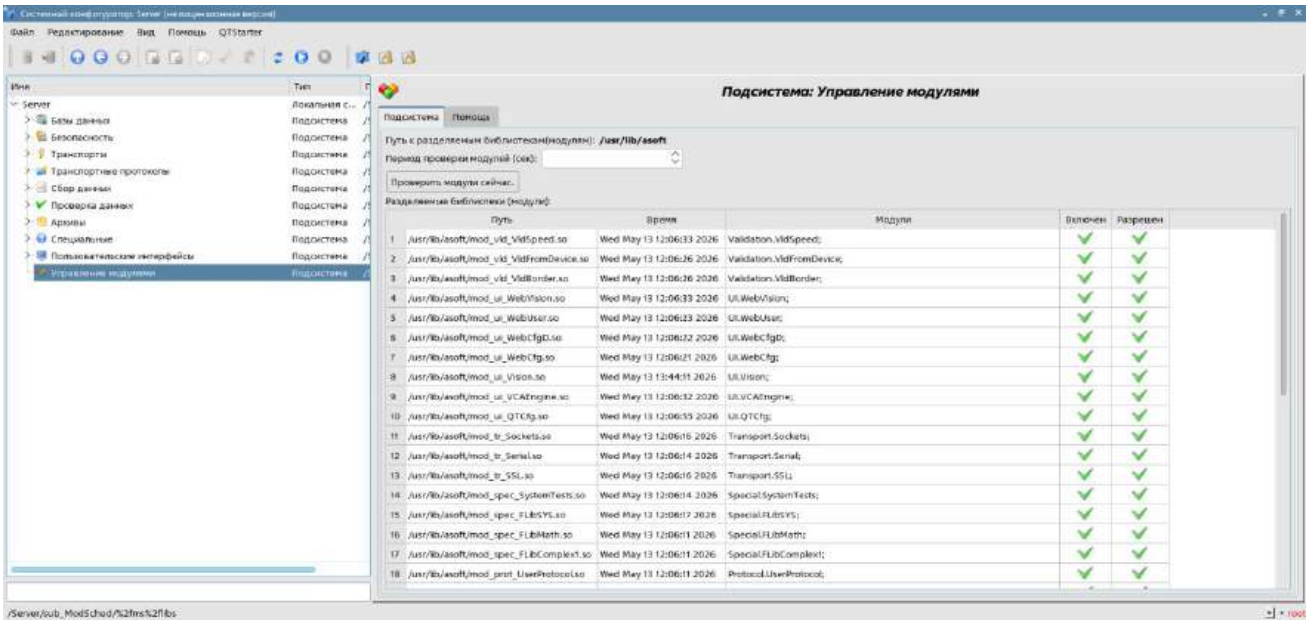


Рисунок 112

Подсистема не является модульной. Для конфигурации подсистемы предусмотрена страница подсистемы "Управление модулями", содержащая вкладки "Подсистема" и "Помощь". Вкладка "Подсистема" 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 СКАДА. Устанавливается параметром <*ModDir*> станции, конфигурационного файла;

- *Запрещённые модули* - информация о списке модулей, запрещённых для автоматического подключения и обновления. Разделителем списка является символ ';'. Допускается пустое значение этого поля. Устанавливается параметром <*ModDeny*> раздела подсистемы "sub_ModSched" в конфигурационном файле СКАДА. Список запрещённых модулей имеет больший приоритет, чем список разрешённых;

- *Разрешённые модули* - информация о списке модулей, разрешённых для автоматического подключения и обновления. Разделителем списка является символ ';'. Значение '*' используется для разрешения всех модулей. Устанавливается параметром <*ModAllow*> раздела подсистемы "sub_ModSched" в конфигурационном файле СКАДА;

- *Период проверки модулей (сек)* - указывает на периодичность проверки модулей на факт их обновления. Модули, допустимые для автоматического подключения и обновления, будут автоматически обновлены;

- *Проверить модули 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 СКАДА. 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.

5.13 Конфигурационный файл СКАДА и параметры командной строки вызова СКАДА

Конфигурационный файл СКАДА предназначен для хранения системной и общей конфигурации СКАДА-станции. Только в конфигурационном файле и через параметры командной строки можно указать часть ключевых системных параметров станции.

Называться конфигурационный файл СКАДА может произвольно, однако принято название scada.xml и производные от него. Конфигурационный файл может быть указан при запуске станции параметром командной строки:

--Config=/ path/scada.xml

где path – путь к конфигурационному файлу.

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

Структурно конфигурационный файл организован на расширяемом языке разметки текста XML. Следовательно, требуется жёсткое соблюдение правил синтаксиса XML. Пример образца типового конфигурационного файла СКАДА, с узлами конфигурации большинства компонентов СКАДА, приведен в приложении 1.

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 СКАДА с параметром --help или во вкладках "Помощь" страниц компонентов конфигурационных файлов СКАДА (рисунок 113).

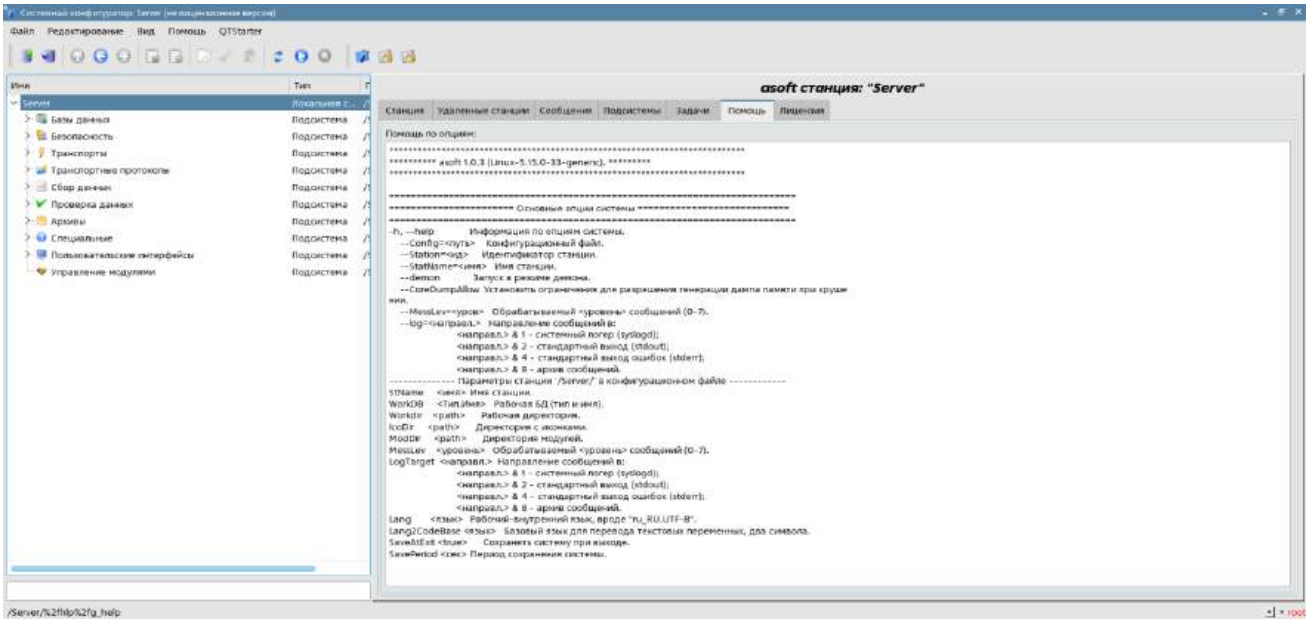


Рисунок 113

Результат вызова команды: # ./scada --help приведен в приложении 2.

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 СКАДА. 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.

6. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

6.1 Описание журнала активных тревог и журнала тревог и событий

6.1.1 Функциональное назначение

СКАДА обеспечивает выполнение следующих функций:

- отображение всех активных тревог в окне "Журнала активных тревог";
- отображение всех тревог и событий в окне "Журнал тревог и событий" с использованием определенных критериев выбора;
- выполнение квитирования аварийных или предупредительных тревог, имеющихся в "Журнале активных тревог";
- сохранение информации о квитировании тревог в "Журнал тревог и событий".

6.1.2 Описание журнала активных тревог и журнал тревог и событий

Журнал тревог и событий содержит таблицу с сообщениями, полученными за заданный пользователем промежуток времени. Журнал активных тревог отображает активные квитированные и не квитированные тревоги до тех пор, пока атрибуты, вызвавшие сообщение, не войдут в норму.

Существует возможность применения фильтра по типам отображаемых сообщений и установка максимального количество отображаемых сообщений. В журнале активных тревог реализована возможность квитирования выбранного пользователем сообщения и перехода к видеокадру, содержащему элемент с вызвавшим тревогу атрибутом.

6.1.3 Взаимодействие журнала активных тревог и журнала тревог и событий с модулями СКАДА

Журнал активных тревог и журнал тревог и событий входят в модуль отображения ЭДЖ. Взаимодействие журнала активных тревог и журнала тревог и событий с модулями СКАДА представлено на рисунке 114. Средствами графического интерфейса этого модуля происходит конфигурация журналов: настройка фильтров типов отображаемых сообщений и максимальное количество строк в таблицах журналов.

Модуль отображения ЭДЖ создает подписку журнал активных тревог на получения сообщений о тревогах от менеджера сообщений модуля архивации. Менеджер сообщений передает вновь поступившие в буфер сообщения о тревогах в

журнал активных тревог, также он сообщает о том, что вызвавший тревогу атрибут вошел в норму. Таким образом, после квитирования сообщение либо удалится, если ранее менеджер сообщений передал информацию о том, что атрибут вошел в норму, либо окрасит его в другой цвет (удаление данного сообщения из журнала произойдет после того как атрибут к которому относится сообщение войдет в норму).

Журнал тревог и событий получает данные за заданный промежуток времени от модуля архивации. При этом если указанный интервал времени присутствует в буфере сообщений, то значение берется из буфера. В противном случае данные запрашиваются у базы данных архивов.

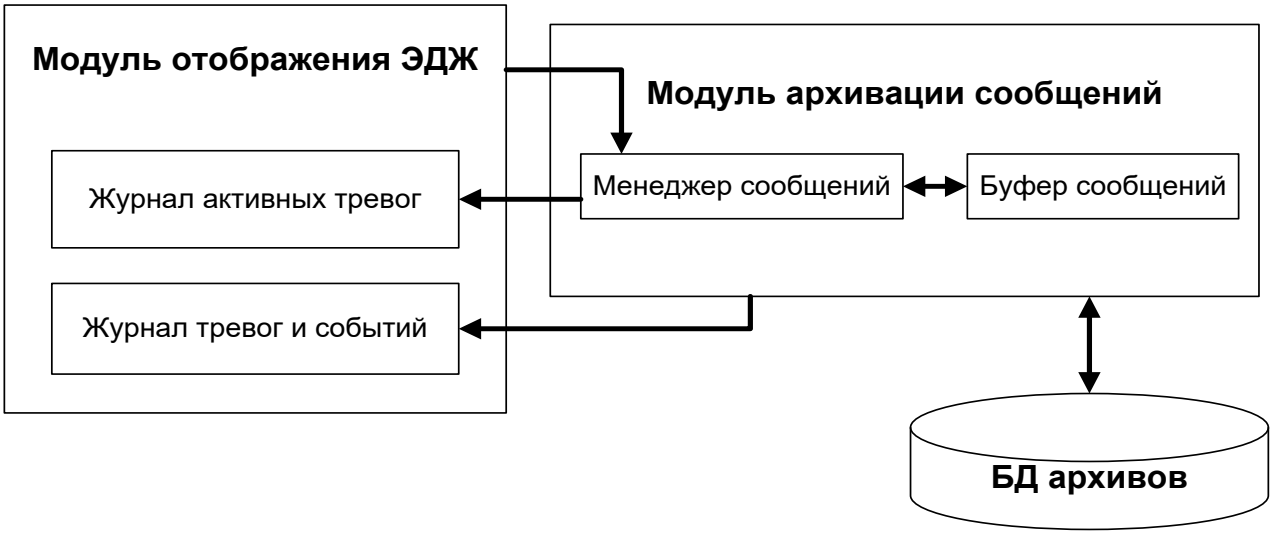


Рисунок 114

6.1.4 Элементы интерфейса журнала активных тревог и журнал тревог и событий

Для работы с журналом активных тревог и журналом тревог и событий интерфейс ЭДЖ содержит кнопки открытия журналов и вкладки, содержащие журнал активных тревог и журнал тревог и событий.

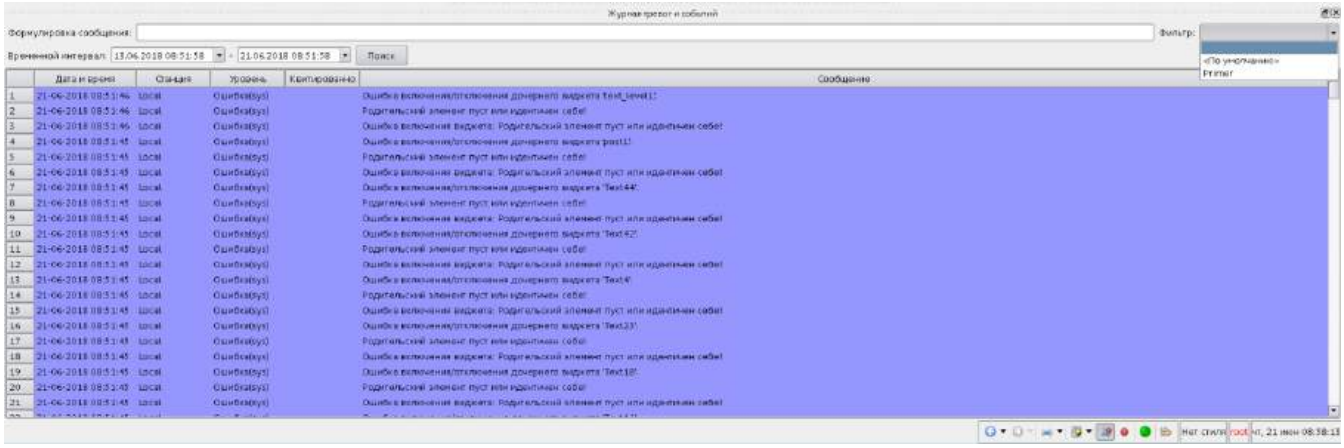


Рисунок 115

В строке журнала тревог и событий по умолчанию отображается следующая информация:

- дата и время;
- станция;
- уровень (ошибка, критическое сообщение, предупреждение, информационное сообщение);
- оборудование;
- отметка квитирования;
- пользователь;
- текст сообщения.

Для определения порядка отображения столбцов в окне журнала тревог и событий из контекстного меню необходимо выбрать строку «Настройка» и проставить крестик в чек-боксе напротив выводимых столбцов (рисунок 116).

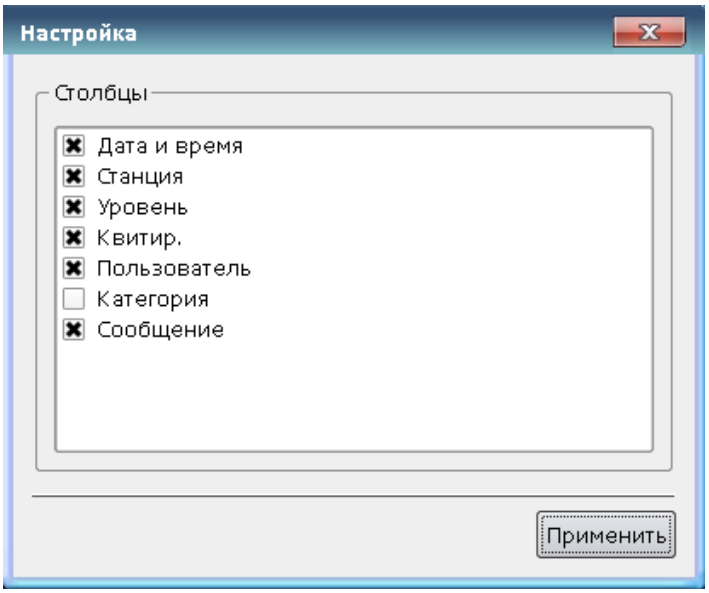


Рисунок 116

В правой части окна журнала тревог и событий находится поле «Фильтр, в котором можно настроить типы выводимых сообщений. На рисунке 117 показана настройка фильтра “Primer”. Добавление фильтра производится выбором строки «Добавить» контекстного меню и вводом ID и имени нового фильтра. После чего необходимо выбрать уровни (типы) воспроизводимых сообщений (поставить крестик в соответствующем чек-боксе), сохранить изменения и закрыть окно редактирования фильтра.

Для просмотра результата работы фильтра необходимо выбрать временной интервал и нажать кнопку «Поиск».

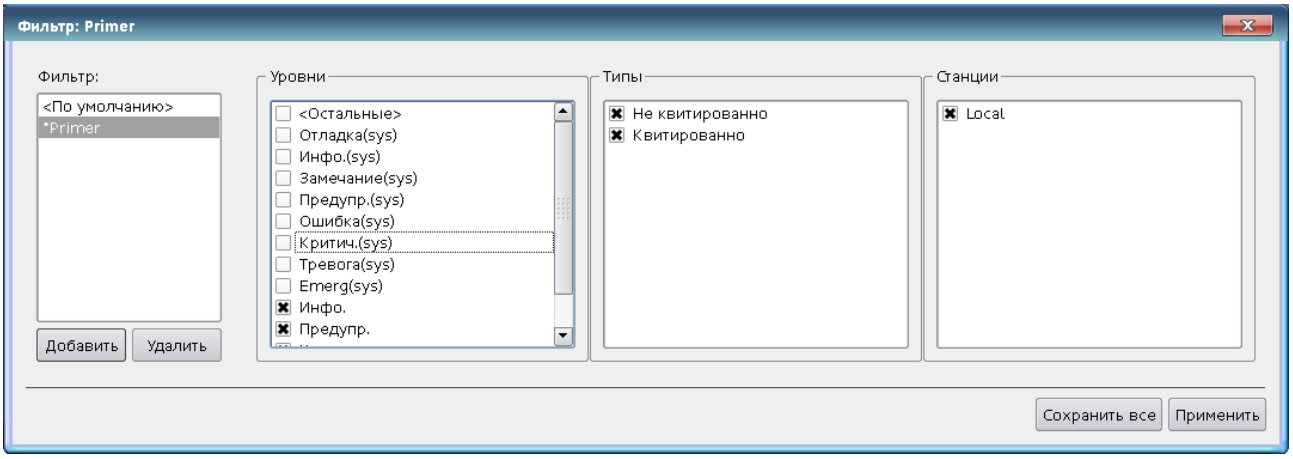


Рисунок 117

Журнал активных тревог служит для просмотра активных тревог и квитирования любой из них. Тревоги от одного источника отображаются вложенной строкой, которую при необходимости можно развернуть.

В строке тревоги отображаются:

- дата и время;
- уровень (ошибка, критическое сообщение);
- оборудование;
- текст сообщения о тревоге.

Тревоги отображаются цветом следующим образом:

тревога активна и не квитирована	– красный фон с черной мигающей рамкой;
тревога активна и квитирована	– красный фон;
тревога вошла в норму, но не квитирована	– белый фон с черной мигающей рамкой.

Квитированная тревога, вошедшая в норму, удаляется из журнала активных тревог.

Квитирование тревоги производится из контекстного меню квитируемой строки.

Для квитирования сообщения необходимо нажать на него правой кнопкой мыши и в появившемся меню выбрать пункт «Квитировать» (рисунок 118). Квитированное сообщение либо изменит цвет, если значение атрибута не вернулось к моменту квитирования в норму, либо удалится из таблицы в противном случае. Если строка имеет вложенные тревоги, то можно в контекстном меню выбрать строку «квитировать все» и будут квитированы все вложенные тревоги.

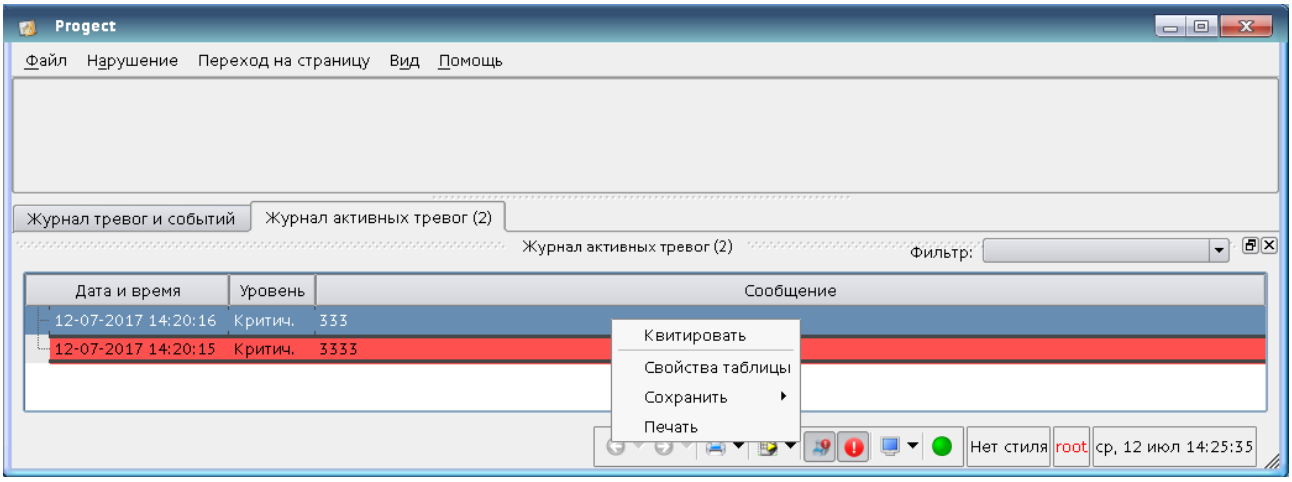


Рисунок 118

Кроме того, обеспечена возможность сохранения данных о тревоге в log-файл, при выборе соответствующего пункта в контекстном меню.

Конфигурирование журнала активных тревог и журнала тревог и событий осуществляется средствами редактора пользовательского интерфейса в окне редактирования свойств визуального элемента «Проект» – «Журнал сообщений».

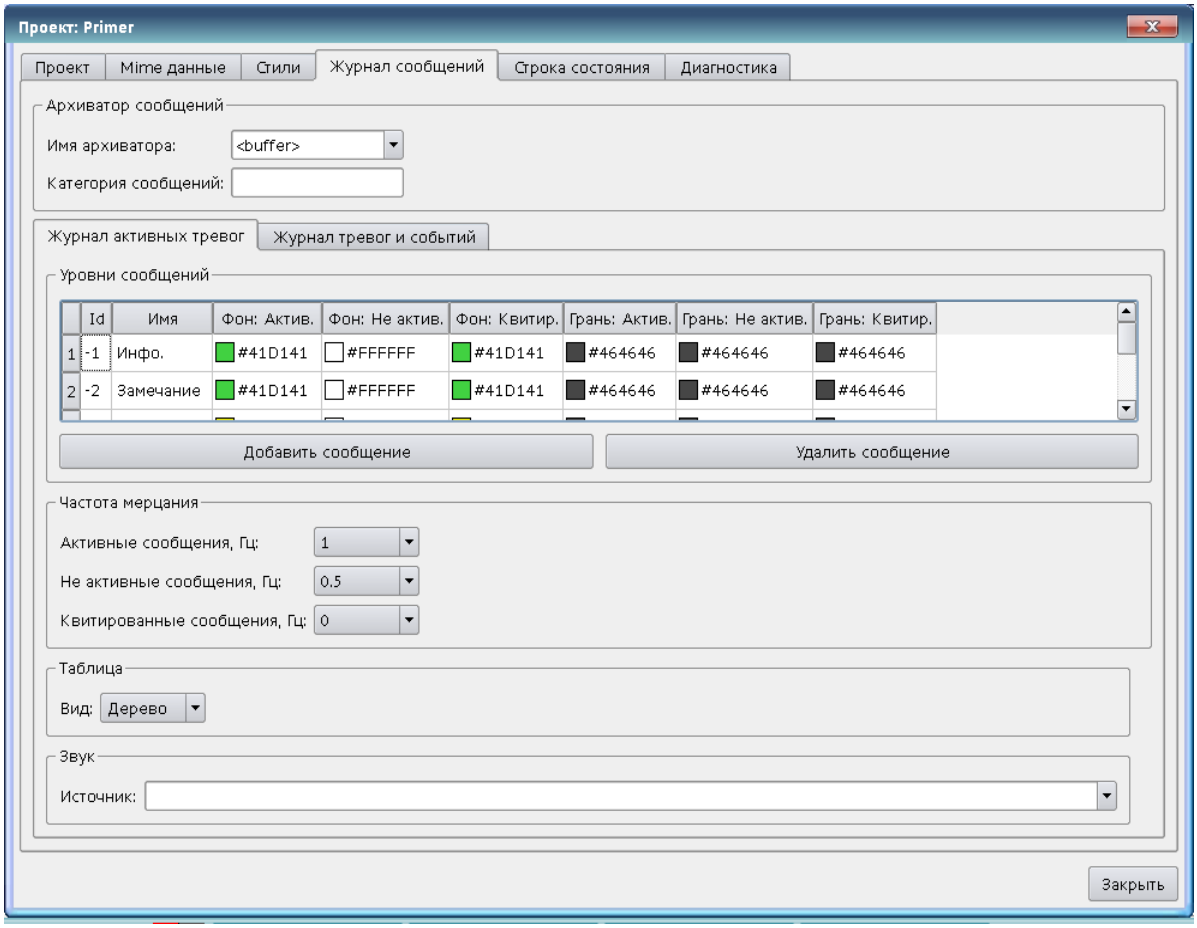


Рисунок 119

Для этого необходимо выбрать вкладку «Журнал сообщений» и настроить атрибуты «Журнала активных тревог» и «Журнала тревог и событий» в зависимости

от поставленной задачи, например, добавить сообщения, изменить частоту и цвет мерцания сообщения (рисунок 119).

6.2 Описание быстрого перехода по видеокадрам

6.2.1 Функциональное назначение

Разработанное ПО обеспечивает выполнение следующих функций:

создание списка истории посещенных видеокадров;

предоставление возможности пользователю перехода на выбранный из ранее сформированного списка видеокадр;

разделение доступа к спискам истории для разных пользователей.

6.2.2 Элементы интерфейса для быстрого перехода по видеокадрам

Для обеспечения возможности предоставления пользователю функций перехода по видеокадрам в строке состояния интерфейса пользователя находятся кнопки перехода (рисунок 120) на предыдущий или на следующий видеокадр (если ранее был осуществлен переход на предыдущий видеокадр), а также возможность выбора конкретного видеокадра из списка посещенных видеокадров.

Рисунок 120

6.2.3 Основные особенности быстрого перехода между видеокадрами

История посещения видеокадров обновляется в момент открытия новой мнемосхемы и хранится до завершения сеанса. Для каждого подключенного пользователя (с разных станций оператора) к серверу формируется своя история посещенных видеокадров. Для того чтобы один пользователь не мог получить доступ к истории посещения видеокадров другого пользователя (на той же самой станции оператора) происходит очищение списка истории при каждой смене пользователя.

7. РЕДАКТОР ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Разработка пользовательского интерфейса в СКАДА выполняется в одном окне – «Редакторе пользовательского интерфейса», реализующем многодокументный интерфейс (MDI), позволяющий одновременно редактировать несколько кадров различных размеров. Для перехода в данное окно необходимо на панели инструментов системного конфигуратора нажать на правую иконку  «Рабочий пользовательский интерфейс (Qt)». В появившемся окне доступны следующие механизмы управления разработкой: панели инструментов, пункты меню и контекстное меню. Большинство действий дублируются. Навигационные интерфейсы реализованы присоединяемыми окнами. Конфигурация панелей инструментов и присоединяемых окон сохраняется при выходе и восстанавливается при старте системы.

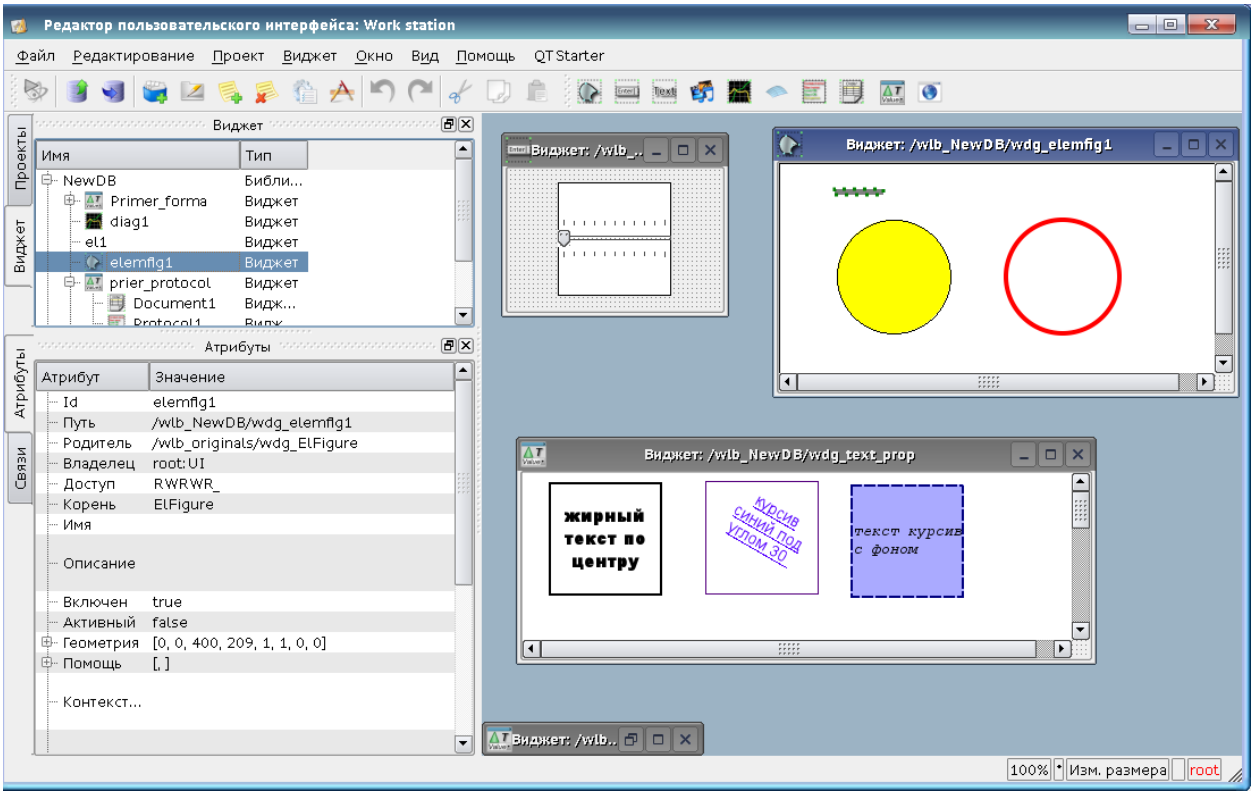


Рисунок 121

Доступ к основным компонентам СВУ производится посредством закрепленных окон:

- *Виджет* – реализовано в виде дерева библиотек виджетов, позвляет быстро найти нужный виджет или библиотеку и провести их редактирование. Содержит следующее контекстное меню:
  - "Новая библиотека" – создание новой библиотеки;
  - "Добавить визуальный элемент" – добавление визуального элемента в библиотеку;

- "Удалить визуальный элемент" – удаление визуального элемента из библиотеки;
- "Очистить изменения визуального элемента" – очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию;
- "Свойства визуального элемента" – конфигурация визуального элемента;
- "Редактировать визуальный элемент" – визуальное редактирование элемента;
- "Найти использования визуального элемента" - в нижней части экрана отображает окно с результатом поиска выделенного визуального элемента во всех библиотеках;
- "Перейти к родительскому визуальному элементу" – переход к элементу, на основании которого он сделан;
- "Вырезать визуальный элемент" – вырезание/перемещение визуального элемента;
- "Копировать визуальный элемент" – копирование визуального элемента;
- "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.

- *Проект* – реализовано в виде дерева страниц проектов. Для формирования пользовательских видеокадров достаточно разместить в окне проекта элементы из библиотек виджетов. Содержит следующее контекстное меню:
  - *"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 изменения визуального элемента" – очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию;
  - "Свойства визуального элемента" – конфигурация визуального элемента;
  - "Редактировать визуальный элемент" – визуальное редактирование элемента;

- "Найти использования визуального элемента" - в нижней части экрана отображает окно с результатом поиска выделенного визуального элемента во всех библиотеках;
- "Перейти к родительскому визуальному элементу" – переход к элементу, на основании которого он сделан;
- "Библиотека: {Имя библиотеки}" – пункты меню для доступа к видеокадрам/виджетам содержащимся в библиотеке;
- "Вырезать визуальный элемент" – вырезание/перемещение визуального элемента;
- "Копировать визуальный элемент" – копирование визуального элемента;
- "Вставить визуальный элемент" – вставка визуального элемента;
- "Загрузить из БД" – загрузка данных визуального элемента из БД;
- "Сохранить в БД" – сохранение данных визуального элемента в БД;
- "Создание альтернативного представления" – создание альтернативной страницы для отображения на оборудовании, выполняющем разные роли (например, коллективное табло отображения, рабочее место оператора или встроенная сенсорная панель);
- "Удаление альтернативного представления" – удаление выбранного альтернативного представления страницы;
- "Выбор альтернативного представления" – выбирает из существующих альтернативную страницу;
- "Обновить проекты" – выполняет перечитывание конфигурации и состава проектов из модели данных.

В основном пространстве рабочего окна размещаются окна страниц проектов, видеокадров библиотек виджетов, пользовательских элементов и элементов примитивов на момент их визуального редактирования.

В меню рабочего стола 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 СКАДА;
- *"Редактирование"* – операции редактирования визуальных элементов*:*
  - "Откат изменений визуального элемента" – осуществление отката последнего изменения визуального элемента;

- "Повтор изменений визуального элемента" – осуществление повтора изменения визуального 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;

- *"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 изменения визуального элемента" – очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию;
  - "Создание альтернативного представления" – создание альтернативной страницы для отображения на оборудовании, выполняющем разные роли (например, коллективное табло отображения, рабочее место оператора или встроенная сенсорная панель);
  - "Удаление альтернативного представления" – удаление выбранного альтернативного представления страницы;
  - "Выбор альтернативного представления" – выбирает из существующих альтернативную страницу;
  - "Свойства визуального элемента" – конфигурация визуального элемента;
  - "Редактировать визуальный элемент" – визуальное редактирование элемента;
  - "Найти использования визуального элемента" - в нижней части экрана отображает окно с результатом поиска выделенного визуального элемента во всех библиотеках;
  - "Перейти к родительскому визуальному элементу" – переход к элементу, на основании которого он сделан.
- *"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 изменения визуального элемента" – очистка визуального элемента с наследованием изменённых свойств или установкой их по умолчанию;
- "Свойства визуального элемента" – конфигурация визуального элемента;
- "Редактировать визуальный элемент" – визуальное редактирование элемента;
- "Найти использования визуального элемента" - в нижней части экрана отображает окно с результатом поиска выделенного визуального элемента во всех библиотеках;
- "Перейти к родительскому визуальному элементу" – переход к элементу, на основании которого он сделан;
- "Вид" – управление расположением визуальных элементов на ивдеокадрах:
  - "Виджет вверх" – поднятие виджета на самый верх;
  - "Виджет вниз" – опускание виджета на самый низ;
  - "Поднять виджет" – поднять виджет выше;
  - "Опустить виджет" – опустить виджет ниже;
  - "Выравнять слева" – выравнивание виджета слева;
  - "Выравнять по центру вертикально" – выравнивание виджета вертикально по центру;
  - "Выравнять справа" – выравнивание виджета справа;
  - "Выравнять сверху" – выравнивание виджета сверху;
  - "Выравнять по центру горизонтально" – выравнивание виджета горизонтально по центру;
  - "Выравнять снизу" – выравнивание виджета снизу;
- "Библиотека: {Имя библиотеки}" – пункты меню для доступа к видеокадрам/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;

    - "Следующее" – активировать следующее окно;
    - "Предыдущее" – активировать предыдущее окно;
- *"Вид"* – управление отображением рабочего окна и панелей на нём:
    - "Весь экран" – отображение на весь экран;
    - "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");
    - "Результаты поиска"- окно отображения результата поиска визуальных элементов в библиотеках;
    - "Проекты" – закрепленное окно управления деревом проектов;
    - "Виджет" – закрепленное окно управления деревом библиотек виджетов;
    - "Атрибуты" – закрепленное окно диспетчера атрибутов;
    - "Связи" – закрепленное окно диспетчера связей;
    - "Библиотека: {Имя библиотеки}" – управление видимостью панелей библиотек виджетов;
- *"Помощь"* – помощь по СКАДа и модулю Vision:
    - "Справка" – содержит справочную информацию о работе в программной платформе;
    - "Про QT" – информация о библиотеке QT, используемой модулем;
    - "Что это" – запрос описания элементов интерфейса окна;
- *"QT Starter"* – запуск UI модулей СКАДА системы*:*
    - системный конфигуратор (QT);
    - рабочий пользовательский интерфейс (QT).

Внизу окна разработки СВУ располагается строка статуса, в которой размещены индикаторы визуального масштаба редактируемого кадра, текущей раскладки клавиатуры, 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 сменить пользователя, 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: редактирование свойств контейнеров визуальных элементов (библиотек виджетов и проектов) и свойств самих визуальных элементов. Диалоги вызываются из

контекстного меню визуального элемента. Изменения, внесённые в диалогах, сразу же попадают в движок СВУ.

На рисунке 122 представлено окно свойств контейнера визуальных элементов, при вызове его для проекта становятся доступными вкладки «Стили», «Журнал сообщений», «Строка состояния» и «Диагностика».

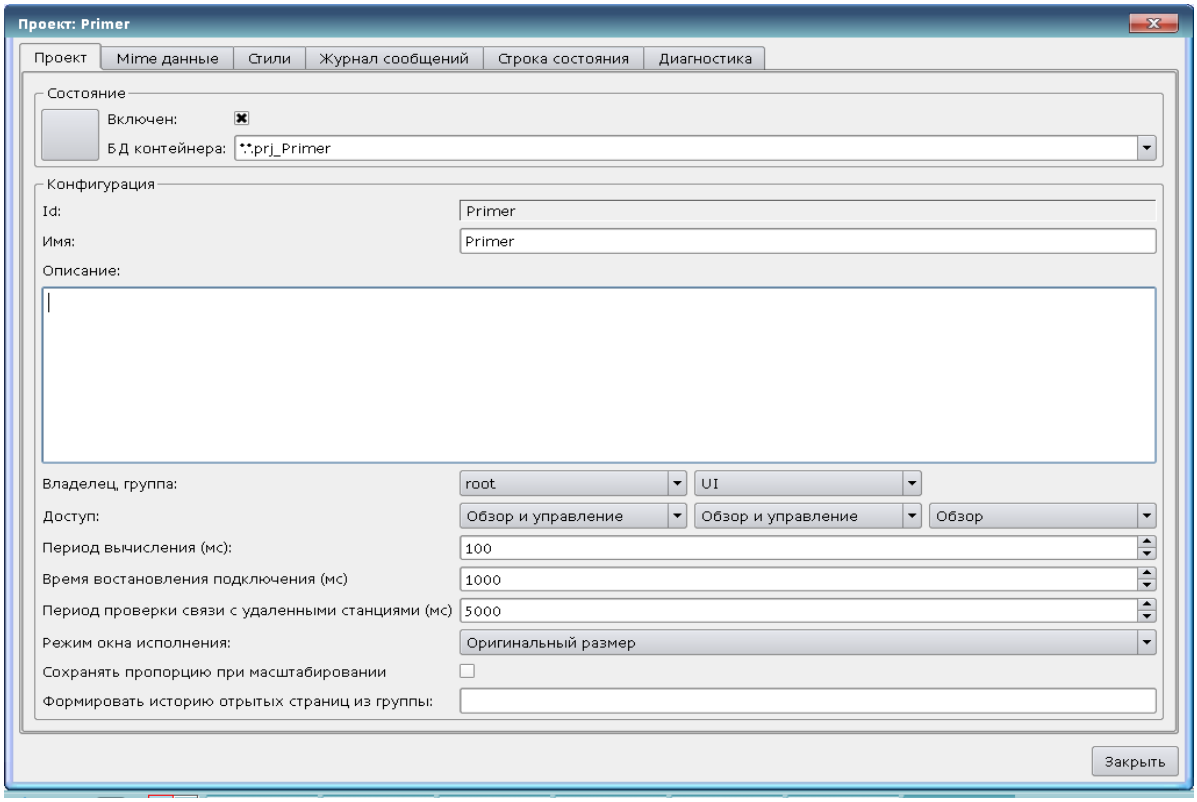


Рисунок 122

С помощью главной вкладки этого окна можно установить:

- состояние контейнера элементов, а именно: "Включен" и БД контейнера;
- идентификатор, имя и описание контейнера;
- пользователя, группу пользователей и доступ пользователей;
- для проекта: период вычисления проекта, время восстановления подключения, период проверки связи с удаленными станциями, режим открытия окна при исполнении, формирование истории открытых страниц из группы и флаг сохранения пропорций при масштабировании.

На вкладке «Строка состояния» можно настроить перечень отображаемых в строке состояния элементов. На вкдладке «Диагностика» отображаются текущие сообщения системы.

С помощью вкладки «Стили» 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.

Окно редактирования свойств визуального элемента виджет представлено на рисунке 123 и содержит следующие вкладки: «Виджет», «Атрибуты», «Обработка» и «Связи».

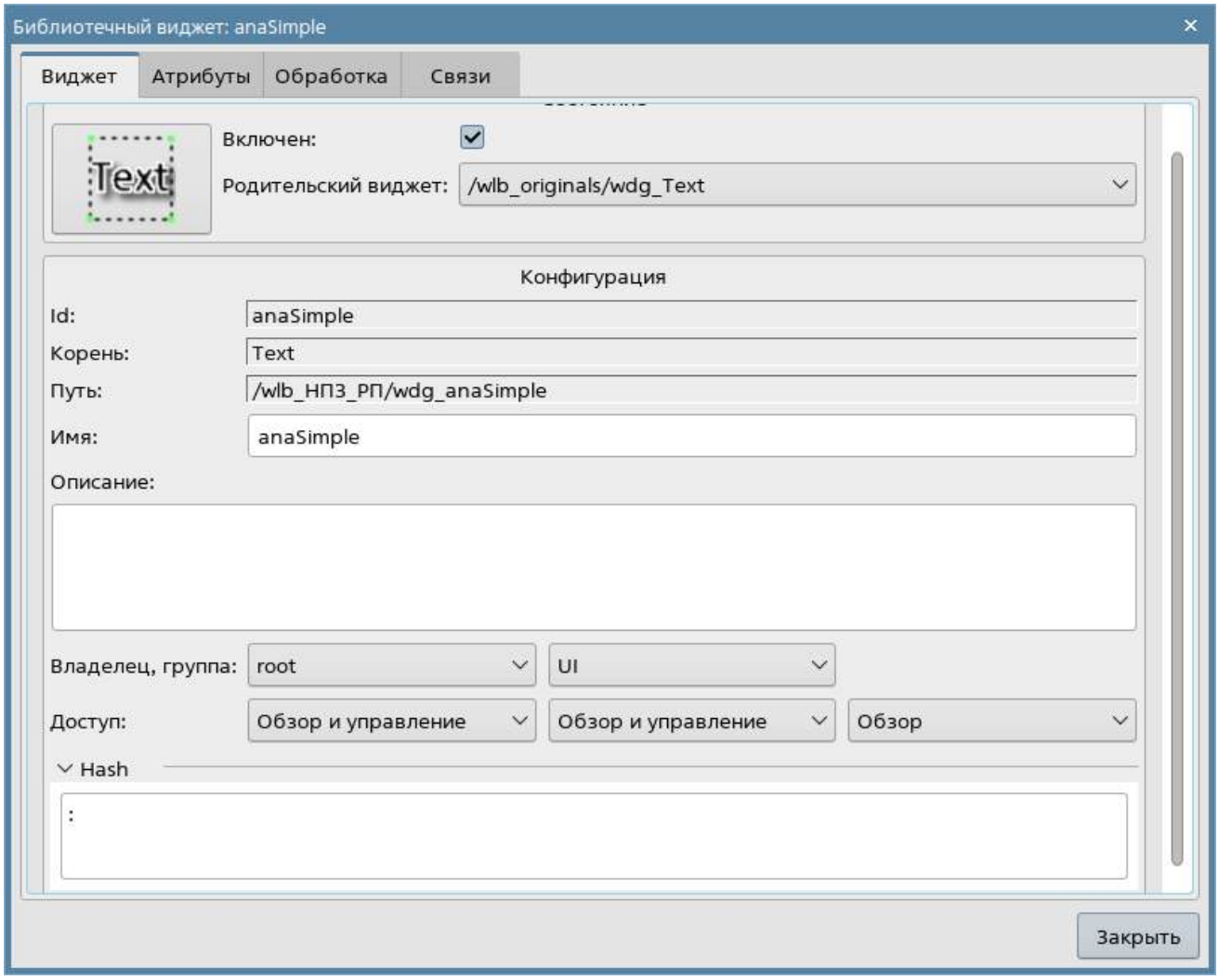


Рисунок 123

На вкладке «Виджет» можно установить:

- состояние элемента, а именно: "Включен" и родительский виджет;
- конфигурацию элемента: идентификатор, корень, путь, имя и описание элемента;
- пользователя, группу пользователей элемента и доступ пользователей.

Вкладка «Атрибуты» содержит набор стандартных и индивидуальных атрибутов элемента и позволяет провести их редактирование (совпадает с содержанием закрепленного окна «Атрибуты»).

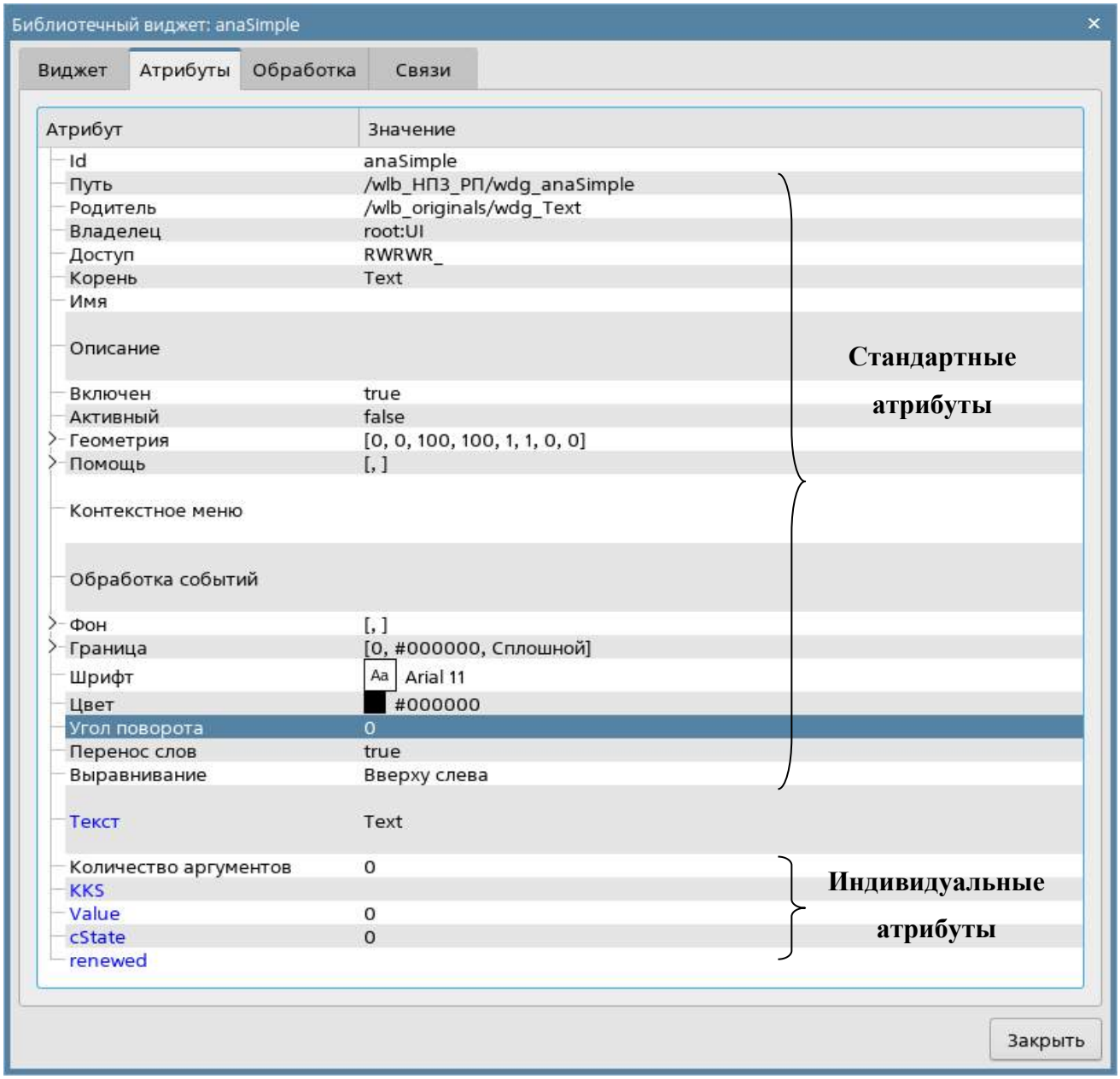


Рисунок 124

Стандартными атрибутами любого виджета являются:

"ID" (id) – идентификатор виджета;

"Путь" (path) – путь к виджету;

"Родитель"(parent) – путь к родительскому виджету;

"Владелец"(owner) – владелец и группа виджета в форме “[владелец]:[группа]” (по умолчанию “root:UI”);

"Доступ "(perm) – права доступа к виджету в форме “[польз.][группа][другие]”:

- _ – нет доступа;
- R_ – только чтение;
- RW – чтение и запись.

По умолчанию RWRWR_.

"Корень"(root) – идентификатор примитива, лежащего в основе виджета;

"Имя"(name) – название виджета;

"Описание" (dscr) – текстовое описание виджета;

"Включен"(en) – состояние виджета включен, отключенный виджет не отображается при исполнении;

"Активный"(activ) – состояние виджета. Активные элементы могут получать фокус при исполнении, а значит получать клавиатурные и иные события с последующей их обработкой;

"Геометрия"(geom) – размеры виджета;

"Помощь"(tipTool,tipStatus) – подсказка оператору и состояние виджета;

"Контекстное меню"(contextMenu) – контекстное меню в формате списка строк: “[ItName]:[Signal]”, где ItName – имя элемента; Signal – имя сигнала, результирующее имя сигнала: “usr_[Signal]”;

"Обработка событий"(evProc) – прямая обработка событий для управления страницами в формате: “[event]:[evSrc]:[com]:[prm]”, где:

- event – ожидаемое событие;
- evScr – источник события;
- com – команда сеанса:
    - open – открытие страницы. Открываемая страница указывается в параметре <prm> как напрямую, 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/*/*/$);
    - prev – открытие предыдущей страницы. Открываемая страница указывается в параметре <prm> в виде шаблона (например: /pg_so/*/*/$);
- prm – параметр команды, где используются:
    - pg_so – прямое имя желаемой 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;
    - * – имя страницы берется из имени предыдущей страницы или подставляется первая доступная страница, если предыдущая открытая страница отсутствует;

- $ – указывает на место, относительно которого открывается страница.

"Фон" – установка цвета виджета или изображения

"Граница" – оформление границы виджета (ширина линий, цвет, стиль)

Вкладка «Обработка виджета» обеспечивает описание конфигурации процесса формирования динамических связей и конфигурации динамики. Вкладка содержит таблицу конфигурации свойств атрибутов виджета и поле текста программы, для описания процедуры обработки виджета (рисунок 125).

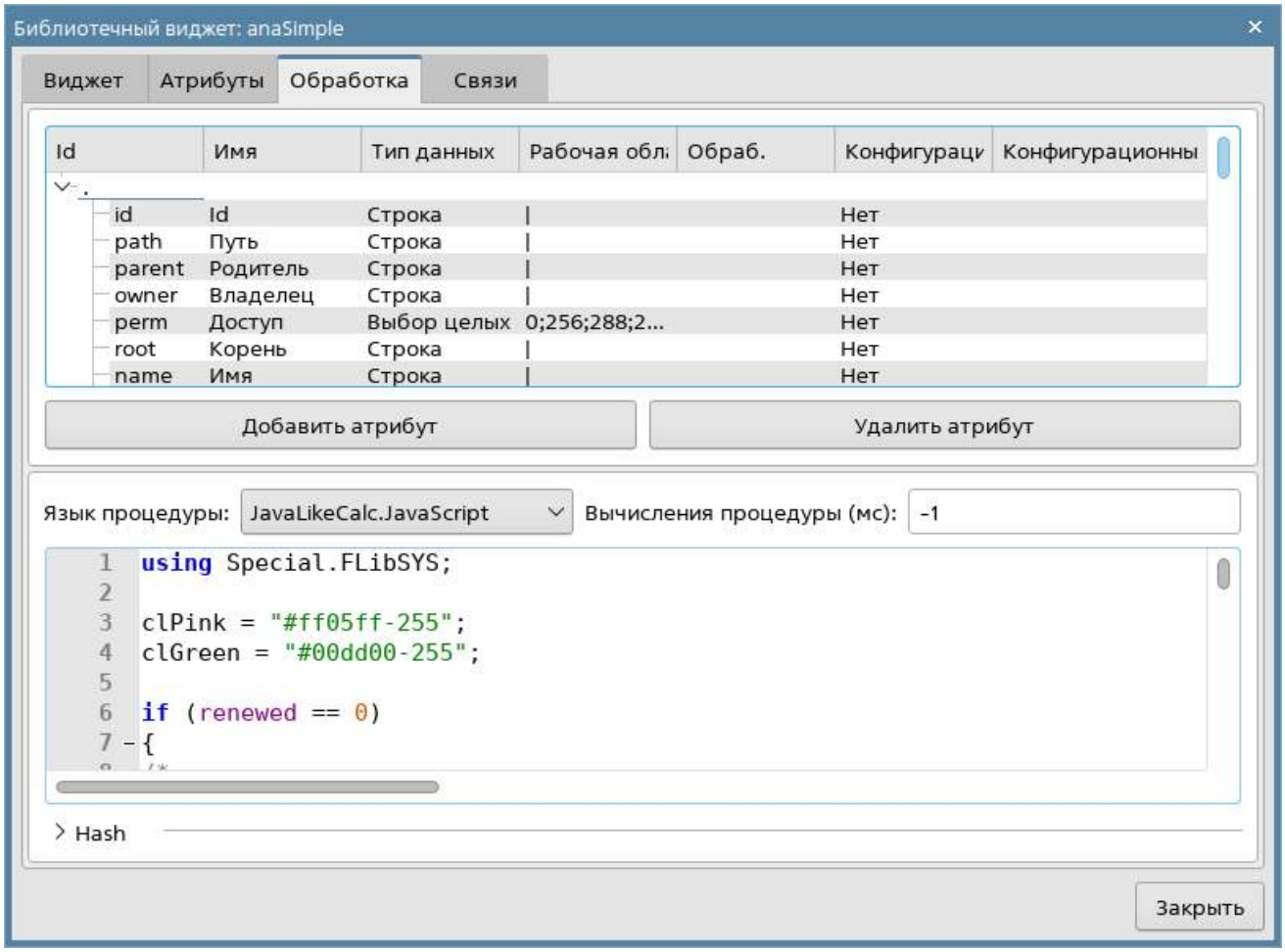


Рисунок 125

В данном окне возможно добавление, редактирование и удаление атрибутов видеокадра. Для каждого пользовательского атрибута можно изменить следующие параметры:

ID – идентификатор атрибута;

Имя – описание атрибута;

Тип данных – может принимать значения: логический, целый, вещественный, строка, объект, выбор целых, выбор вещественных, выбор строк, текст, цвет, изображение, шрифт, ДатаВремя, адрес;

Рабочая область – позволяет задать диапазон значений;

Обработка – установка в значение «True» позволяет использовать атрибут в вычислительной процедуре виджета;

Конфигурация – позволяет указать тип связи для атрибута и принимает значение: постоянная, входная связь, выходная связь, полная связь, из стиля, тревога;

Конфигурационный шаблон – позволяет описать группы динамических атрибутов (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.

Например, для настройки стиля для атрибута цвет необходимо в поле «Конфигурация» этого атрибута выбрать «Из стиля», а в «Конфигурационном шаблоне» указать идентификатор стиля.

При выборе в поле «Конфигурация» для атрибута значения типа связи – постоянная, входная связь (связь с динамикой только для чтения), выходная связь (связь с динамикой только для записи) или полная связь (чтение и запись) – на вкладке «Связи» становится доступной настройка связи с динамикой (рисунок 126). При этом тип связи может принимать значение:

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". Знак "(+)", в конце адреса, сигнализирует об успешной линковке и присутствии целевого объекта.

wdg: – связь на атрибут другого виджета или виджет в целом, для группы атрибутов.

Настройка связи осуществляется последовательным выбором значения для выбранного атрибута.

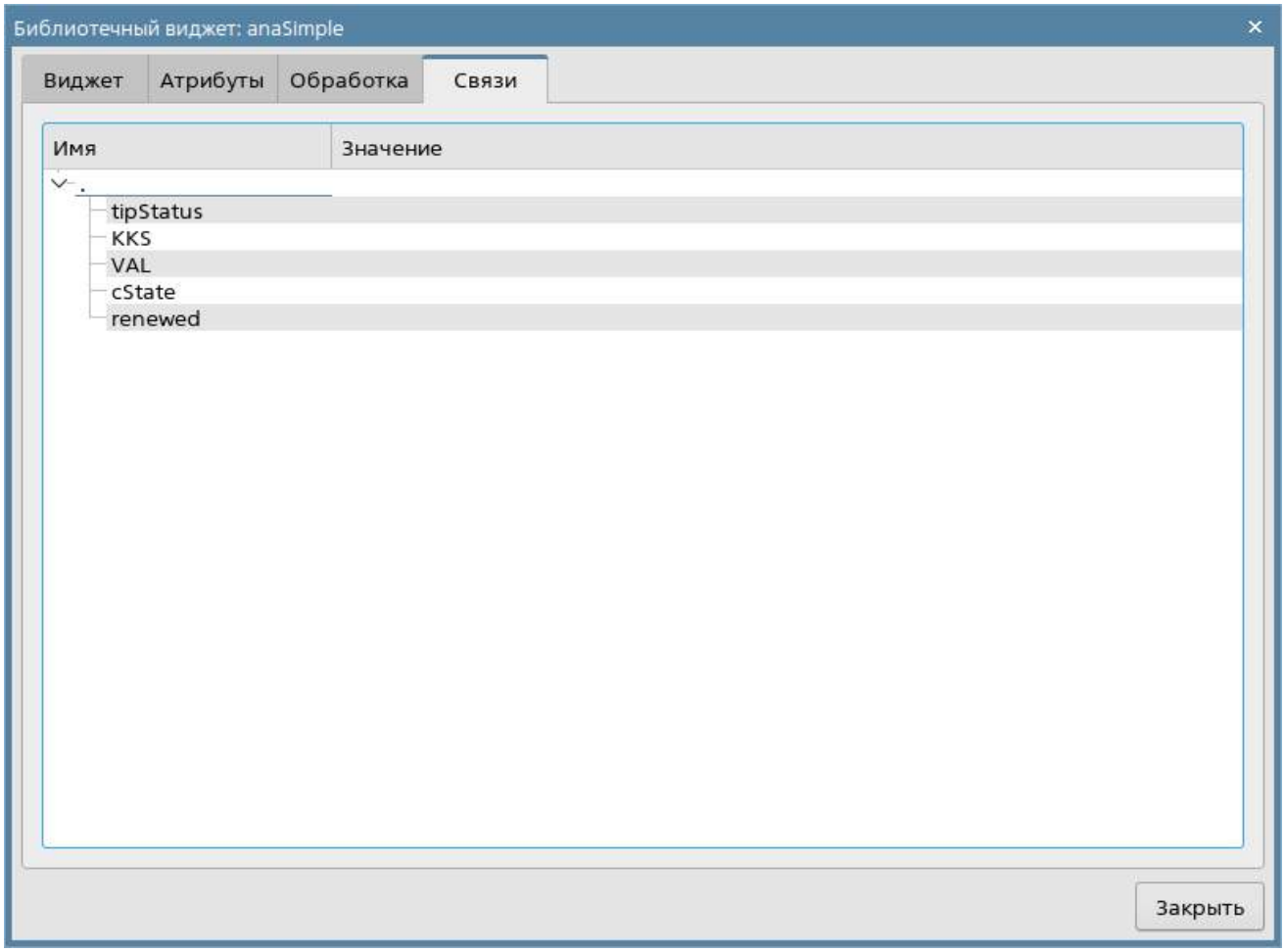


Рисунок 126

Обработка связей происходит с периодичностью вычисления видеокадра в следующем порядке:

- получение данных входных связей;
- выполнение вычисления скрипта;
- передача значений по выходным связям.

При размещении виджета, содержащего конфигурацию связей, в контейнер виджетов все связи исходного виджета добавляются в список результирующих связей контейнера виджетов.

Для создания кадров общего назначения с функцией предоставления детализированных данных разных источников одного типа предусмотрен механизм динамической установки связей 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.

8. СОБЫТИЯ, ИХ ОБРАБОТКА И КАРТЫ СОБЫТИЙ

В СКАДА предусмотрен механизм управления интерактивными пользовательскими событиями.

Событие – это сообщение, которое возникает в различных точках исполняемого кода при выполнении определенных условий. События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения (например, открытие окна при нажатии пользователем на кнопку).

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, сгенерировавшему событие.

В таблице 6 приведен перечень стандартных событий, поддерживаемых в СКАДА.

Таблица 6

ID	Описание
Клавиатурные события: key_[pres|rels][Ctrl|Alt|Shift]{Key}:	
*SC#3b	скан код клавиши
*#2cd5	код неименованной 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 - "->"
*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 - '['

ID	Описание
*BackSlash	обратная наклонная линия - '/'
*BracketRight	правая квадратная скобка - ']'
*QuoteLeft	левая кавычка - "'
События клавиатурного фокуса:	
ws_FocusIn	фокус получен виджетом
ws_FocusOut	фокус утерян виджетом
События от манипулятора мышь:	
key_mouse[Pres|Rels][Left|Right|Midle]	нажата/отпущена кнопка мыши
key_mouseDblClick	двойное нажатие левой кнопки мыши
ws_mouseEnter	наведение курсора на область виджета
ws_mouseLeave	покидание курсором области виджета
События квитирования на стороне среды визуализации:	
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[Left|Right|Midle|DblClick]	активация фигур (заливок) клавишей мыши
ws_Fig[n][Left|Right|Midle|DblClick]	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

ID	Описание
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
ws_MediaFinished	окончание проигрывания Медиа-потока

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 стандартном атрибуте виджета "evProc" и описан в разделе 7.

Механизм "Обработка событий с помощью вычислительной процедуры виджета" основан на атрибуте "event" и пользовательской процедуре вычисления на одном из языков пользовательского программирования СКАДА. 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 СКАДА:

```
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;
```

9. СИГНАЛИЗАЦИЯ

Важным элементом любого интерфейса визуализации является уведомление пользователя про нарушения – сигнализация. Для упрощения восприятия, а также в виду тесной связности визуализации и уведомления, интерфейс уведомления интегрирован в интерфейс визуализации. Во всех виджетах предусмотрены 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;

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.

9.1 Формирование сигнала и получение его средой визуализации

Формирование сигнала производится самим виджетом путём установки собственного атрибута "alarm" нужным образом, и в соответствии с ним устанавливается атрибут "alarmSt" текущего и вышестоящих виджетов. Среда визуализации получает 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.

9.2 Квитирование

Квитирование производится путём указания корня ветви виджетов и типов уведомления. Это позволяет реализовать квитирование на стороне среды визуализации 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;
```

10. 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 в форме: <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.

11. БИБЛИОТЕКИ ГРАФИЧЕСКИХ ЭЛЕМЕНТОВ

Пользовательский интерфейс СКАДА представляет собой набор графических страниц (мнемосхем). Каждая графическая страница располагается в рабочей области окна. Иерархия страниц определяет навигацию пользователя по проекту.

Иерархия страниц разрабатывается в виде многоуровневых мнемосхем, доступ к которым в любой момент времени зависит от прав текущего пользователя.

Мнемосхемы являются средствами визуального контроля за технологическим процессом и предоставляют доступ в режиме реального времени к используемому технологическому оборудованию, отображаемому на них, в части:

- текущего состояния;
- управления (при наличии соответствующих прав пользователя);
- измерения текущих параметров;
- достоверности сигналов;
- состояния интерфейсных каналов связи ПТС, позволяющие в режиме реального времени формировать управляющие воздействия для управления технологическим оборудованием.

Мнемосхемы могут быть объединены в группы. На рисунке 127 приведен пример вызова мнемосхем с помощью кнопок «Главная», «Резервуары 1-3», «Резервуары 4-6» и т.д.

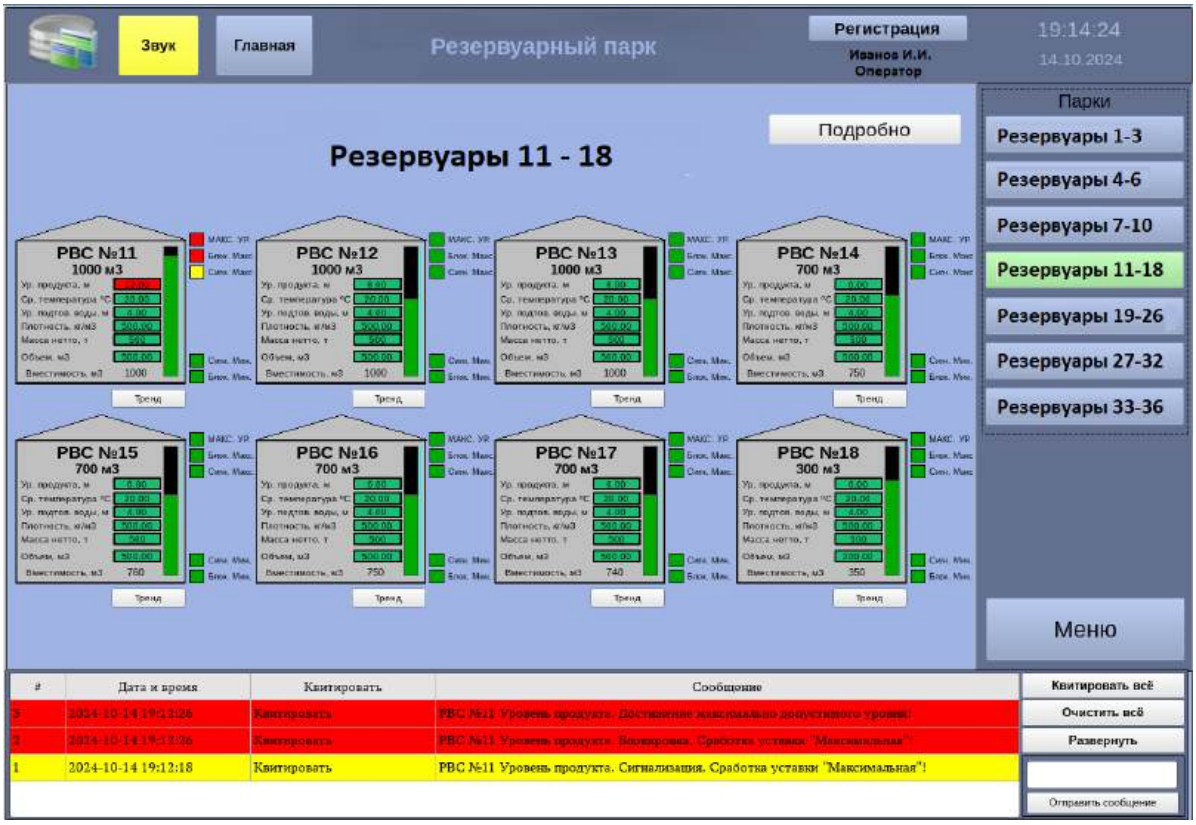


Рисунок 127

В рабочей области мнемосхем могут размещаться статические и динамические объекты.

Статические объекты представляют собой текстовые надписи, рисунки, линии и прочие фигуры, выполненные в однотонной цветовой гамме, либо состоящие из разноцветных элементов.

Динамические объекты представляют собой пиктограммы, отображающие состояние механизмов, процессов, аналоговых и дискретных параметров.

Операция выбора динамических элементов на мнемосхемах позволяет вызывать окна управления данными элементами, либо окна, которые предназначены для более детального отображения информации (графики).

Все виджеты элементов мнемосхем распределены по библиотекам, подключаемым по мере необходимости.

11.1 Библиотека «Базовые виджеты» (originals)

Библиотека «Базовые виджеты» (originals) содержит образы базовых элементов (примитивов) используемых для разработки проектов.

Таблица 7

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., которые можно изменять при необходимости. Для редактирования элементарных фигур используется специальный графический редактор, описанный в подразделе 11.2
FormEl	Элементы формы	Включает поддержку стандартных компонентов формы: редактирование строки; редактирование текста; флажок; кнопка; поле выбора из списка; список;

Id	Наименование	Функция
		таблица; дерево; слайдер; строка прокрутки
Text	Текстовые поля	Элемент текста (метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием
Media	Медиа	Элемент отображения растровых и векторных изображений различных форматов, проигрывания анимированных изображений, проигрывания аудио фрагментов, просмотра видеофрагментов и атрибутов СКАДА-системы типа массив
Diagram	Диаграмма	Элемент построения графиков и диаграмм с поддержкой возможности отображения нескольких потоков трендов и различных режимов отображения
Protocol	Протокол	Элемент визуализации данных архива сообщений путём формирования протоколов с различными способами визуализации, начиная от статического сканирующего просмотра и заканчивая динамическим отслеживанием протокола сообщения
Document	Документ	Элемент формирования отчётов, журналов и другой документации на основе указанных данных
Box	Группа элементов (контейнер)	Содержит механизм размещения других виджетов с целью формирования новых, более сложных виджетов и страниц конечной визуализации
Surface	Поверхность	Виджет, позволяющий строить трехмерные поверхности различных типов по заданным пользователем или архивным данным

Более детально рассмотрим индивидуальные атрибуты каждого примитива.

11.1.1 Примитив элементарная фигура (ElFigure)

Реализована поддержка элементарных фигур: линия, эллиптическая дуга, кривая 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.

– 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.

– Индивидуальными атрибутами данного примитива являются:

Линия:ширина (lineWdth) – ширина линии;

Линия:цвет (lineClr) – имя цвета в виде "color[-alpha]", где:

- "*color*" – стандартное имя цвета или числовое представление из трёх шестнадцатеричных чисел цвета "#RRGGBB";
- "*alpha*" – уровень альфа-канала (0-255).

Примеры:

– "red" – сплошной красный цвет;

– "#FF0000" – сплошной красный цвет в цифровом коде;

– "red-127" – полупрозрачный красный цвет;

Линия:стиль (lineStyle) – стиль линии (сплошная, пунктир, точечная);

Граница:ширина (bordWdth) – ширина бордюра линии. Нулевая ширина указывает на отсутствие бордюра;

Граница:цвет (bordClr) – цвет бордюра;

Заполнение:цвет (fillColor) – цвет заливки;

Заполнение:изображение (fillImg) – имя изображения в форме "[src:]name", где: "*src*" – источник изображения:

- file – прямо из локального файла по пути;
- res – из таблицы mime ресурсов БД.

– "*name*" – путь файла или идентификатор mime-ресурса.

Примеры:

– "res:backLogo" и "backLogo" – из таблицы mime ресурсов БД для идентификатора "backLogo";

– "file:/var/tmp/backLogo.png" – из локального файла по пути "/var/tmp/backLogo.png".

Угол поворота (orient) - угол поворота содержимого виджета;

Список элементов (elLst) - список графических примитивов в формате:

Линия:
line:(x|y)|{1}:(x|y)|{2}:[width|w{n}]:[color|c{n}]: [bord_w|w{n}]:[bord_clr|c{n}]:[line_stl|s{n}]

Дуга: arc:(x|y)|{1}:(x|y)|{2}:(x|y)|{3}:(x|y)|{4}:(x|y)|{5}:[width|
w{n}]:[color|c{n}]:[bord_w|w{n}]:[bord_clr|c{n}]:[line_stl|s{n}]

Кривая Безье: bezier:(x|y)|{1}:(x|y)|{2}:(x|y)|{3}:(x|y)|{4}:[width|w{n}]:
[color|c{n}]:[bord_w|w{n}]:[bord_clr|c{n}]:[line_stl|s{n}]

Заливка:
fill:(x|y)|{1},(x|y){2},...,(x|y)|{n}:[fill_clr|c{n}]:
fill_img|i{n}]

Где:

- (x|y) – прямая точка (x,y) координаты в пикселах с плавающей точкой;
- {1}...{n} – динамические точки 1...n;
- width, bord_w – прямая ширина линии и бордюра в пикселах с плавающей точкой;
- w{n} – динамическая ширина 'n';
- color, bord_clr, fill_clr – прямой цвет линии, бордюра и заполнения в виде имени или 32-битного кода с альфа: {имя}-AAA, #RRGGBB-AAA;
- c{n} – динамический цвет 'n';
- line_stl – прямой стиль линии: 0-сплошная, 1-пунктирная, 2-точечная;
- s{n} – динамический стиль 'n';
- fill_img – прямое изображение заполнения в формате "[src%3Aname]", где:
  - "src" – источник изображения:
    - file – непосредственно из локального файла по пути;
    - res – из таблицы mime-ресурсов БД.
  - "name" – путь файла или идентификатор mime-ресурса;
- i{n} – динамическое изображение заполнения 'n'.

На рисунке 125 представлен пример видеокадра, содержащий элементарные фигуры с атрибутами этих фигур.

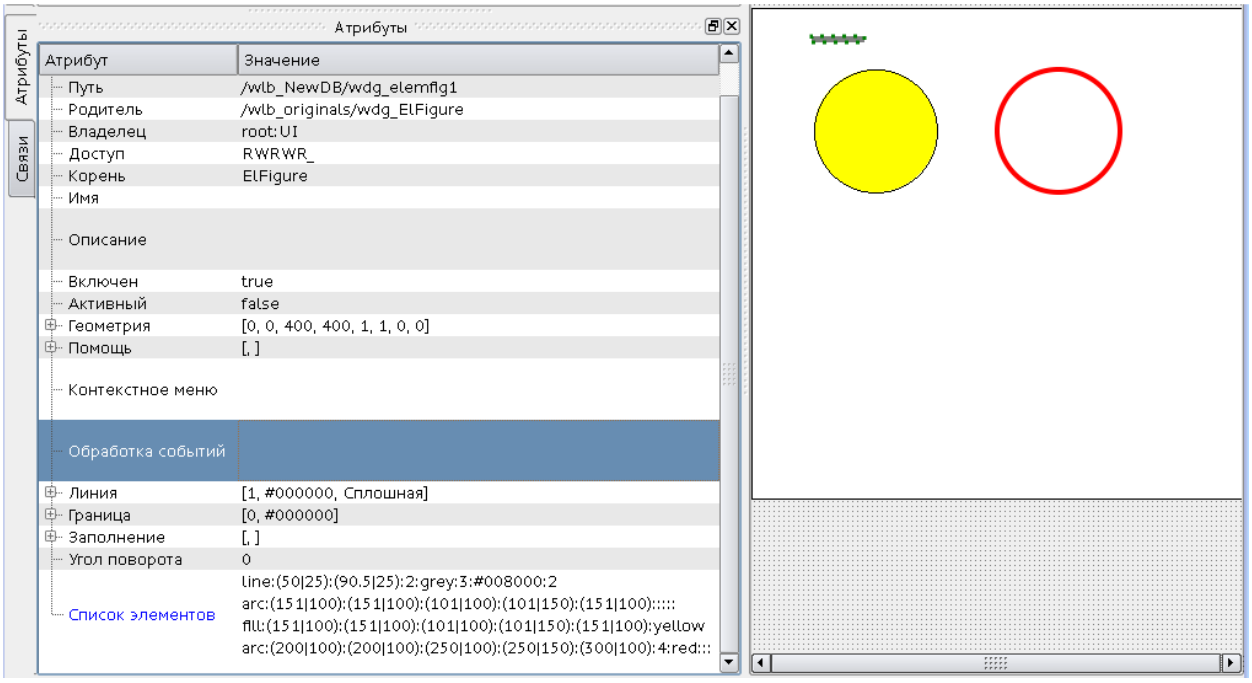


Рисунок 128

11.1.2 Примитив текста (Text)

Индивидуальными атрибутами примитива текст являются:

- *шрифт (front)* – имя шрифта в виде: "{family}{size}{bold}{italic}{underline}{strike}", где:
  - "family" – семейство шрифта, для пробелов используйте символ '_', вроде: "Arial", "Courier", "Times_New_Roman";
  - "size" – размер шрифта в пикселах;
  - "bold" – усиление шрифта (0 или 1);
  - "italic" – наклонность шрифта (0 или 1);
  - "underline" – подчёркивание шрифта (0 или 1);
  - "strike" – перечёркивание шрифта (0 или 1);
- *цвет текста (color)*;
- *ориентация текста (orient)*, поворот на угол;
- *автоматический перенос по словам (wordWrap)*;
- *выравнивание текста по горизонтали и вертикали (aligment)*;
- *отображение фона в виде цвета и/или изображения (backColor, backImg)*;
- *отображение бордюра* вокруг текста, с указанным цветом, шириной и стилем (bordWight, boardColor, boardStyle);
- *формирование текста из аргументов различного типа и свойств*. Аргумент может быть трех типов: integer, real, string. Конфигурация

аргумента: целое – [len] - ширина значения; вещественное – [wdth];[form];[prec] - ширина значения, форма значения(‘g’, ‘e’, ‘f’); строка-[len] - ширина строки.

На рисунке 129 представлен видеокадр, содержащий примеры текста с использованием различных параметров.

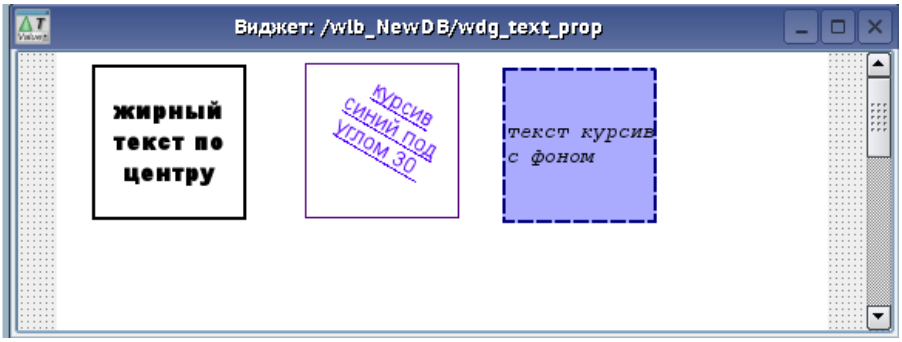


Рисунок 129

11.1.3 Примитив элементы формы (FormEl)

Реализованы следующие типы элементов формы:

- *Строка редактирования* - представлено следующими видами: «Текст», «Combo», «Целое», «Вещественное», «Время», «Дата», «Время и Дата»;
- *Текстовый редактор* - представляет редактор плоского текста с подтверждением или отказом от ввода;
- *Check Box* - предоставляет поле выбора бинарного флажка;
- *Кнопка* - предоставляет кнопку с поддержкой: цвета кнопки, изображения в кнопке и режима фиксации;
- *Combo Box (выбор из списка)* - предоставляет поле выбора элемента, со списка указанных элементов;
- *Список* - предоставляет поле списка с контролем за текущим элементом;
- *Дерево* - элемент иерархической структуры;
- *Таблица* - предоставляет редактор создания таблиц;
- *Слайдер* - элемент слайдера;
- *Полоса прокрутки*.

Для указанных элементов реализованы режимы: «Включен» и «Активен», а также передача изменений и событий в модель данных СВУ. Все элементы могут быть использованы для создания форм пользовательского ввода.

Индивидуальные атрибуты каждого элемента формы представлены в таблице 8.

Таблица 8

Название (ID)	Значение
Строка редактирования	
Значение (value)	Содержимое строки
Вид (view)	Вид строки редактирования: текст, комбобокс, целое, вещественное, время, дата, время и дата
Конфигурация (cfg)	Конфигурация строки. Формат значения данного поля для различных видов строки: *Текст* – ввод строки текста по шаблону с символьными элементами: A - необходим ASCII алфавитный символ: A-Z, a-z; a - разрешён, но не необходим ASCII алфавитный символ; N - необходим ASCII алфавитно-цифровой символ: A-Z, a-z, 0-9; n - разрешён, но не необходим ASCII алфавитно-цифровой символ; X - необходим любой символ; x - разрешён, но не необходим любой символ; 9 - ASCII цифра необходима: 0-9; 0 - ASCII цифра разрешена, но не необходима; D - ASCII цифра необходима: 1-9; d - ASCII цифра разрешна, но не необходима (1-9); # - ASCII цифра или знаки плюс/минус разрешёны, но не необходимы; H - необходим символ шестнадцатиричного числа: A-F, a-f, 0-9; h - разрешён, но не необходим символ шестнадцатиричного числа; B - необходим бинарный символ: 0-1; b - разрешён, но не необходим бинарный символ; **>** - все следующие алфавитные символы в верхнем регистре; **<** - все следующие алфавитные символы в нижнем регистре; **!** - выключение преобразования регистра; **** - используйте в разделителях для экранирования специальных символов, которые перечислены. *Комбобокс* - список значений редактируемого комбо-бокса по строкам; *Целое* - значение целого числа в форме: "[Минимум]:[Максимум]:[ШагИзменения]:[Префикс]:[Суффикс]"; *Вещественное* - значение вещественного числа в форме:

Название (ID)	Значение
	"[Минимум]:[Максимум]:[ШагИзменения]: [Префикс]:[Суффикс]:[ЗнаковПослеТочки]"; *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 название месяца ("Jan" ... "Dec"); MMMM - полное наименование 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
Подтверждать (confirm)	Включение режима подтверждения
Шрифт (front)	Имя шрифта в формате: "{family} {size} {bold} {italic} {underline} {strike}", где: - "*family*" - семейство шрифта, для пробелов используйте символ '_', вроде: "Arial", "Courier", "Times_New_Roman"; - "*size*" - размер шрифта в пикселах; - "*bold*" - усиление шрифта (0 или 1); - "*italic*" - наклонность шрифта (0 или 1); - "*underline*" - подчёркивание шрифта (0 или 1); - "*strike*" - перечёркивание шрифта (0 или 1) –
Текстовый редактор	

Название (ID)	Значение
Значение(value)	Содержимое редактора
Перенос слов (wordWrap)	Автоматический перенос текста по словам
Подтверждать (confirm)	Включение режима подтверждения
Шрифт (front)	Имя шрифта (формат описан выше)
Check Box	
Имя (name)	Имя/метка флага
Значение(value)	Значение флага
Шрифт (front)	Имя шрифта (формат описан выше)
Кнопка	
Имя (name)	Имя, надпись на кнопке
Значение(value)	Значение для фиксированной кнопки
Изображение (img)	Изображение на кнопке. Имя изображения в форме "[src:]name", где: "*src*" - источник изображения: - file - прямо из локального файла по пути; - res - из таблицы mime ресурсов БД. "*name*" - путь файла или идентификатор mime-ресурса. Примеры: "*res:backLogo*" или "*backLogo*" - из таблицы mime ресурсов БД для идентификатора "backLogo"; "*file:/var/tmp/backLogo.png*" - из локального файла по пути "/var/tmp/backLogo.png"
Цвет (color)	Цвет кнопки. Имя цвета в виде "**color[-alpha]**", где: "*color*" - стандартное имя цвета или числовое представление из трёх шестнадцатеричных чисел цвета "#RRGGBB"; "*alpha*" - уровень альфа-канала (0-255). Примеры: "*red*" - сплошной красный цвет;

Название (ID)	Значение
	"*#FF0000*" - сплошной красный цвет в цифровом коде; "*red-127*" - полупрозрачный красный цвет.
Цвет:текст (colorText)	Цвет текста
Переключатель(checkable)	Признак функционирования как фиксированная кнопка
Шрифт (front)	Имя шрифта (формат описан выше)
Список и выбор из списка	
Значение(value)	Выбранное значение списка
Элементы (items)	Перечень элементов списка в формате “[Текст]|[ID]|[Icon]”, Где "Текст" - элемент списка; "ID" – ID элемента списка; "Icon" - источник, может быть указан путь к файлу - "file" или "res" - из таблицы mime ресурсов БД
Шрифт (front)	Имя шрифта (формат описан выше)
Слайдер и полоса прокрутки	
Значение(value)	Положение слайдера
Конфигурация (cfg)	Конфигурация слайдера в формате: "[ВертОриент]:[Минимум]:[Максимум]:[ОдинШаг]:[СтрШаг]". Где: "*ВертОриент*" - признак вертикальной ориентации, по умолчанию ориентация горизонтальная; "*Минимум*" - минимальное значение; "*Максимум*" - максимальное значение; "*ОдинШаг*" - размер одного шага; "*СтрШаг*" - размер страничного шага.
Таблица	
Значение(value)	Текущее значение
Элементы	XML тег“tbl” для заполнения таблицы: <tbl> <h><s>{Заголовок1}</s><s>{Заголовок2}</s></h> <r><s>{Ряд1Колонка1Строка}</s>{Ряд1Колонка2Целое}</r>

Название (ID)	Значение
	<r>{Ряд2Колонка1Логическое}<r>{Ряд2Колонка2Вещественное}</r></r> </tbl> Теги: tbl – Таблица, свойства таблицы в целом: sel –режим выбора-выделения элементов таблицы: “row” –по строкам, “col”- по колонкам, “cell”- ячейками по умолчанию; KeyID –номер ключевой строки-колонки, для получения значения выбора; colsWdthFit – подстраивать размер колонок (размер которых не фиксирован)под заполнениевсей ширины таблицы; hHdrVis,vHdrVis – установка видимости горизонтального, вертикального заголовков; sortEn – включение прямой сортировки по колонкам. h- Строка заголовков. Возможные атрибуты тегов ячеек заголовка для колонки в целом: width – ширина колонки в пикселах или процентах(10%); edit – возможность редактирования(0 или 1) ячеек колонки, по умолчанию –нет(0); color – цвет колонки в целом, в виде имени цвета или его кода; colorText – цвет текста колонки в целом, в виде имени цвета или его кода; font – шрифт текста колонки в целом, в виде типовой строки SCADA; sort- сортировка по данной колонке [0-по убыванию; 1 –по возрастанию]; r – Ряд значений. Возможные атрибуты тегов ячеек ряда для ряда в целом: color – цвет ряда в целом, в виде имени цвета или его кода; colorText – цвет текста ряда в целом, в виде имени цвета или его кода; font – шрифт текста ряда в целом, в виде типовой строки SCADA; s, i, r, b – ячейки типов данных строка, целое, вещественное и логическое. Возможные атрибуты: color – цвет фона ячейки, в виде имени цвета или его кода;

Название (ID)	Значение
	colorText – цвет текста ячейки, в виде имени цвета или его кода; font – шрифт текста ячейки, в виде типовой строки SCADA; img – изображение ячейки в форме “[{src}:]{name}” edit – возможность редактирования (0 или 1) ячейки колонки, по умолчанию – нет (0)
Шрифт (front)	Имя шрифта (формат описан выше)
Дерево	
Значение(value)	Выбранное значение
Элементы	Список элементов в виде пути: «/кат/кат/элем» по строкам
Шрифт (front)	Имя шрифта (формат описан выше)

На рисунке 130 представлен видеокадр, содержащий элементы формы

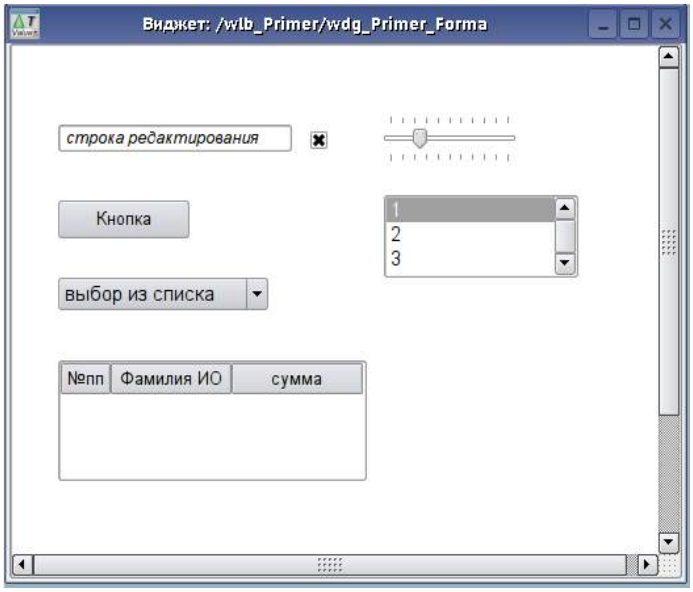


Рисунок 130

11.1.4 Примитив отображения медиа-материалов (Media)

Индивидуальными атрибутами данного примитива являются:

источник медиа данных (изображения или видео-материала) в формате “[src:]name”, где src - источник:

- file-прямо из файла,
- res - из таблицы mime ресурсов БД,
- stream - URL потока проигрывания видео или аудио,
- prm - из атрибута контроллера(тоько для типа медиа - массив);

- *тип медиа(type):* изображение, анимация, полное видео, массив
- *область карты (areas)* – количество активных областей;
- *заполнять виджет(fit)* – согласовать содержимое с размером виджета.

В зависимости от типа медиа-данных становятся доступными дополнительные атрибуты:

для анимации(Movie): скорость проигрывания в процентах от оригинальной скорости. Если значение меньше 1% - проигрывание прекращается;

для полноформатного видео (Full video):

- играть(play);
- Завор.проигр.(roll) – повторение проигрывания по завершению;
- пауза(pause);
- размер (size) – общий размер видео (в мс);
- положение (seek) – позиция проигрывания видео (в мс);
- громкость (volume) – громкость звука(0..100);

для массива:

- период слежения, с (trcPer) – интервал обновления виджета;
- пауза(pause);
- формат данных – формат отображения данных: нет, RGB, монохромный;
- архиватор значений(arch) – в формате «МодульАрхивов.IDАрхиватора»;
- промежуток времени;
- цвет минимального значения - считывается при незаданном формате данных;
- цвет максимального значения - считывается при незаданном формате данных;
- изображение: ширина и высота отображения.

На рисунке 131 представлена часть экрана с видеокадром, содержащим примеры просмотра/проигрывания медиа-данных.

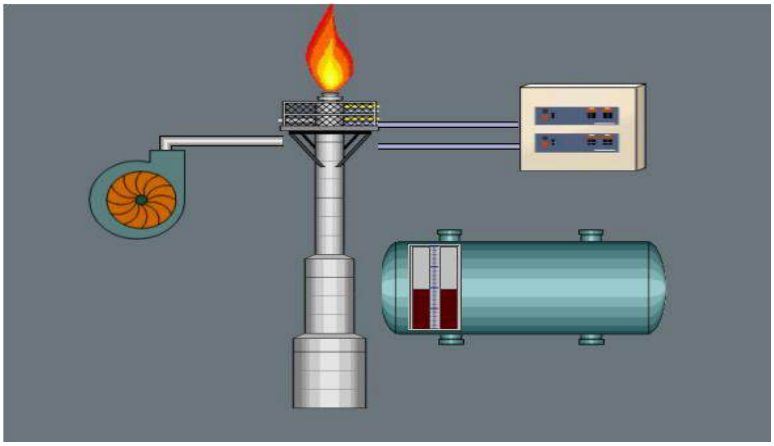


Рисунок 131

11.1.5 Примитив построения диаграмм/графиков (Diagram)

Содержит следующие типы диаграмм:

"График" – построение зависимости величины какого-то измеряемого параметра от времени;

"Массив" – отображение объектных типов данных. По шкале абсцисс откладывается целые значения – соответствующие номеру элемента в массиве, а по шкале ординат откладываются сами элемента массива. Таким образом, имея некий массив данных из 1500 элементов, данные будут распределены в диапазоне [0:1500];

"Зависимость" – строит график зависимости одного параметра от другого.

Для всех типов диаграмм в качестве источника данных возможно указание:

- параметра контроллера подсистемы "Сбор Данных";
- архива значений.

Индивидуальнными атрибутами виджета «Диаграмма» являются:

- *период слежения (trcPer)* – задает интервал обновления виджета. При выставлении значения в «0» трассировка прекращается, при этом запускается обновление текущей области отображения с дозагрузкой недостающих данных;
- *тип (type)* – график, массив, график зависимости;
- *время* – задает правую границу временной шкалы графика и состоит из двух вложенных атрибутов:
  - сек (указывается секундная составляющая, например 16.01.2023 11:21:20);
  - микросек (указывается микро секундная составляющая, например 146070);
- *размер* – устанавливает размер временной шкалы в секундах, так например, если мы установим атрибут «время» 16.01.2023 11:21:20, а атрибут «размер» установим в 60 секунд, то временная шкала (ось абсцисс – ось x) будет лежать в диапазоне [11:20:20; 11:21:20];
- *строгий диапазон* – использует данные только из указанного диапазона и в основном используется для графиков типа «Массив»;
- *курсор* – задает левую границу графика (точку отсчета), работает в связке с атрибутом статические оси и состоит из трех вложенных атрибутов:
  - сек (указывается секундная составляющая, например 16.01.2018 11:21:20);
  - микросек (указывается микро секундная составляющая, например 146070);

- цвет (указывается цвет точки отсчета).

– *архиватор значений* – позволяет указать, откуда брать данные для построения графика:
  - из буфера – значение <buffer>;
  - из архива - в поле ввести архиватор значений в форме: «Модуль архивов.IdАрхиватора» (например DBArch.postgresArhiv);
  - из архива, если в нем данные на единицу времени отсутствуют, брать из буфера (оставить поле пустым);

– *параметры* состоит из динамического числа атрибутов, один из которых является обязательным - «число графиков» и указывает на число поддерживаемых виджетом графиков (максимум 10). Остальные вложенные атрибуты формируются для каждого графика в зависимости от выставленного значения. Вложенными атрибутами являются:
  - адрес – указывает адрес к источнику данных в формате:
    - полный адрес к атрибуту параметра DAQ или Архива в форматах: Модуль/Сбор_данных/Контроллер/Параметр/Атрибут или /Архив/va_адрес к архиву,
    - прямая установка данных по префиксу data:<XMLNodeData>,
    - прямая установка данных по префиксу line:<значение>;
  - ширина линии;
  - цвет линии;
  - цвет заливки графика;
  - заливать график;
  - отображать график;
  - маркеры - ошибка(цвет), норма(цвет), отображать (ошибки, не отражать, все), тип предупредительной линии (нет, сплошная, пунктир, точечная, пунктир точка, пунктир точка точка, жирная);
  - значения;
  - свойства реального архива в виде “BegArh:EndArh:DataPeriod”, где “BegArh” – начало архива, “EndArh” – конец архива, “DataPeriod” – период данных архива в секундах, в реальном представлении вплоть до микросекунд(1e-6);
  - тип линии – нет, линия, ступенька справа, ступенька по центру, импульс, ступенька слева;

– *шкалы* – настройка следующих свойств шкал виджета:
  - цвет – задает цвет шкал;

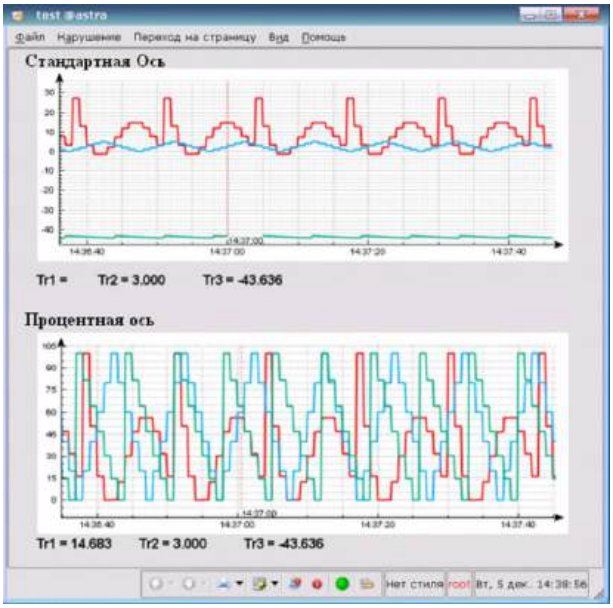
- статические оси – поддерживается только для типа Диаграммы «График», по умолчанию установлению значение «false» (описание в таблице 4);
- процентные оси *(sclPercent)* - позволяет установить флаг, переводящий оси в процентный масштаб (описание в таблице 4);
- маркеры – позволяет изменять отображение числовых значений на шкалах посредством вложенных атрибутов цвет и шрифт;
- X оси - позволяет настраивать шкалы абсцисс, посредством установки следующих вложенных атрибутов тип и диапазон. Тип шкалы абсцисс позволяет настраивать отображение осей - не рисовать, решетка, маркеры, решетка и маркеры. Диапазон рассмотрен в таблице 4;
- Y оси - состоит из динамического числа атрибутов, один из которых является обязательным «число осей» и указывает на число отображаемых осей ординат. Остальные вложенные атрибуты формируются в зависимости от установленного значения. Каждый из динамических атрибутов представлен вложенным набором параметров, позволяющими настроить каждую из осей:
  - тип – способ отображения осей: не рисовать, глобально, маркеры, решетка и маркеры, маркеры (лог), решетка и маркеры (лог);
  - расположение – положение маркеров внутри или вне области построения графиков;
  - автомасштабирование – при включении игнорируются значения атрибутов мин. значение и макс. значение;
  - мин. значение – устанавливает нижнюю границу оси ординат;
  - макс. значение – устанавливает верхнюю границу оси ординат;
  - множитель – кожффициент, на значение которого умножается каждое значение параметра графика;
  - прикрепленные графики;

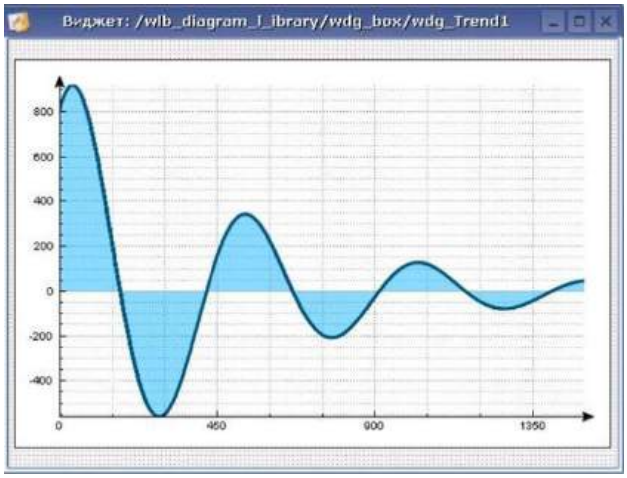
– *базовый масштаб* – установка атрибутов «время» и «размер» в формате «Time:значение;Size:значение;» (например: Time:0;Size:60;). В примере в качестве времени устанавливается текущее время, а размер отображаемой области равен 60 секунд. Установка этих значений позволяет указать параметры по умолчанию, к которым пользователь сможет вернуться в

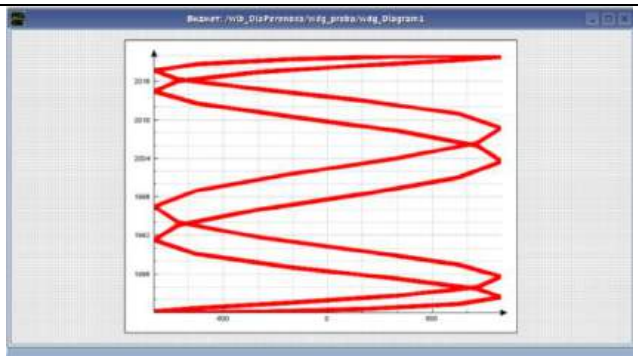
случае если изменял размеры графика (двигал, масштабировал и т.д.) в процессе работы.

В таблице 9 описаны индивидуальные атрибуты элемента диаграммы в зависисмости от выбранного типа.

Таблица 9

Тип диаграммы	*Индивидуальные атрибуты*
График 	Вложенный атрибут «диапазон» у атрибута X оси – позволяет выбирать диапазон отображения из трех возможных вариантов: – s – секунды; – ms – мили секунды; – mks – микро секунды. По умолчанию стоят s – секунды. Процентные оси - позволяет установить флаг, переводящий оси в процентный масштаб. В этом случае диапазон шкалы Y будет установлен от 0 до 100%, все графики будут вписаны в этот диапазон. Статические оси – при установке флага левая граница шкалы абсцисс устанавливается согласно значению атрибута «Базовое время», а график начинает строиться от этой точки к правой границе шкалы ординат, чье значение высчитывается посредством прибавления к значениям атрибута «Базовое время» значения атрибута «Размер». Разница между использованием этого атрибута и стандартной работой осей заключается в том, что в первом случае значения на оси абсцисс

Тип диаграммы	*Индивидуальные атрибуты*
	никуда не смещаются при превышении значения области отображения, тогда как во втором случае смещение значений идет справа на лево при превышении области отображения (атрибут «размер»)
Массивы 	Вложенные атрибуты X оси: - «мин. значение»; - «макс. значение»; - «множитель»; - «смещение» устанавливает смещение по оси абсцисс, так например, если взять массив данных размером в 1500 элементов, задать значение атрибута равным 900, то построится тот же график, однако его данные будут распределены в диапазоне [900; 2400]
График зависимости	Вложенный атрибут «адрес (гориз.)» в атрибуте Параметры; Вложенные атрибуты X оси: - «мин. Значение»;

Тип диаграммы	*Индивидуальные атрибуты*
	- «макс. Значение»; - «множитель».

11.1.6 Примитив формирования протокола (Protocol)

Реализована поддержка элемента формирования протокола со свойствами:

- формирование протокола из архива сообщений за указанное время с заданной глубиной;
- 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.

Индивидуальными атрибутами элемента являются:

- *заголовок видим (headVis)* – видимость заголовка таблицы;
- *время (time)* – текущее время;
- *размер (tSize)* – размер запроса, секунды. Установите значение в '0' для получения всех нарушений, для "lev" < 0;
- *период слежения, с (trcPer)* – режим и периодичность слежения;
- *архиватор (arch)* – архиватор сообщений в форме "МодульАрхивов.IdАрхиватора";
- *шаблон (tmpl)* - Шаблон категории или регулярное выражение "/{re}/". Для шаблона зарезервированы символы:
  - '*' – множество любых, группа символов;
  - '?' – любой, один символ;
  - '\\' – используйте для экранирования специальных символов.
- *уровень (lev)* – уровень сообщений. Для получения текущих нарушений значение дожно быть < 0;
- *порядок отображения (viewOrd)* - порядок отображения: "По времени", "По уровню", "По категории", "По сообщению", "По времени (обратно)", "По уровню (обратно)", "По категории (обратно)", "По сообщению (обратно)";

- *показать колонки (col)* – Список видимых и порядок колонок, разделённый символом ';'. Поддерживаются колонки:
    - "pos" – номер строки;
    - "tm" – дата и время сообщения;
    - "utm" – микросекундная часть времени сообщения;
    - "lev" – уровень сообщения;
    - "cat" – категория сообщения;
    - "mess" – текст сообщения;
- *свойства элемента (ItProp)* – количество свойств элемента.

На рисунке 132 представлен пример сформированного протокола.

	Дата и время	мкс	Уровень	Категория	Сообщение
5	25.07.2018 14:56:34	132014	4	/WorkStation/sub_DAQ/mod_OPC_UA/cntr_test/	0x80050000:Ошибка подключения к Internet сокету: Операция выполняется в данный момент!
6	25.07.2018 14:56:40	951804	1	/WorkStation/UI/VCAEngine/1-3392e6a2-aed5-490d-b901-4e36c9ed00cc/	Включение сеанса.
7	25.07.2018 14:56:40	953414	1	/WorkStation/UI/VCAEngine/1-3392e6a2-aed5-490d-b901-4e36c9ed00cc/	Запуск сеанса.
8	25.07.2018 14:56:40	956627	1	/WorkStation/UI/VCAEngine/	Пользователь 'root' подключился к сессии '1-3392e6a2-aed5-490d-b901-4e36c9ed00cc'.
9	25.07.2018 14:56:40	991081	3	/WorkStation/sub_UI/mod_Vision/	Ошибка открытия: /dev/input/by-path/platform-pcspkr-event-spkr
10	25.07.2018 14:56:41	9218	0	/WorkStation/sub_UI/mod_Vision/	Масштаб корневой страницы [1.000000:1.000000].
11	25.07.2018 14:56:43	186255	1	/WorkStation/UI/VCAEngine/	Пользователь 'root' отключился от сессии '1-3392e6a2-aed5-490d-b901-4e36c9ed00cc'.
12	25.07.2018 14:56:43	186342	1	/WorkStation/UI/VCAEngine/1-3392e6a2-aed5-490d-b901-4e36c9ed00cc/	Останов сеанса.

Рисунок 132

11.1.7 Примитив формирования отчётной документации (Document)

Реализована поддержка элемента формирования отчётной документации со свойствами:

- гибкое формирование структуры документа на основе языка гипертекстовой разметки, что дает возможность форматирования документов;
- формирование документов по команде или по графику - 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 СКАДА в виде <?dp<procedure>?>. Результирующий документ формируется путём исполнения процедур и вставки их результата в документ.

Индивидуальными атрибутами виджета формирования отчетности являются:

- *CSS (style)* – правила CSS в строках, например, для изменения цвета документа необходимо набрать: "body {background-color:#818181;}";
- *шаблон(tmpl)* – шаблон документа в XHTML. Начинается с тега "body" и включает процедурные вставки:

```
<body docProcLang="JavaLikeCalc.JavaScript">
 <h1>Значение<?dp return wCod+1.314;?></h1>
</body>
```

- *документ (doc)* - финальный документ в XHTML. Начинается с тега "body";
- *шрифт (front)* – базовый шрифт текста документа;
- *время начала документа(bTime)*;
- *текущее время (time)* – время генерации документа, записать время для генерации документа от этой точки или нуль для его регенерации;
- *размер архива (n)* – глубина архива, при положительном значении становятся активными атрибуты архива:
  - архив:курсор:текущий (aCur) – позиция текущего документа в архиве. Запись значения <0 производит архивацию текущего документа;
  - архив:курсор:вид (vCur) – текущий визуализируемый документ архива. Запись значения -1 – выбор следующего документа для отоббражения, -2 – выбор предыдущего документадля отображения;
  - архив:текущий документ (aDoc) – текущий документ архива в XHTML. Начинается с тега “body”;
  - архив:размер (aSize) – реальный размер архива документа.

Динамика шаблона документа определяется вставками исполняемых инструкций вида <?dp {procedure}?>. В процедурах могут использоваться одноимённые атрибуты виджета и функции пользовательского интерфейса программирования. Кроме атрибутов виджета зарезервированы специальные атрибуты, теги и атрибуты тегов (таблица 10).

Таблица 10

Имя	**Назначение**
Атрибуты	
rez	Атрибут результата исполнения процедуры, содержимое которого помещается в дерево документа
lTime	Время последнего формирования. Если документ

Имя	Назначение
	формируется впервые, то <lTime> = <bTime>
rTime	Содержит время для перебираемых значений в секундах. Определяется внутри тегов с атрибутом "docRept"
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*). В шаблоне запроса может указываться регулярное выражение в виде /{re}/. 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

Имя	Назначение
	документа. Не устанавливается пользователем!
table.export="1"	Включение возможности экспорта содержимого указанной таблицы в CSV-файл и другие табличные форматы

На рисунке 133 представлен пример сформированного документа.

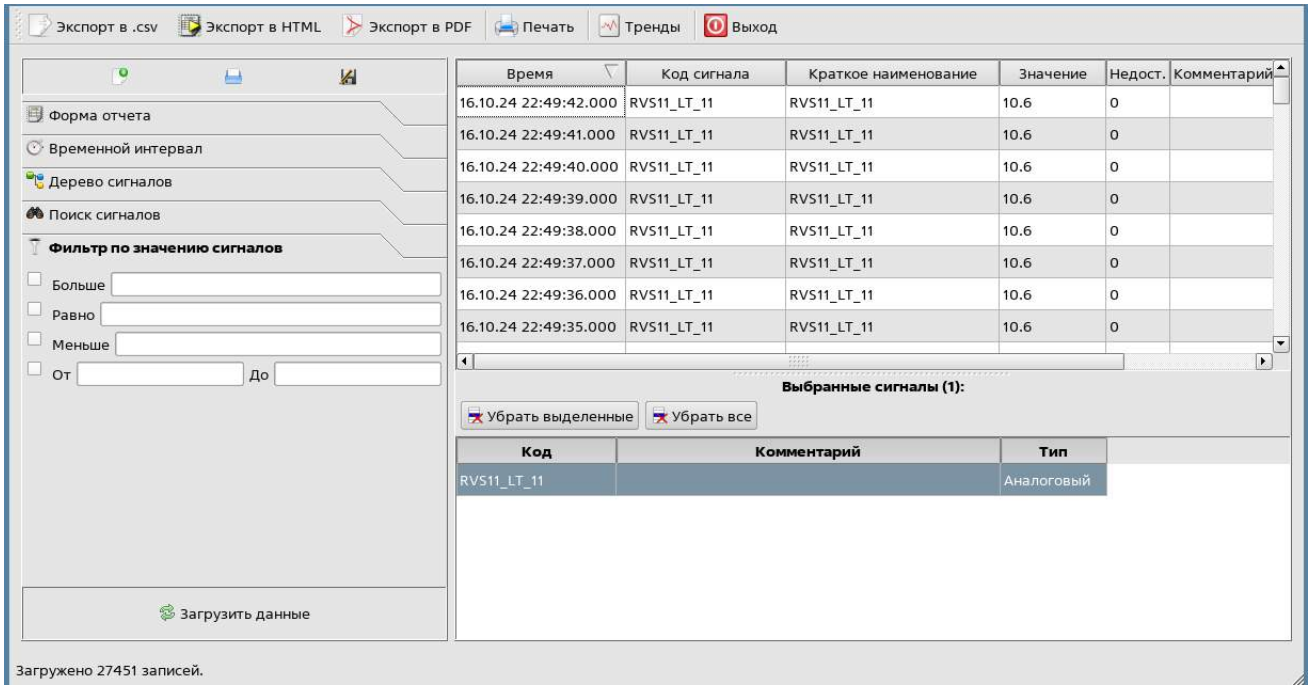


Рисунок 133

11.1.8 Примитив контейнера (Box)

Примитив контейнера используется для формирования составных виджетов и/или страниц пользовательского интерфейса. Данный примитив может включать в себя ссылки на видеокадры из библиотеки, формируя тем самым пользовательские элементы нужной конфигурации. Индивидуальные атрибуты данного примитива:

- *контейнер (box)* – позволяет формировать нужные объекты путём группировки базовых в рамках данного примитива;
- *страница (pgOpenSrc)* – элементы, построенные на данном примитиве, могут выполнять роль страницы пользовательского интерфейса. Полный адрес страницы, которая включена внутрь данного контейнера;
- *контейнер страниц* (pgGrp) – свойство замещения собственного содержимого другой страницей, в процессе исполнения. Используется для формирования фреймов на страницах пользовательского интерфейса;
- *фон (backColor, backImg)* – фон в виде цвета или изображения;

– *бордюр (bordWidth,bordColor,bordStyle)* – поддерживает возможность отображения бордюра с указанным цветом, толщиной и стилем.

–

11.1.9 Примитив поверхность (Surface)

Реализована возможность построения трехмерной поверхности по заданным пользователем или архивным данным.

Индивидуальными атрибутами являются:

- *источник* – исходные данные в формате:
  - prm:Модуль/контроллер/параметр/атрибут
  - Например: prm:TANGO/tango_test/surf/FourtyEight). Данные, обра-батываемые виджетом, должны быть представлены в виде объекта типа Matrix или Array;
- *период слежения* - позволяет задавать интервал обновления виджета;
- *пауза* – позволяет зафиксировать изображение поверхности в момент времени выставления значения этого атрибута в единицу. Так же, если значение данного атрибута установлено в единицу или true, то виджет не будет обновлять форму поверхности;
- *архиватор значений* – позволяет отображать поверхность по данным, хранящимся в архиве в формате «Модуль Архивов.Id архиватора». Данный атрибут работает совместно с атрибутом «Промежуток времени» (чтобы поверхность строилась по архивируемым данным, необходимо указывать оба атрибута);
- *промежуток времени* – позволяет указать промежуток времени архивируемых данных, по которым будет строиться поверхность. Значение этого атрибута устанавливаются согласно следующему формату:
  «начало(sec) начало(usec) конец(sec) конец(usec)».
  В случае если в указанном интервале архивируемые данные различны, то будет построена лишь та поверхность, которая соответствует архивируемым данным с меткой времени наиболее близкой к конечной. Данный атрибут работает совместно с атрибутом «Архиватор значений»;
- *поверхность* – предназначен для изменения визуальных стилей поверхности и цвета рендерной сетки, в случае наличия последней. Атрибут стиль может принимать следующие значения:
  - не рисовать;

- каркас – поверхность представлена в виде прозрачной сетки, позволяющей видеть все точки поверхности одновременно (рисунок 134а);
- непрозрачная сетка – поверхность представлена в виде непрозрачной сетки, отображающей только точки поверхности образующие ее передний план (рисунок 134б);
- Цветная – так называемая радужная раскраска, цвета которой распределяются в вертикальном направлении снизу вверх по цветам радуги – от фиолетового до красного (рисунок 134в);
- Цветная + сетка – закрашенная поверхность с накладываемой на неё рендерной сеткой (рисунок 134г).

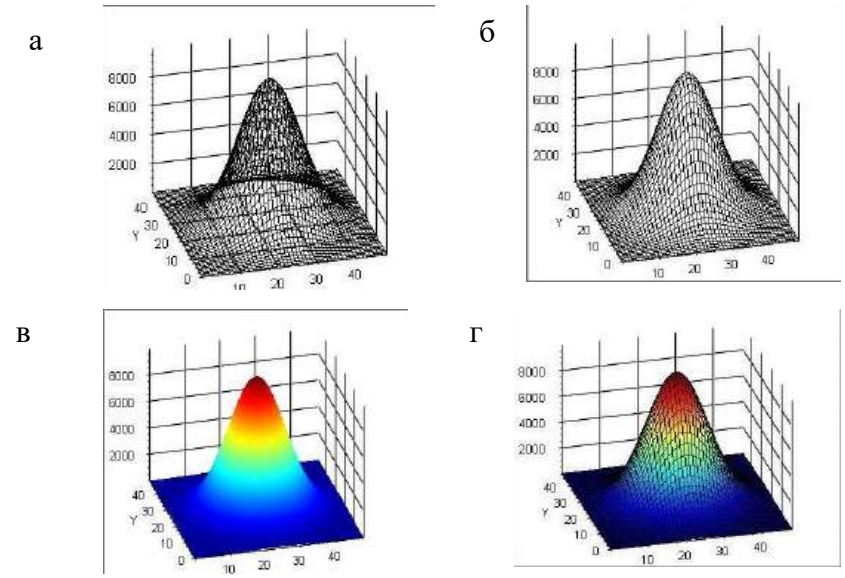


Рисунок 134

- *Шкалы* – предназначен для изменения параметров каждой из трех осей поверхности и представлен вложенным набором атрибутов, позволяющими задать стили отображения осей (такие как не рисовать, каркас, коробка), имен осей, числовых значений;
- *Расположение* – предназначен для детальной настройки положения отображаемой поверхности (угол поворота, смещение, масштаб, зуммирование, расположение по умолчанию);
- *Курсор* – предназначен для установки параметров трассировщика: координаты курсора X и Y в трехмерном пространстве, ширину линии и цвет трассировщика;
- *Значение* – предназначен для хранения Z координаты под курсором мыши.

11.2 Графический редактор для виджетов, основанных на примитиве элементарная фигура

Для редактирования виджетов, основанных на примитиве элементарная фигура, в СКАДА используется векторный графический редактор, позволяющий изображать объекты, характеристики которых могут быть динамически изменены. Вызов графического редактора производится в окне редактирования виджета, основанного на примитиве элементарная фигура, выбором строки «Вход в редактирование виджета» (при нажатии на правую кнопку мыши). Основными элементами графического редактора являются три графических примитива: линия, дуга, кривая Безье (рисунок 135). К динамически изменяющимся характеристикам этих примитивов относятся:

- координаты контрольных точек - используются для задания формы линии, дуги или кривой Безье. При этом линия имеет 2 контрольные точки, дуга 5 контрольных точек (1 - начало дуги, 2 - конец дуги, 3 - центр окружности, 4 - середина дуги (для круга - четверть окружности), 5 - контрольная точка (для круга – половина окружности)), кривая Безье 4 контрольные точки;
- ширина линии;
- ширина бордюра (границы);
- цвет бордюра (границы);
- стиль линии (сплошная, пунктирная, точечная).

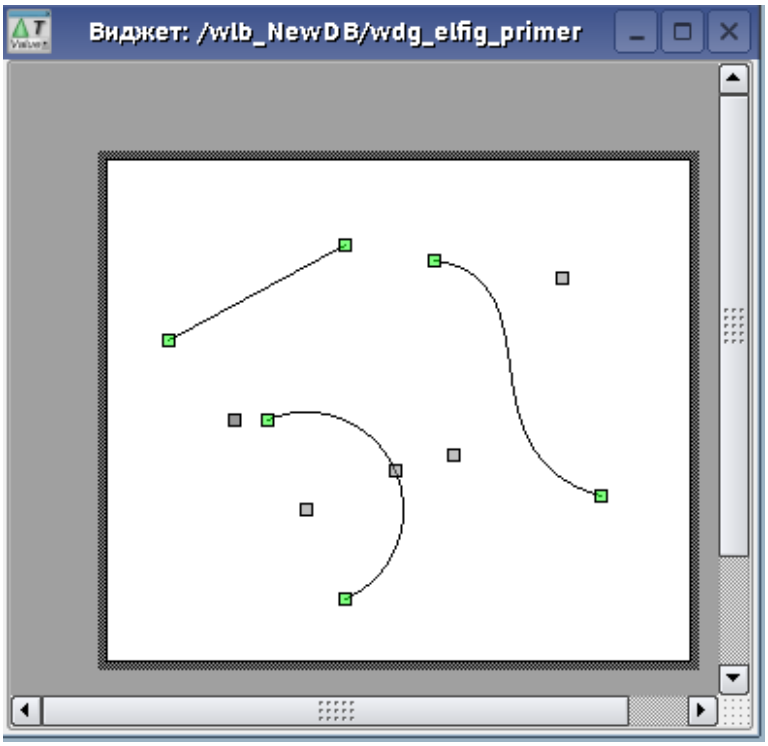


Рисунок 135

Комбинируя указанные элементарные фигуры можно создать более сложные графические объекты. Если связанные примитивы образуют замкнутый контур, то он может быть залит цветом и/или изображением.

К возможностям редактора также относятся выделение, перемещение, копирование и удаление фигур.

Для наглядного управления свойствами элементарной фигуры можно вызвать «Диалог свойств элементарной фигуры» (рисунок 136). Для этого необходимо выделив фигуру нажать правую кнопку мыши и выбрать строку «Показать свойства элементарной фигуры». В результате появится окно диалога, в котором будут отображаться данные выбранной фигуры. Также, предусмотрена возможность включать/исключать отдельные свойства диалога (кнопка ). В случае исключения отдельных свойств они не будут обрабатываться при подтверждении диалога (кнопка «Принять»). При подтверждении диалога все указанные данные для включенных свойств будут применены для всей группы фигур. Диалог для свойств заливки позволяет управлять свойствами отдельной заливки. Кнопки «Дин/Стат» делают соответствующие свойства динамическими либо статическими.

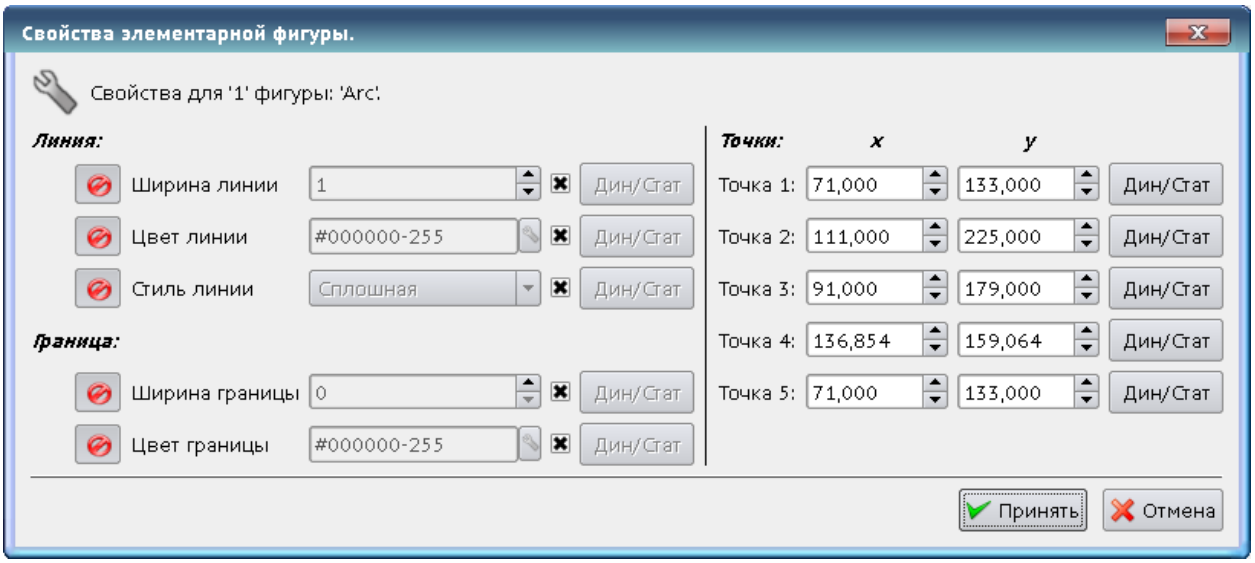


Рисунок 136

11.3 Библиотека основных элементов пользовательского интерфейса (Main)

В своём составе библиотека содержит около двух десятков графических элементов, наиболее востребованных при формировании пользовательского интерфейса управления технологическим процессом (рисунок 137). Для использования большинства элементов библиотеки необходимо добавить виджет на мнемосхему и связать с параметром источника данных (см. часть 3 настоящего руководства).

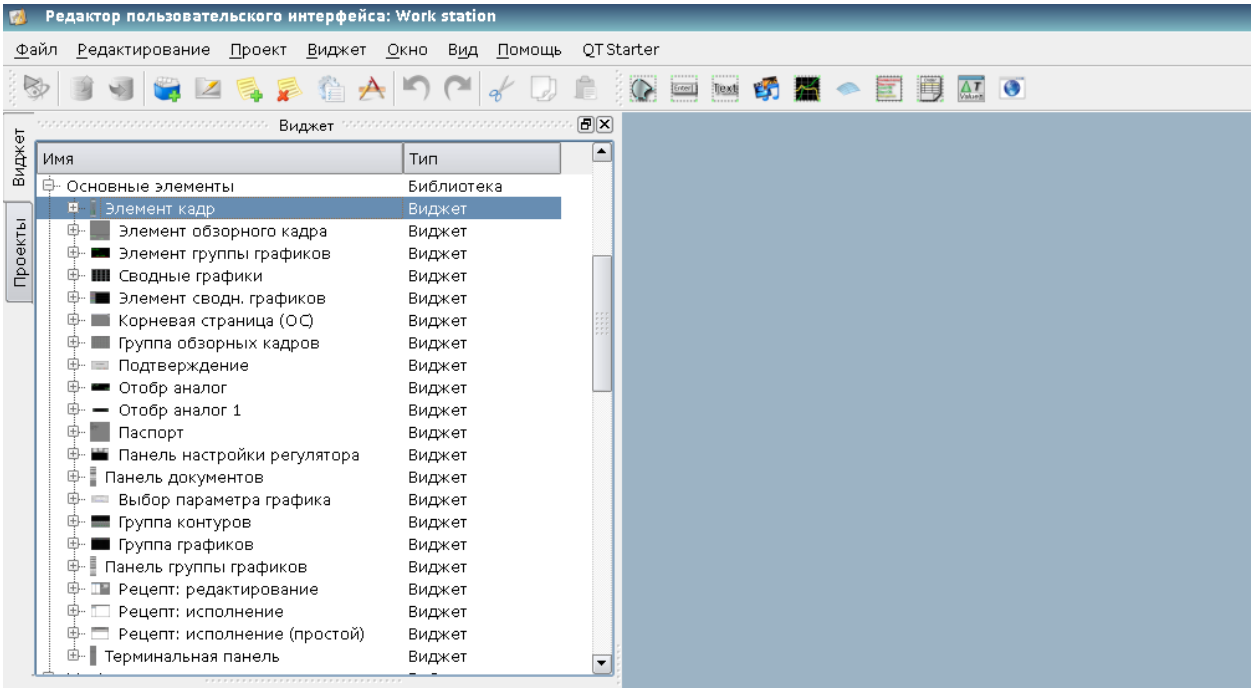
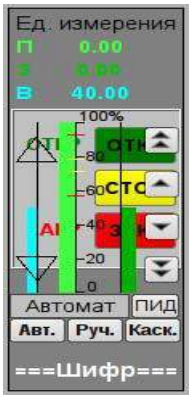
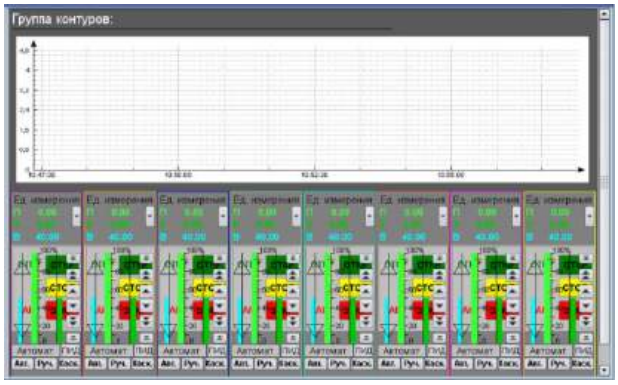
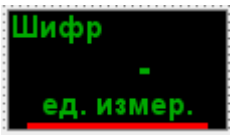


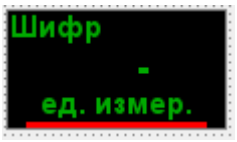
Рисунок 137

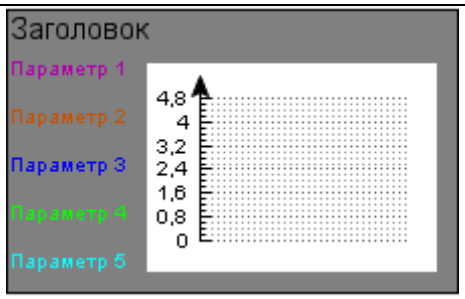
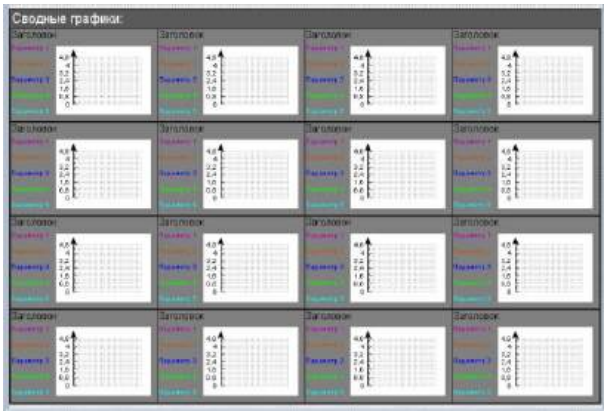
В таблице 11 приведено описание элементов библиотеки.

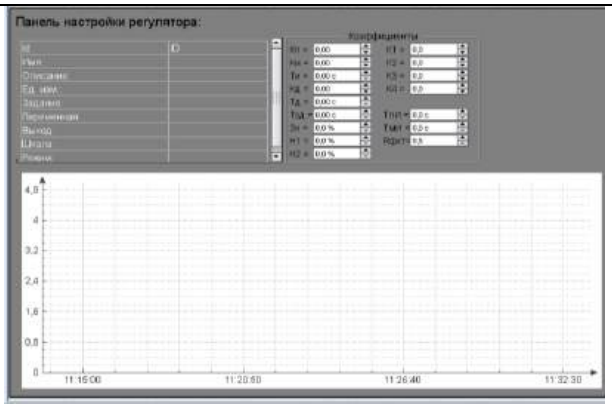
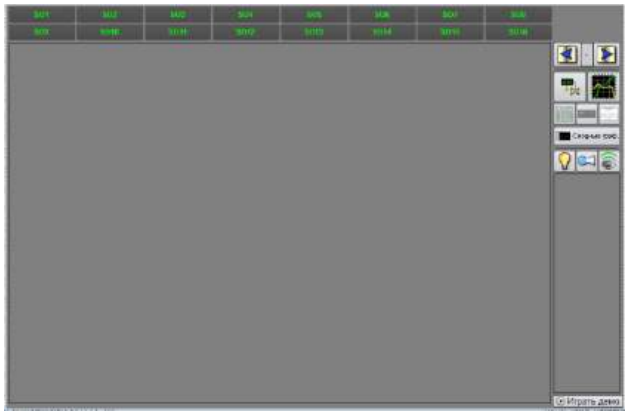
Таблица 11

Графическое изображение	Описание
	*Отобр аналог* Поле отображения аналоговых параметров, таких как: – текущие значения параметров, – наличие сигнализации о превышении уставок, – наличие сигнализации о неисправностях измерительного канала. Текущие значения параметра выводятся в виде чисел с плавающей запятой. Формат чисел определяется при проектировании.
	*Отобр аналог 1* Служит для отображения текущего значения аналогового параметра с односимвольным префиксом типа измеряемой величины.

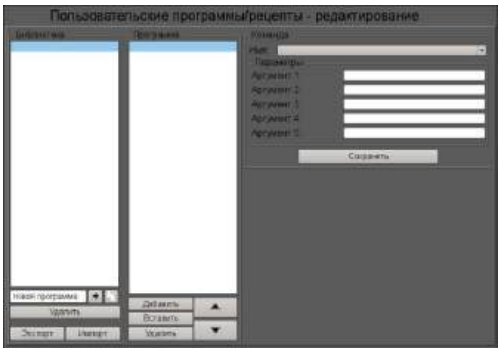
Графическое изображение	Описание
	*Элемент кадр* Элемент является универсальной панелью управления различными устройствами: – аналоговыми: показания, ручные вводы значений и регуляторы (аналоговые и импульсные); – дискретными: клапаны, отсекатели, задвижки, двигатели, вентиляторы и всевозможные переключатели. Заложен в шаблоне проекта "Объекты сигнализации" и, если новый проект создаётся на основе этого шаблона, вызов осуществляется автоматически.
	*Группа контуров* Элемент служит для одновременного наблюдения и управления несколькими контурами (до восьми), включает в себя как экземпляры виджета "Элемент кадр" для каждого контура, так и виджет "Диаграмма" для наблюдения за трендами контуров и просмотра истории. Предназначен для выполнения роли страницы-шаблона и должен непосредственно помещаться в дерево проекта.
	*Элемент обзорного кадра* Служит основой обзорного кадра, отражает текстовую информацию о параметре в виде наименования и значения, а также график (тренд) параметра за небольшой (настраиваемый) промежуток времени для наблюдения за текущей тенденцией поведения параметра с авто-масштабированием по шкале значения. Данный виджет не предназначен для самостоятельного использования, в

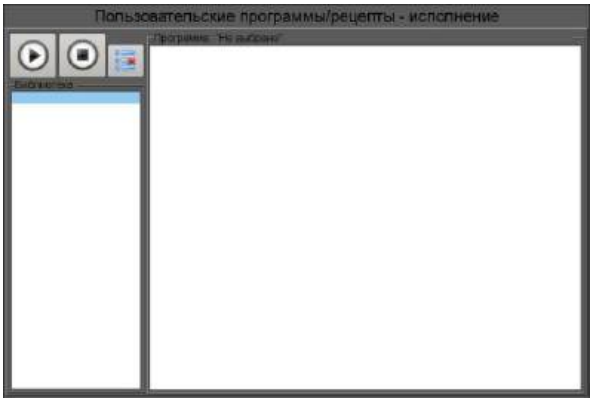
Графическое изображение	Описание
	отрыве от обзорного кадра, но использовать его можно, например, поместив на мнемосхему и установив связь с параметром источника данных.
	*Группа обзорных кадров* Служит для отображения текущих трендов по параметрам объекта сигнализации в количестве до 24 штук, поддерживает функцию масштабирования элементов в зависимости от их количества. Состоит из виджетов "Элемент обзорного кадра". Виджет предназначен для выполнения роли страницы-шаблона и должен непосредственно помещаться в дерево проекта.
	*Элемент группы графиков* Служит для создания групп графиков. Содержит информацию о параметре, режим регулятора, если параметр является таковым, единицы измерения аналогового параметра, а также цвет, соответствующий параметру тренда.
	*Группа графиков* Служит для одновременного наблюдения тренда и управления параметрами объекта сигнализации, включает в себя как экземпляры виджета "Элемент группы графиков" для каждого параметра, так и виджет "Диаграмма" для наблюдения за графиками параметров и просмотра истории, а также горизонтальную полосу прокрутки для быстрой навигации по доступной истории выбранных для отображения параметров.

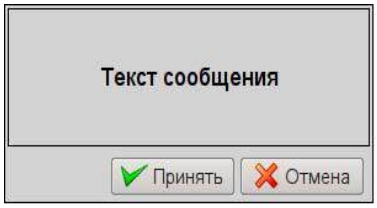
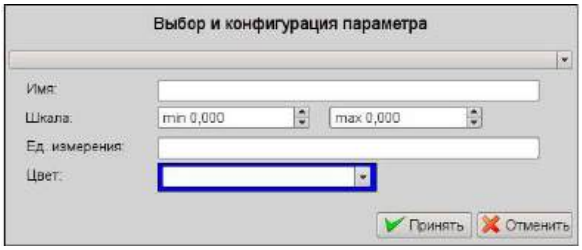
Графическое изображение	Описание
	Виджет предназначен для выполнения роли страницы-шаблона и должен непосредственно помещаться в дерево проекта. К каждому кадру может подключаться до восьми параметров путём установки связей.
	*Элемент сводных графиков* Элемент позволяет отображать тренды по пяти параметрам за указанный промежуток времени и до текущего времени.
	*Сводные графики* Служит для отображения трендов основных параметров по всему проекту визуализации. Предназначен для выполнения роли страницы-шаблона и должен непосредственно помещаться в дерево проекта. К каждому виджету-видеокадру может подключаться до 16*5 параметров путём установки связей. Графики, для которых не будут установлены связи, будут скрыты при исполнении, также возможно масштабирование подключенных графиков для заполнения области всего виджета.
	Панель настройки регулятора Служит для настройки ПИД регулятора, включает в себя информацию о параметре-регуляторе, поля настроек регулятора, и виджет "Диаграмма" для наблюдения за трендами регулятора и просмотра истории. Может использоваться как в роли панели, вызываемой из панели управления параметрами "ElCadr", так и в роли

Графическое изображение	Описание
	страницы-шаблона. Виджет должен непосредственно помещаться в контейнер панелей дерева проекта, где будет осуществляться динамическая линковка на параметр регулятора. Для создания статического перечня контуров настроек регуляторов, с возможностью последующего листания по ним, необходимо поместить их в контейнер контуров регуляторов "greg" каждого объекта сигнализации и статически связать с соответствующим параметром, а также обеспечить равенство идентификатора панели и связанного параметра.
	*Корневая страница* Служит базой для создания пользовательских интерфейсов управления технологическими процессами, основанными на объектах сигнализации. Корневая страница содержит четыре области: 1 "Область кнопок-индикаторов объектов сигнализации" (вверху) — служит для предоставления информации о наличии аварий в объекте сигнализации, а также для переключения между ними. 2 "Область кнопок-режимов отображения" (справа-вверху) — индикация выбора и выбор режима отображения. Содержит также кнопки квитации, которые появляются при возникновении нарушений и кнопки перелистывания страниц мнемосхем. 3 "Контейнер мнемосхем и основных кадров интерфейса оператора" (в центре) — область контейнера для включения в неё мнемосхем и основных кадров при выборе их кнопками режимов отображения или смене объекта сигнализации.

Графическое изображение	Описание
	4 "Контейнер панелей управления" (справа внизу) — область контейнера для включения в ней панелей управления различными объектами в области контейнера мнемосхем, например: панель параметра, документа, графика и т.д. Под контейнером панелей управления располагается кнопка запуска демонстрационного режима – режима, при котором осуществляется периодическое переключение показательных кадров, изменение режимов и других операций согласно сценарию. Данный виджет может использоваться только в режиме корневой страницы, которая должна помещаться в дерево проекта как элемент "/*/so".
	Паспорт Служит для отображения паспорта параметра: детальной информации, включающей шифр, описание, единицы измерения, аварийные границы и т.д. Документ формируется полностью динамически. Данный элемент должен помещаться в логический контейнер панелей дерева проекта. В режиме редактирования этот виджет представляет собой пустой "Документ". Связывание с параметром осуществляется динамически при вызове из элементов представления данных параметра.
	*Панель документов* Служит для управления документами и навигации по их истории. Элементом поддерживаются динамические и архивные документы. Данный элемент должен помещаться в логический контейнер панелей дерева проекта. Связывание с

Графическое изображение	Описание
	параметром осуществляется динамически при вызове из элемента документа.
	*Панель группы графиков* Служит для управления виджетом "Диаграмма", позволяет просмотреть историю графиков за определенный период времени в нужном разрешении, поддерживается масштабирование шкалы, выбор архиваторов для отображения, а также представление графиков в виде спектра присутствующих частот. Данный элемент должен помещаться в логический контейнер панелей дерева проекта. Связывание с параметром осуществляется динамически при вызове из элемента диаграмма.
	*Терминальная панель* Служит для заполнения пустого места, когда не выбран элемент для управления. Элемент должен помещаться в логический контейнер панелей дерева проекта.
	*Рецепт: редактирование* Служит для пользовательского редактирования программ-рецептов. Программа-рецепт представляет собой последовательный вызов блоков функций - команды (макросы), принимающие до пяти аргументов и возвращающие строку результата, с кодом завершения вначале: "Работа" (0), "Завершен" (> 0) и "Ошибка" (< 0). Кадр "Рецепт: редактирование" содержит слева на право: – "Библиотека" — библиотека со списком программ и элементами управления библиотекой.

Графическое изображение	Описание
	– "Программа" — список шагов-команд выбранного в библиотеке рецепта-программы с элементами управления. – "Команда" — поле редактирования выбранного шага рецепта в составе выбора команды и установки значений доступных атрибутов, а также кнопки сохранения изменений. Данный кадр должен быть помещен в логический контейнер мнемосхем или панелей дерева проекта.
	*Рецепт: исполнение* Служит для непосредственного исполнения или наблюдения за исполнением во внешнем вычислителе программ-рецептов, ранее сформированных в кадре Рецепт: редактирование. Кадр "Рецепт: исполнение" содержит слева на право: – "Запуск/останов/пропуск" — две кнопки запуска и останова выбранной программы, а также кнопка пропуска выполнения текущего шага. – "Библиотека" — библиотека со списком программ. – "Программа" — документ списка шагов-команд выбранного в библиотеке рецепта-программы. При исполнении в этом поле отслеживается текущее состояние исполнения путём соответствующей подсветки шагов. Исполняемый рецепт-программа может быть приостановлен путём нажатия кнопки "Пауза" в месте кнопки "Запуск" или прерван путём нажатия кнопки "Останов". Также возможно пропустить шаг, нажав кнопку "Пропустить", в момент исполнения

Графическое изображение	Описание
	шага. По любому завершению рецепта-программы происходит генерация сообщения с параметрами сеанса, а также архивирование документа сеанса. По умолчанию архив документов настроен на глубину 10 документов.
	*Подтверждение* - реализует простейший диалог подтверждения операций. Элемент содержит сообщение с вопросом и две кнопки "Принять" и "Отмена". Данный виджет может быть использован разработчиком при создании кадров динамического взаимодействия в операциях, требующих подтверждения у пользователя. Для использования нужно добавить данный элемент в логический контейнер панелей дерева проекта.
	*Выбор параметра графика* реализует диалог выбора источника данных, часто архивных, для формирования графика в кадре "Группа графиков". Выбор предоставляется из перечня указанного в атрибуте "Параметры доступные для выбора" кадра-инициатора. Для выбранного источника можно указать имя, шкалу, единицу измерения и цвет графика.

11.4 Библиотека «Элементы мнемосхемы» (mnEls)

Библиотека строится на основе примитивов виджетов и модуля JavaLikeCalc, позволяющего создавать вычисления на Java-подобном языке.

Для подключения библиотеки «Элементы мнемосхем» пользовательского интерфейса к проекту необходимо в панели управления графического редактора выбрать пункт Виджеты и из всплывающего меню выбрать библиотеку mnEls (рисунок 138).

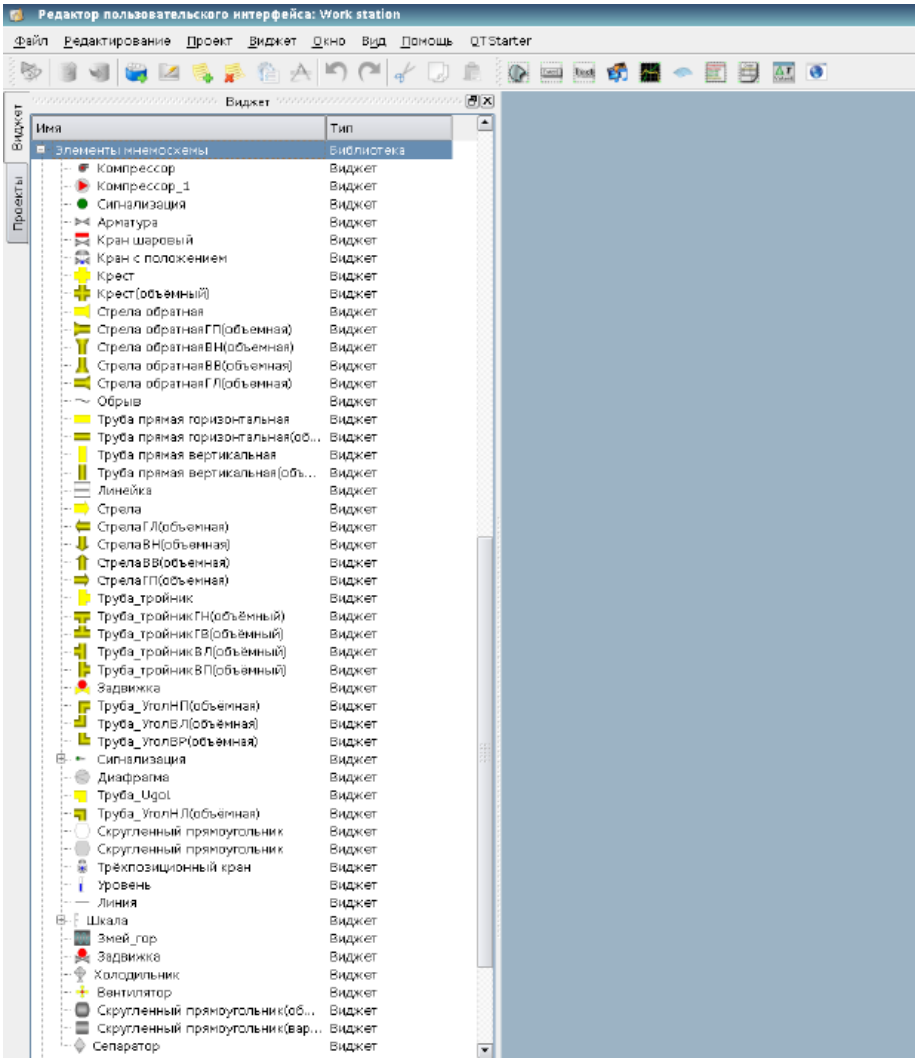


Рисунок 138

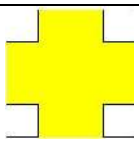
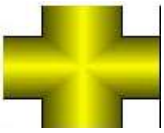
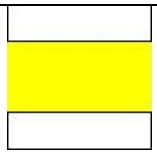
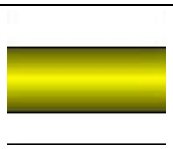
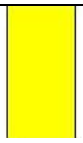
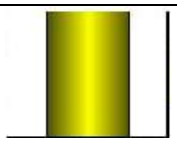
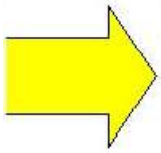
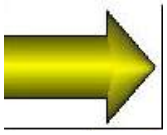
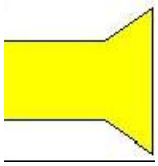
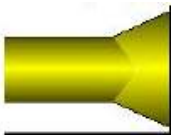
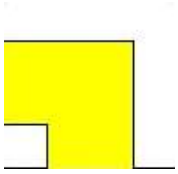
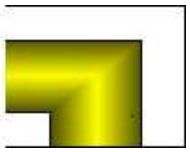
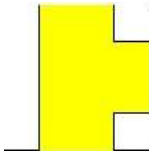
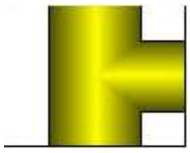
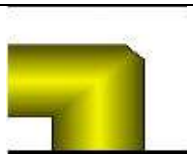
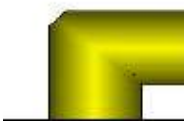
Библиотека содержит 50 графических элементов, необходимых при формировании мнемосхем пользовательского интерфейса управления технологическим процессом. Большинство элементов имеют квадратную форму, позволяющую легко поворачивать и масштабировать их при необходимости (угол поворота всех виджетов по умолчанию равен "0"). Некоторые из них содержат скрипт, описывающий их поведение.

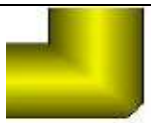
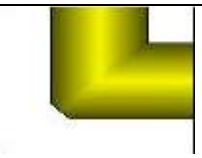
11.4.1 Элементы трубопровода

В таблице 12 приведен перечень элементов, при помощи которых можно выстроить трубопровод любой сложности. По умолчанию все элементы залиты желтым цветом (объемные - полупрозрачными изображениями в градациях серого). Соответствуют ГОСТ 21.206-93.

Таблица 12

Условное изображение		Описание
плоское	объемное	

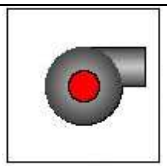
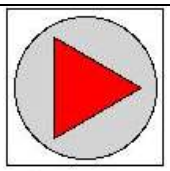
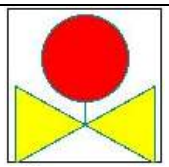
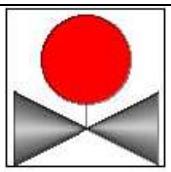
Условное изображение		Описание
плоское	объемное	
		Крест
		Труба прямая горизонтальная
		Труба прямая вертикальная
		Стрела. Виджеты объемного изображения представлены в четырех вариантах в соответствии разными углами поворота
		Стрела обратная. Виджеты объемного изображения представ-лены в четырех вариантах в соответ-ствии разными углами поворота
		Труба_Угол. Виджеты объемного изображения представ-лены в четырех вариантах в соответ-ствии разными углами поворота.
		Труба_тройник. Виджеты объемного изображения представле-ны в четырех вариантах в соответ-ствии с разными углами поворота
		Труба_УголСкруглНЛ(объёмная)
		Труба_УголСкруглНП(объёмная)

Условное изображение		Описание
плоское	объемное	
		Труба_УголСкруглВЛ(объёмная)
		Труба_УголСкруглВП(объёмная)

11.4.2 Элементы, изображающие различные технологические устройства

В таблице 13 приведен перечень элементов библиотеки «Элементы мнемосхемы» - изображений технологических устройств, часто встречающихся при построении мнемосхем различных технологических процессов. В таблице **Ошибка! Источник ссылки не найден.** описаны параметры связывания элементов с источниками данных.

Таблица 13

Графическое обозначение	Описание
	Компрессор
	Компрессор_1
	Задвижка
	Задвижка (объемная)

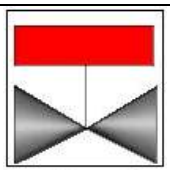
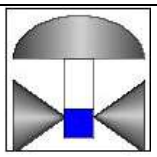
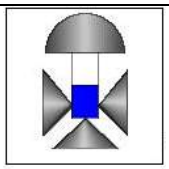
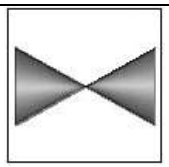
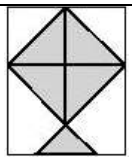
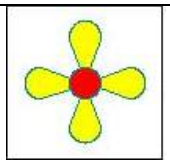
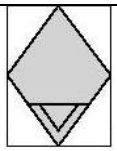
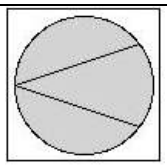
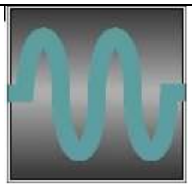
Графическое обозначение	Описание
	Кран шаровый. Включает время хода и время отрыва. В зависимости от состояния меняет цвет.
	Кран с положением. В зависимости от состояния меняет форму.
	Трёхпозиционный кран. В зависимости от состояния меняет форму.
	Арматура
	Холодильник
	Вентилятор
	Сепаратор. Модель сепаратора с двумя фазами, жидкой и газовой
	Диафрагма
	Змей_гор(теплообменник) Модель теплообменника, рассчитывающая теплообмен двух потоков

Таблица 14

ID	Параметр	Тип	Конфигу-рация	Конфигурацион-ный шаблон	Описание
Виджет "Кран шаровый" (El_Kran_Sh)					
com	Команда	Логический	Полная связь	Parametr|com	Команда на закрытие/открытие
shifr	Шифр	Строка	Полная связь	Parametr|NAME	Короткое имя, шифр, параметра
st_close	Состояние "Закрыто"	Логический	Полная связь	Parametr|st_close	Закрытое состояние крана
st_open	Состояние "Открыто"	Логический	Полная связь	Parametr|st_open	Открытое состояние крана
Виджет "Кран с положением" (El_Kran_polozh)					
out	Положение	Вещественный	Входная связь	Parametr|out	Степень открытия/закрытия крана
Виджет "Трёхпозиционный кран" (Kran_3_pos)					
out	Положение	Вещественный	Входная связь	Parametr|out	Степень открытия/закрытия крана
Виджет "Компрессор" (Compressor)					
com	Команда	Логический	Полная связь	Parametr|com	Команда на пуск/останов

Для виджетов "Кран шаровый", "Кран с положением", "Трёхпозиционный кран" описана процедура открытия/закрытия, доступная для просмотра и редактирования на вкладке «Обработка» виджета (рисунок 139).

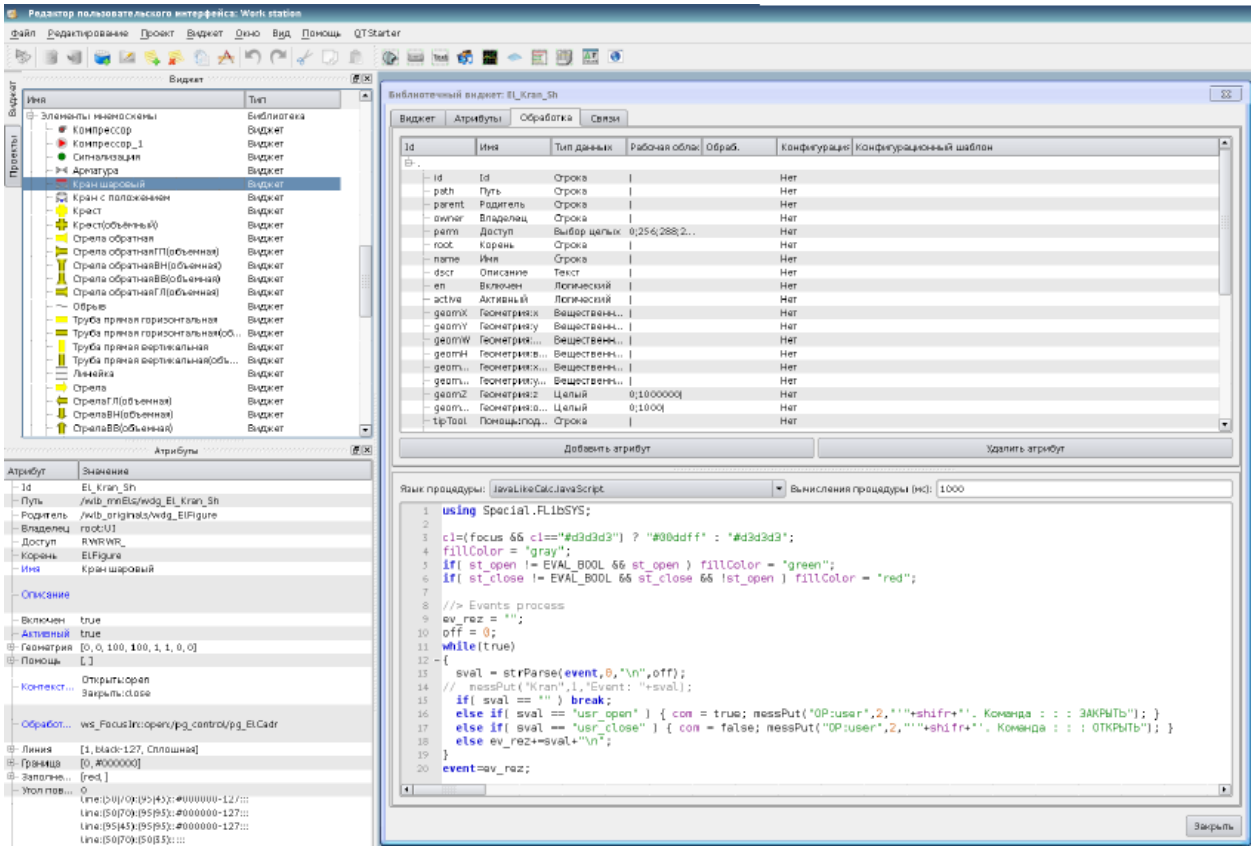
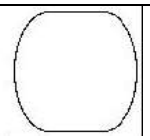


Рисунок 139

11.4.3 Другие элементы

В таблице 15 приведен перечень вспомогательных элементов для построения мнемосхем. Некоторые из них содержат скрипт, описывающий их поведение. В таблице 16**Ошибка! Источник ссылки не найден.** представлены параметры связывания виджетов с источниками данных.

Таблица 15

Условное обозначение	Описание
	Скругленный прямоугольник
	Скругленный прямоугольник (объемный)
	Скругленный прямоугольник (вариант 2)

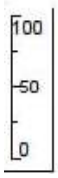
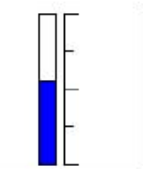
Условное обозначение	Описание
	Шкала
	Уровень
	Линия
	Сигнализация

Таблица 16

ID	Параметр	Тип	Конфигурация	Конфигурационный шаблон	Описание
Виджет "Уровень"					
max	Максимум	Вещественный	Входная связь	Parametr|max	Максимум шкалы
min	Минимум	Вещественный	Входная связь	Parametr|min	Минимум шкалы
var	Значение	Вещественный	Входная связь	Parametr|var	Значение уровня

11.5 Библиотека электроэлементов мнемосхем пользовательского интерфейса (ElectroEls)

Библиотека электроэлементов создана для предоставления электроэлементов мнемосхем пользовательского интерфейса, строится на основе примитивов виджетов и модуля JavaLikeCalc.

Для подключения библиотеки «Электроэлементов» к проекту необходимо в панели управления графического редактора выбрать Виджеты и из всплывающего меню выбрать библиотеку ElectroEls (рисунок 140).

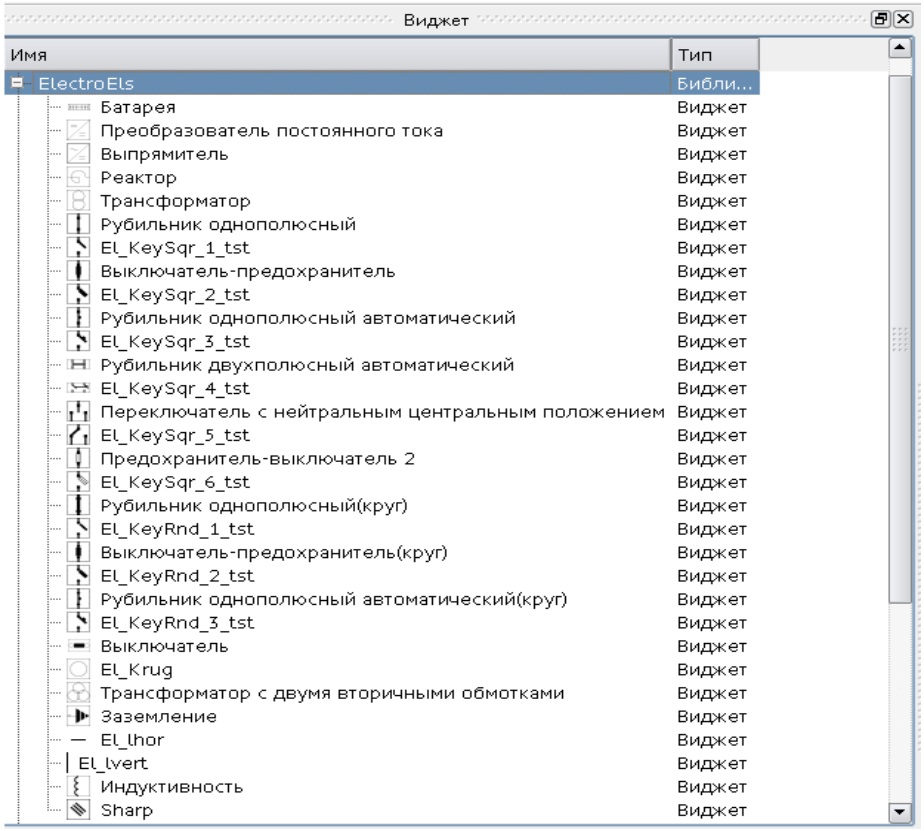


Рисунок 140

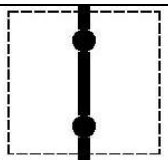
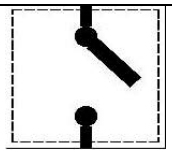
В своём составе библиотека содержит более 20 графических элементов, используемых при формировании мнемосхем пользовательского интерфейса управления технологическим процессом в области электроэнергетики.

По умолчанию все виджеты имеют масштаб по обеим осям, равный "1", а их угол поворота составляет "0" градусов. Возможно изменение значений указанных атрибутов для задания желаемых пропорций.

11.5.1 Динамические элементы библиотеки

В таблице 17 приведен перечень различного вида выключателей и переключателей в различном состоянии. В таблице 18 приведены параметры, посредством которых осуществляется связь элемента с источником данных.

Таблица 1

Графическое изображение элемента		Описание
состояние включен	состояние выключен	
		Рубильник однополюсный(круг)

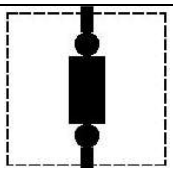
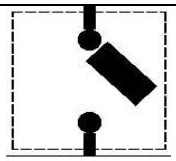
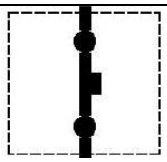
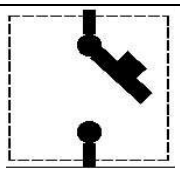
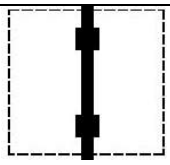
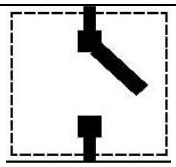
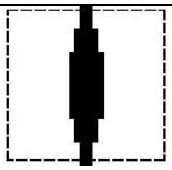
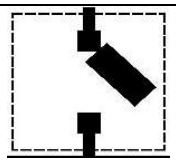
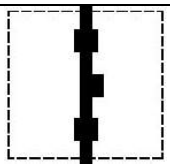
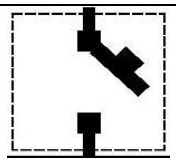
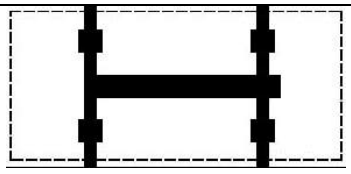
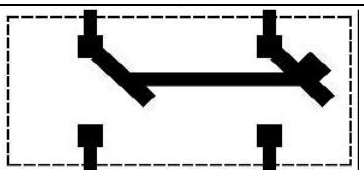
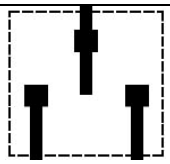
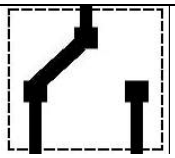
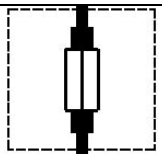
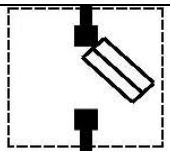
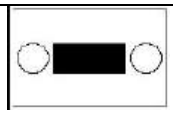
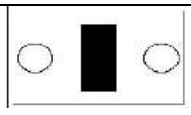
Графическое изображение элемента		Описание
состояние включен	состояние выключен	
		Предохранитель-выключатель(круг)
		Рубильник однополюсный автоматический(круг)
		Рубильник однополюсный
		Предохранитель-выключатель
		Рубильник однополюсный автоматический
		Рубильник двухполюсный автоматический
		Переключатель с нейтральным центральным положением
		Предохранитель-выключатель 2
		Выключатель

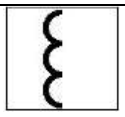
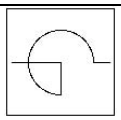
Таблица 28

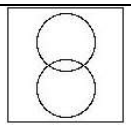
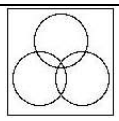
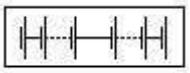
ID	Параметр	Тип	Конфигурация	Конфигурационный шаблон	Описание
Виджеты: "Рубильник однополюсный(круг)" (El_Key_1), "Предохранитель-выключатель(круг)" (El_Key_2), "Рубильник однополюсный автоматический(круг)" (El_Key_3)					
val	Значение	Логический	Входная связь	Parameter|val	
Виджеты: "Рубильник однополюсный" (El_KeySqr_1), "Предохранитель-выключатель" (El_KeySqr_2), "Рубильник однополюсный автоматический" (El_KeySqr_3), "Рубильник двухполюсный автоматический" (El_KeySqr_4), "Предохранитель-выключатель 2" (El_KeySqr_6)					
val	Значение	Логический	Входная связь	Parameter|var	
DESCR	Описание	Строка	Входная связь	Parameter|DESCR	
st	Статус ошибки	Логический	Входная связь	Parameter|st	
Виджет "Переключатель с нейтральным центральным положением" (El_KeySqr_5)					
val	Значение	Логический	Входная связь	Parameter|var	
val1	Значение	Логический	Входная связь	Parameter|var	
st	Статус ошибки	Логический	Входная связь	Parameter|st	

11.5.2 Статические элементы библиотеки «Электроэлементы»

В таблице 39 представлены статические элементы библиотеки. Графическое изображение элементов соответствует требованиям ГОСТ 2.723-68.

Таблица 3

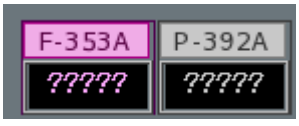
Графическое изображение	Описание
	Индуктивность
	Реактор

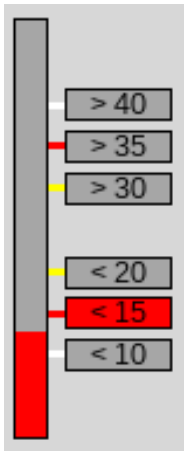
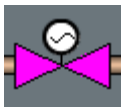
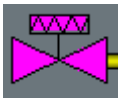
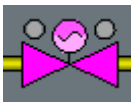
	Трансформатор
	Трансформатор с двумя вторичными обмотками
	Заземление
	Батарея
	Выпрямитель
	Преобразователь постоянного тока

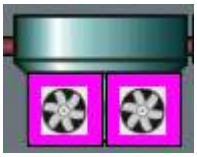
11.6 Элементы библиотеки «NT-tmp»

Библиотека «NT-tmp» содержит элементы технологического оборудования АСУ ТП (таблица 20).

Таблица 20

Графическое изображение	Описание
	*Аналог* Поле отображения аналоговых параметров. В зависимости от типа параметра окрашен в разные цвета: температура (T) -зеленый; давление (P) – серый; уровень (L) – синий; расход (F) – розовый.

Графическое изображение	Описание
	*Гистограмма аналогового параметра* Справа от шкалы гистограммы находятся блоки, отвечающие за отображение срабатывания дискретных уставок (верхних/нижних аварийных/ предупреди-тельных/физических). Внутри блоков находятся значения уставок. Блоки соединены со шкалой линиями, цвет которых зависит от типа уставки: - аварийная уставка – красная линия, - предупредительная уставка – жёлтая линия, - физическая уставка – белая линия
	*Насос* Цвет пиктограммы зависит от режима управления и активности насоса.
	*Электрозадвижка* Состояние задвижки зависит от ее формы и цвета.
	*Клапан отсечной* Изображение зависит от состояния клапана: открыт, закрыт, промежуточное положение, неопределенное положение (одновременно присутствуют сигналы открытого и закрытого состояния задвижки).
	*Клапан регулирующий* Рядом с клапаном изображается регулируемый параметр. Состояние задвижки зависит от ее формы и цвета.

Графическое изображение	Описание
	*Аппарат воздушного охлаждения* В зависимости от состояния аппарата элемент окрашивается в разные цвета: зеленый - включен, розовый - выключен.

На мнемосхемах состояние технологического оборудования индицируется изменением цветовой гаммы пиктограммы следующим образом:

- зеленый цвет – кран открыт, авария отсутствует;

- красный цвет – кран закрыт, аккумулятор включен, питание датчика – включено, авария питания/ связи/аккумулятора/220;

- желтый цвет – кран в промежуточном состоянии;

- серый цвет – аккумулятор отключен, питание датчика отключено.

По всем элементам таблицы выводится окно управления. Окно управления аналогом представлено на рисунке 141.

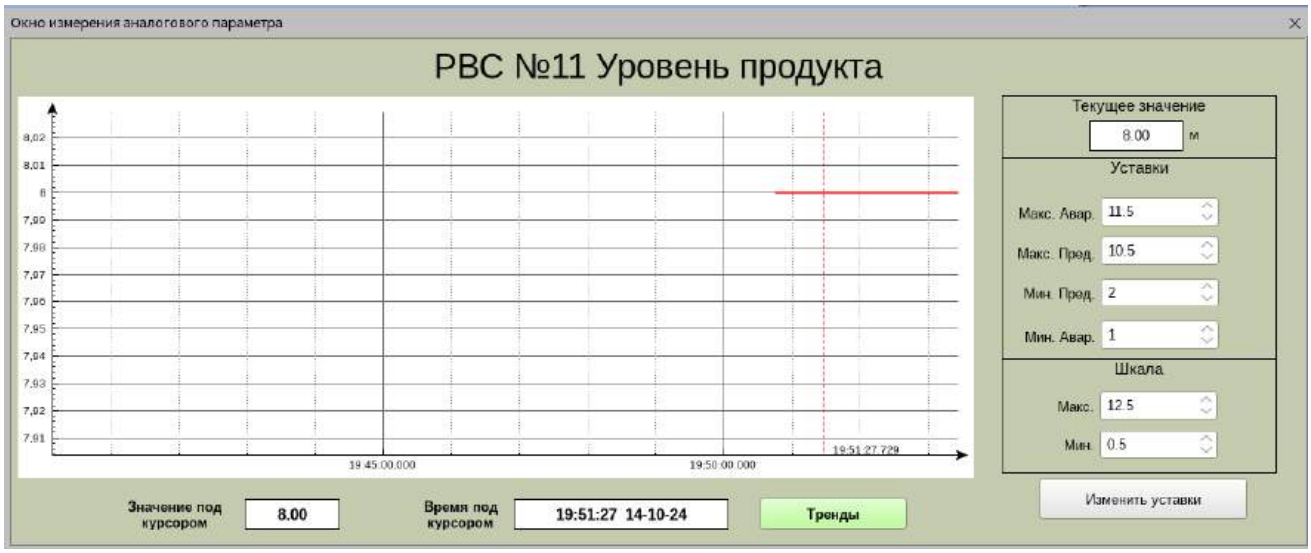


Рисунок 141

12. РАЗРАБОТКА ПРОЕКТА

12.1 Общая часть

Непосредственная конфигурация и свойства конечного интерфейса визуализации содержатся в проекте интерфейса визуализации СВУ. Может быть создано множество проектов интерфейсов визуализации.

Каждый проект включает видеокадры из библиотек видеокадров/виджетов. Видеокадр предоставляет инструмент для привязки динамики к описанным в нём свойствам. Все свойства видеокадра могут быть связаны с динамикой или

определены константами, а могут выступать в роли шаблона для формирования производных страниц.

Для создания нового проекта в графике в СКАДА необходимо:

– создать базу данных для хранения проекта в PostgreSQl;

– в системном конфигураторе СКАДА создать БД с обязательной привязкой к БД, сделанной в предыдущем пункте;

– создать в системном конфигураторе источники данных и переменные;

– в редакторе пользовательского интерфейса в подсистеме «Виджеты» создать:

- библиотеки визуальных элементов проекта путем копирования либо проектирования новых;
- библиотеки видеокадров;
- создать и заполнить видеокадры элементами;

– создать проект в подсистеме «Проекты» редактора пользовательского интерфейса с подключением созданных ранее библиотек и элементов.

Источниками данных являются контроллеры с параметрами, созданные в подсистеме «Сбор данных» с использованием шаблонов или спроектированные самостоятельно в зависимости от требований поставленной задачи.

Далее рассмотрим выполнение каждого шага более подробно.

12.2 Создание БД для базового проекта

Проект имеет модульную структуру и создается на основе различных библиотек: библиотеки видеокадров, библиотеки динамических и статических элементов видеокадра, библиотеки сервисных элементов (кнопки вызова видеокадров, индикаторы времени и т.д.).

В СКАДА все элементы хранятся в БД (предпочтительнее использовать PostgreSQL). Допустимо произвольное разделение элементов по БД: возможно хранение всех элементов в одной БД, каждого элемента в отдельной БД, хранение каждой группы элементов в своей БД. Для больших проектов удобнее разделение библиотек по группам элементов. Рассмотрим далее создание проекта, данные которого хранятся в одной БД PostgreSQL.

12.2.1 Создание новой БД в PostgreSQL

Создать новую БД можно двумя способами: используя приложение pgAdmin или выполнив команду в терминале Fly.

12.2.1.1 Создание БД через приложение pgAdmin

Для создания новой БД в PostgreSQL необходимо зайти на вкладку «Разработка» из меню «Пуск» и запустить приложение pgAdmin: 

.

В окне «Браузер объектов» выбрать postgres (localhost:5432) → Базы данных и нажать на правую кнопку мыши (рисунок 1). Из всплывающего меню выбрать пункт «Новая база данных…» и в появившемся окне «Новая база данных» ввести имя «NewDB» (рисунок 2).

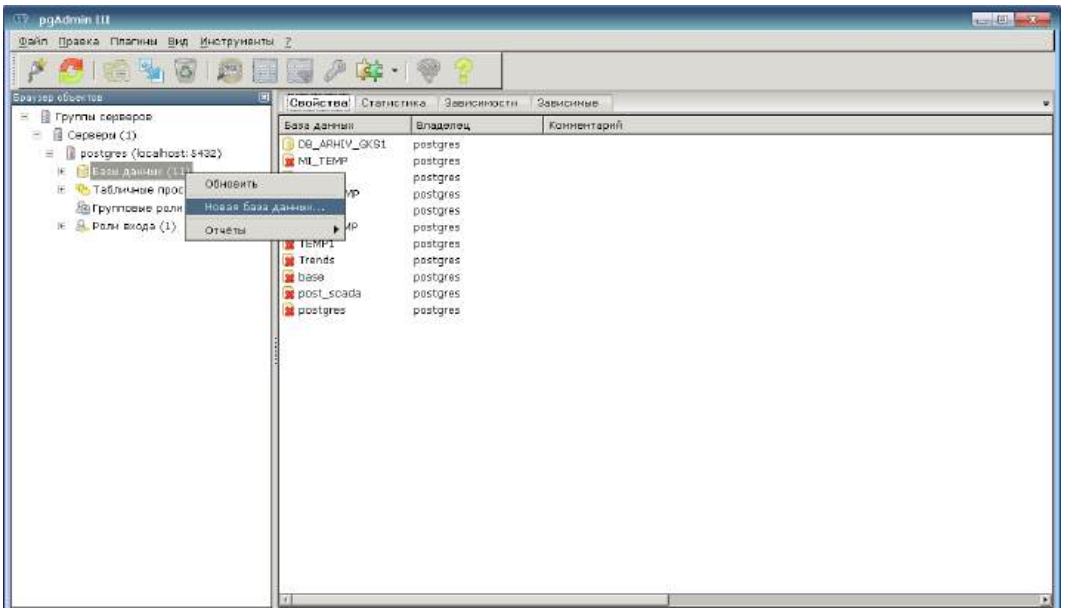


Рисунок 142

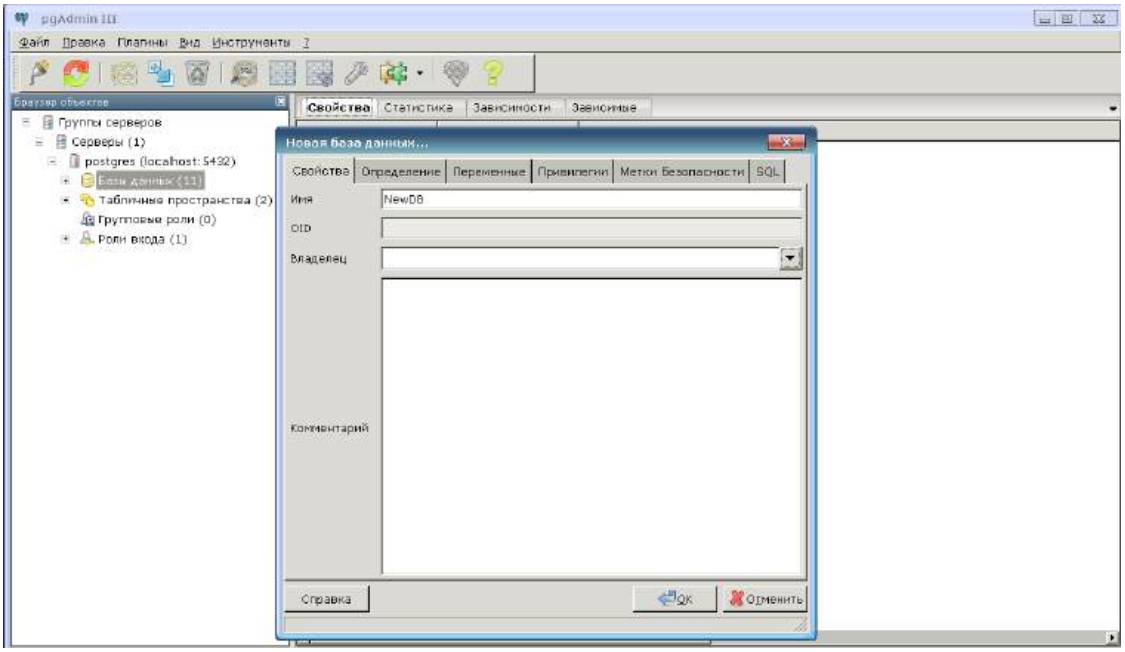


Рисунок 143

Перейдя на вкладку «Определение» можно задать характеристики БД: кодировку, шаблон БД и другие (рисунок 3). Подтвердить создание новой БД нажатием на кнопку ОК и выйти из приложения pgAdmin.

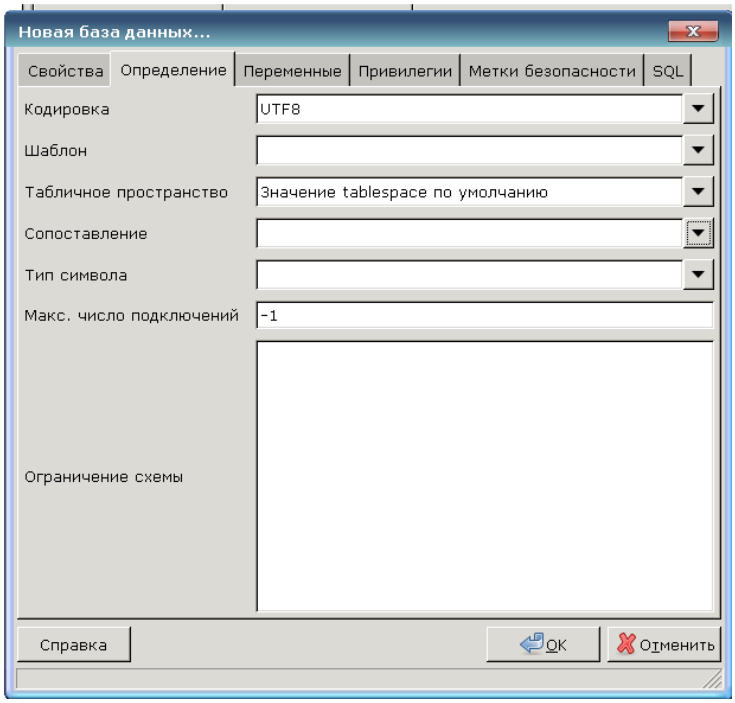


Рисунок 144

12.2.1.2 Создание новой БД командой в терминале Fly

Для создания новой БД необходимо вызвать окно терминала и запустить оболочку psql от имени непривилегированного пользователя ОС - postgres (администратор БД), выполнив команду:

psql –U postgres

При этом система попросит ввести пароль для пользователя postgres. В результате, в окне терминала высветится приглашение postgres=#.

Далее следует создать новую базу данных командой:

create database <имя БД>;

Выйти из программной оболочки psql командой \q.

12.2.2 Добавление новой БД в СКАДА.

Для создания новой БД в левой части окна конфигуратора СКАДА (рисунок 4) раскрыть вкладку «Базы данных». В появившемся списке выбрать вкладку «БД PostgreSQL» и нажатием правой клавиши «мыши» на ней вызвать контекстное меню, в котором выбрать пункт «Добавить».

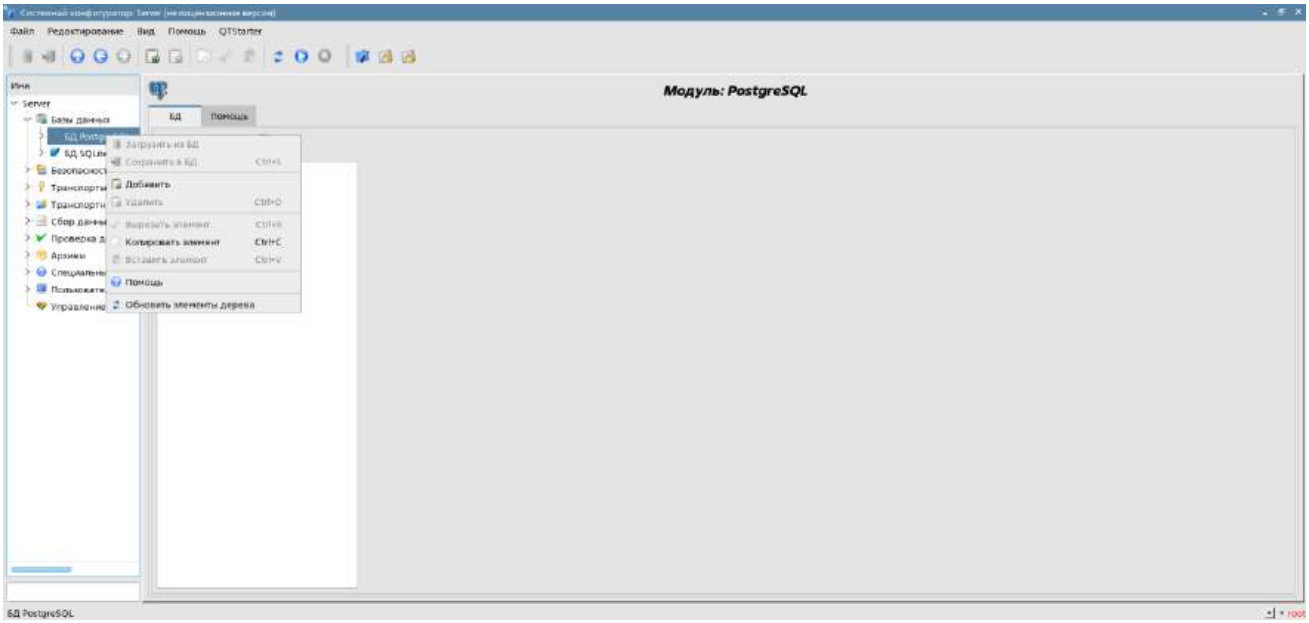


Рисунок 145

В появившемся окне (рисунок 5) необходимо задать имя и «ID» (идентификатор) БД, в которой будет храниться проект. Имя используется для отображения пользователю. Имена любых объектов СКАДА ограничены размером в 50 символов и могут содержать любые символы (цифры, буквы любого алфавита, знаки и т.д.). Если имя объекта остается пустым, то вместо него для отображения будет использоваться идентификатор объекта. Идентификатор (ID) используется для обращений к БД внутри СКАДА. Идентификатор может содержать только буквы латинского алфавита, цифры, символы «-» и «_», кроме того начинать идентификатор рекомендуется с буквы. Длина «ID» ограничена 20 символами. После задания имени и идентификатора БД закрыть окно нажатием кнопки «OK» (рисунок 5).

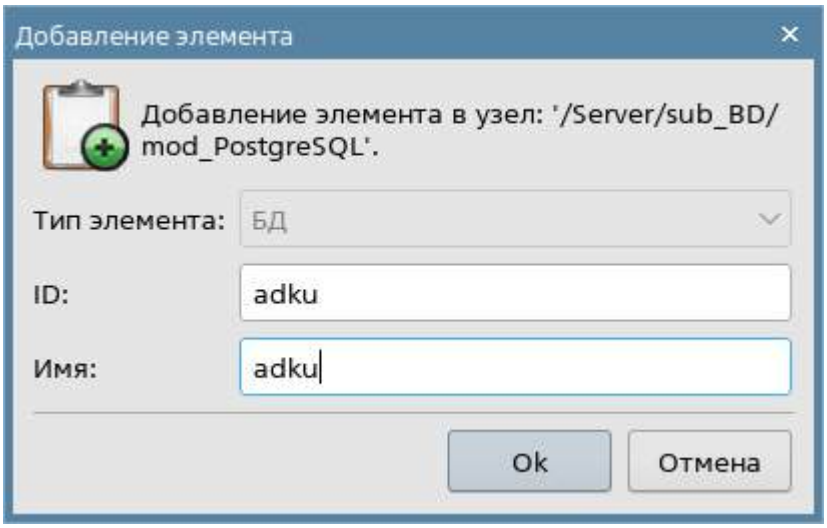


Рисунок 146

В появившемся окне (рисунок 6) в поле ввода «Адрес» необходимо задать полный путь к файлу БД, в котором будет храниться проект, и установить отметку около пункта «Включать» (указывает на состояние БД при загрузке).

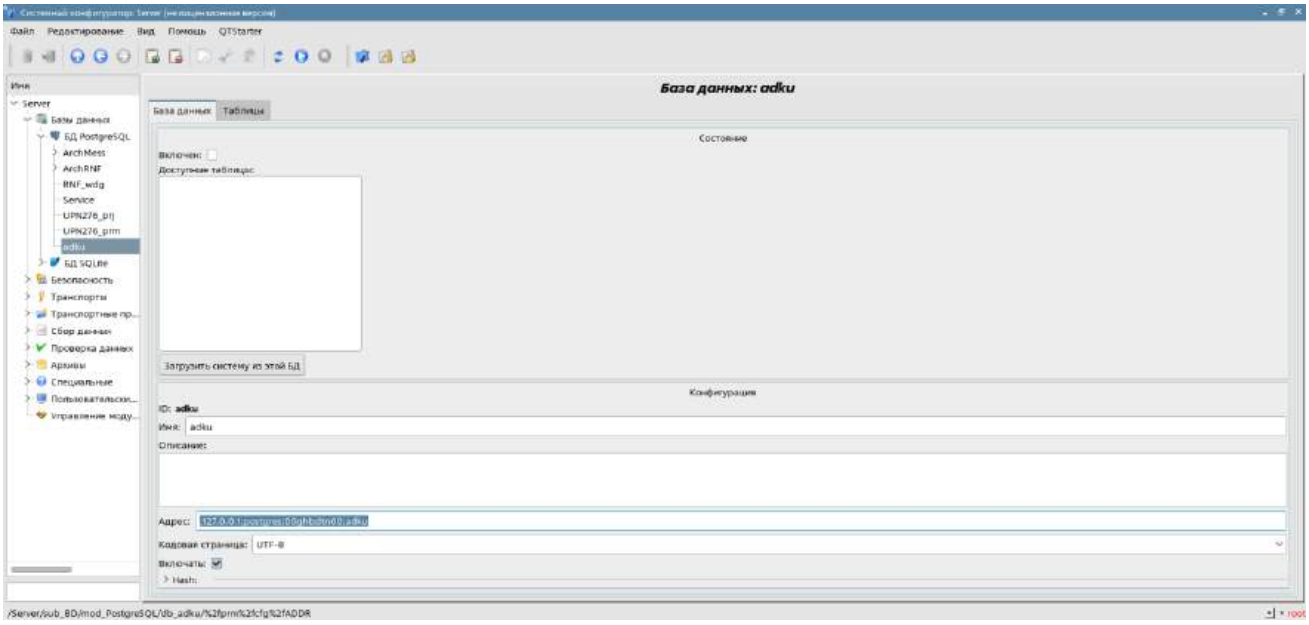


Рисунок 147

Включить БД, установив отметку возле пункта «Включен» (рисунок 148).

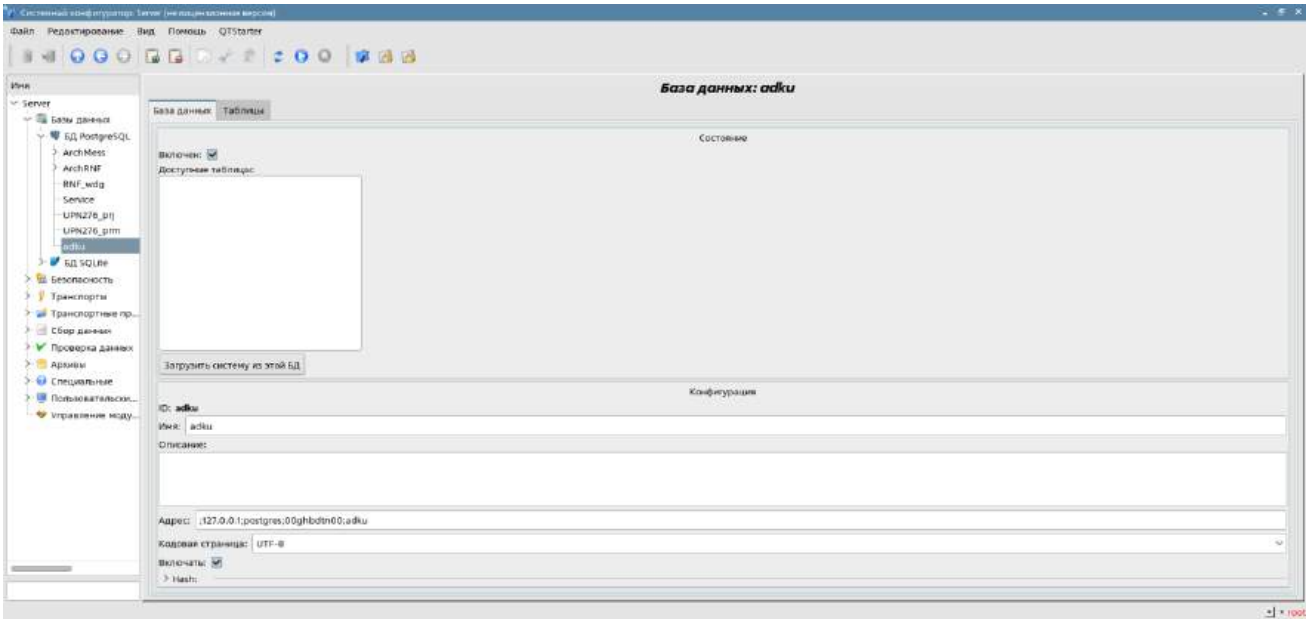


Рисунок 148

12.2.3 Сохранение БД.

Сохранить изменения можно, нажав на пиктограмму окна конфигуратора  либо выбрав правой кнопкой мыши «Сохранить в БД», либо нажав на клавиатуре сочетание клавиш «Ctrl+S» (сохранить) и подтвердить сохранение.

12.3 Создание источников данных

Для примера создадим контроллер «primer» для отображения параметров на Видеокадре 1.

Для этого в Системном конфигураторе СКАДА выберем «Сбор данных» → «Логический уровень» и, выбрав в контекстном меню строку «Добавить», создадим контроллер с ID и именем «primer» (рисунок 149).

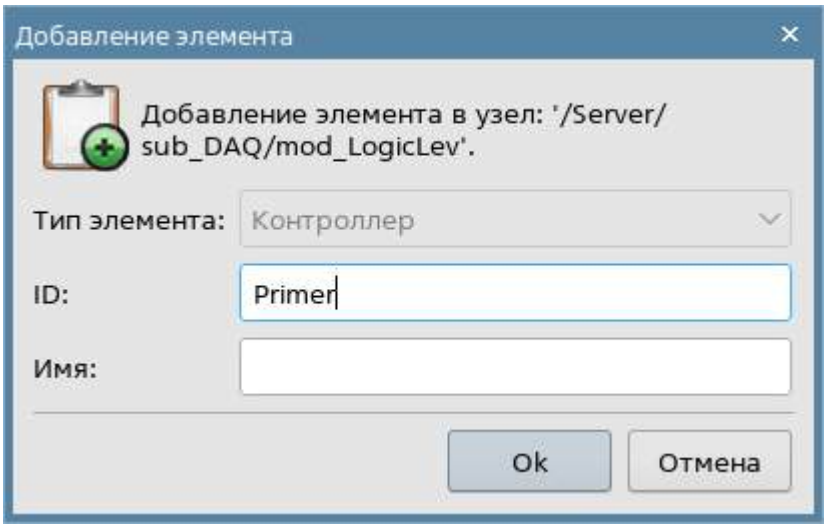


Рисунок 149

Далее необходимо сконфигурировать контроллер. Для этого на вкладке «Контроллер» установить БД контроллера, выбрав из всплывающего списка «БД контроллера». Элемент списка должен иметь имя вида «PostgreSQL.<Имя БД>». Где <Имя БД> соответствует «ID» ранее созданной БД - «adku». После чего, поставить галочки «включен» и «запущен» (рисунок 150).

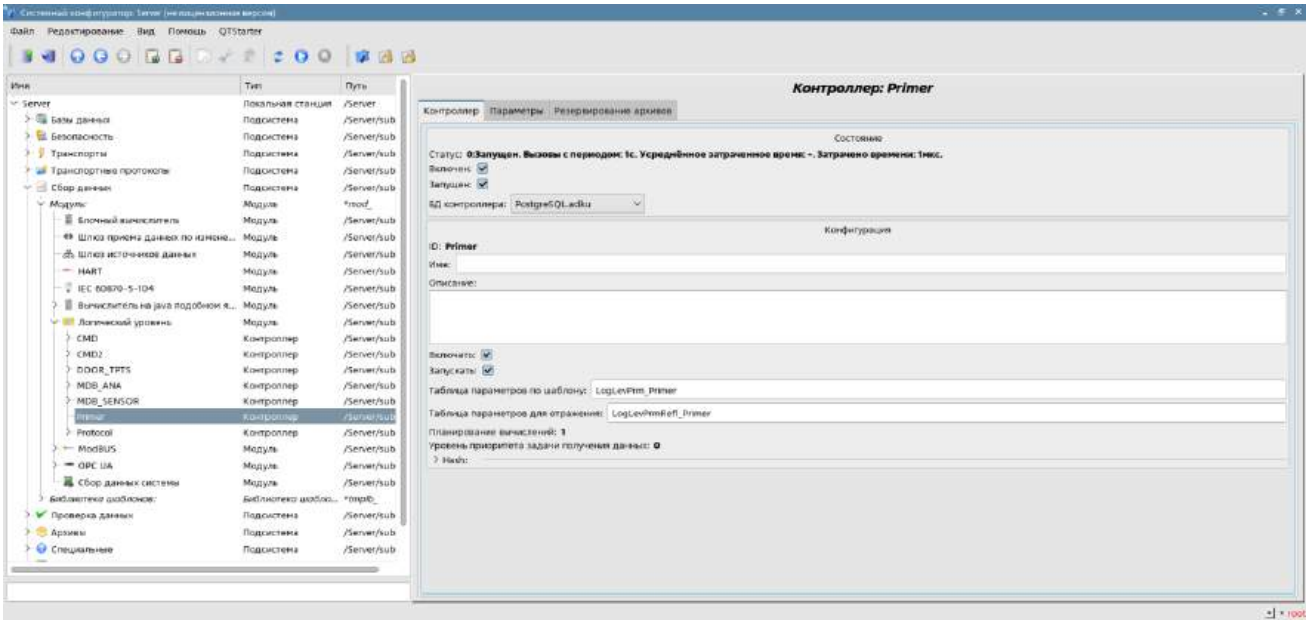


Рисунок 150

Открыв вкладку «Параметры» контроллера «primer», добавить, например, 5 параметров: A1_Ti, A1_To, A2_Ti, A2_To, C (для отображения температуры на входе, на выходе и суммы) (рисунок 151).

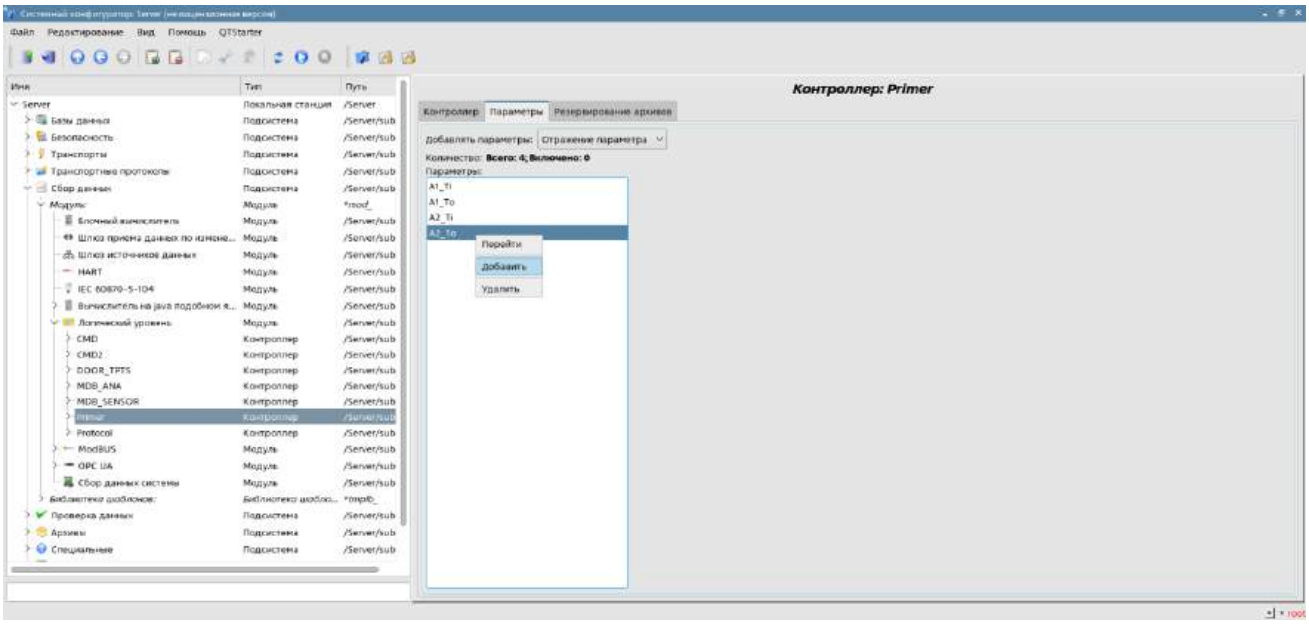


Рисунок 151

Для всех параметров необходимо указать состояние «включен» и задать шаблоны (рисунок 152).

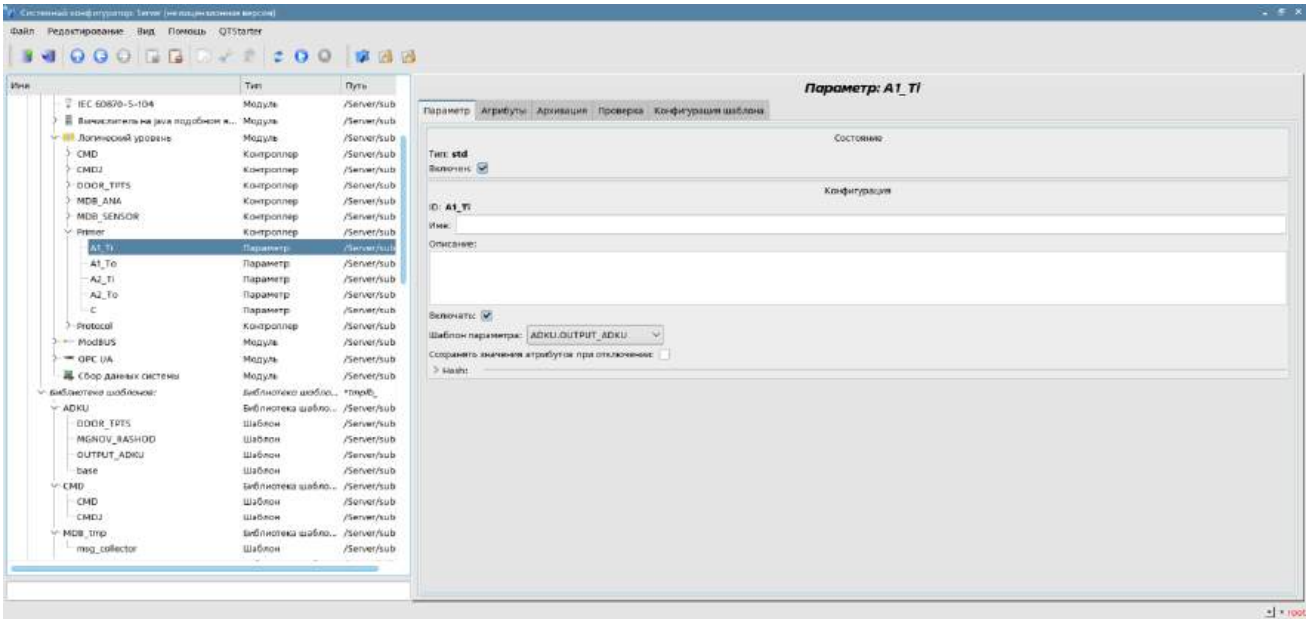


Рисунок 152

На этом создание контроллера Primer считается оконченным.

12.4 Создание библиотеки базовых визуальных элементов

Базовыми визуальными элементами для проекта являются такие элементы как: главное окно, кнопки перехода на видеокадры, индикатор времени, базовые

элементы отображения значений параметров контроллеров и т.д. Новые видеокадры, предназначенные впоследствии для помещения в проект, принято создавать в библиотеке виджетов.

Для открытия окна интерфейса "Vision" нажать крайнюю правую иконку на панели инструментов конфигуратора «Рабочий пользовательский интерфейс (Qt)» (рисунок 153).

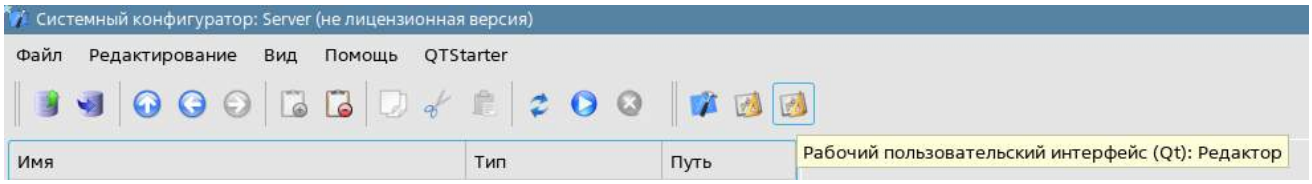


Рисунок 153

В открывшемся окне интерфейса визуализации выбрать в главном меню пункт «Виджет» и в открывшемся контекстном меню выбрать пункт «Новая библиотека» (рисунок 154).

Рисунок 154

В появившемся окне (рисунок 155) задать идентификатор (ID) и имя библиотеки. Нажать кнопку «Принять». Идентификатор используется для обращений к библиотеке внутри системы. Имя библиотеки используется для отображения в списке библиотек виджетов.

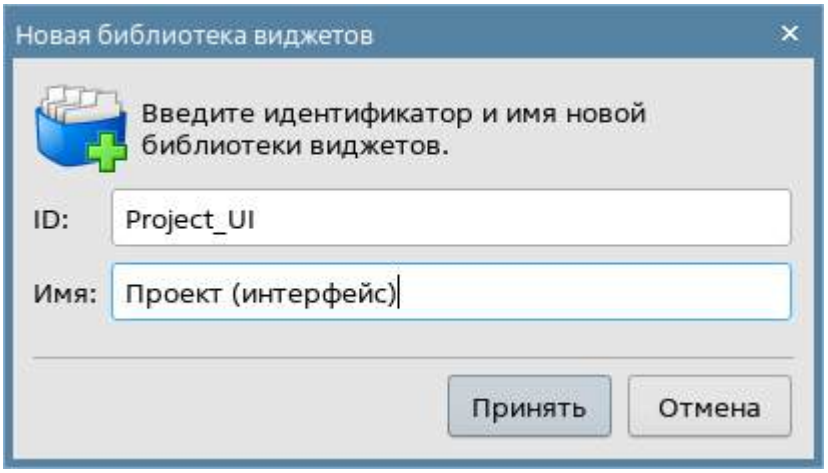


Рисунок 155

Выбрать в левой части окна интерфейса визуализации вертикальную вкладку «Виджет» (рисунок 156) и в появившемся списке выбрать созданный элемент «Проект (интерфейс)».

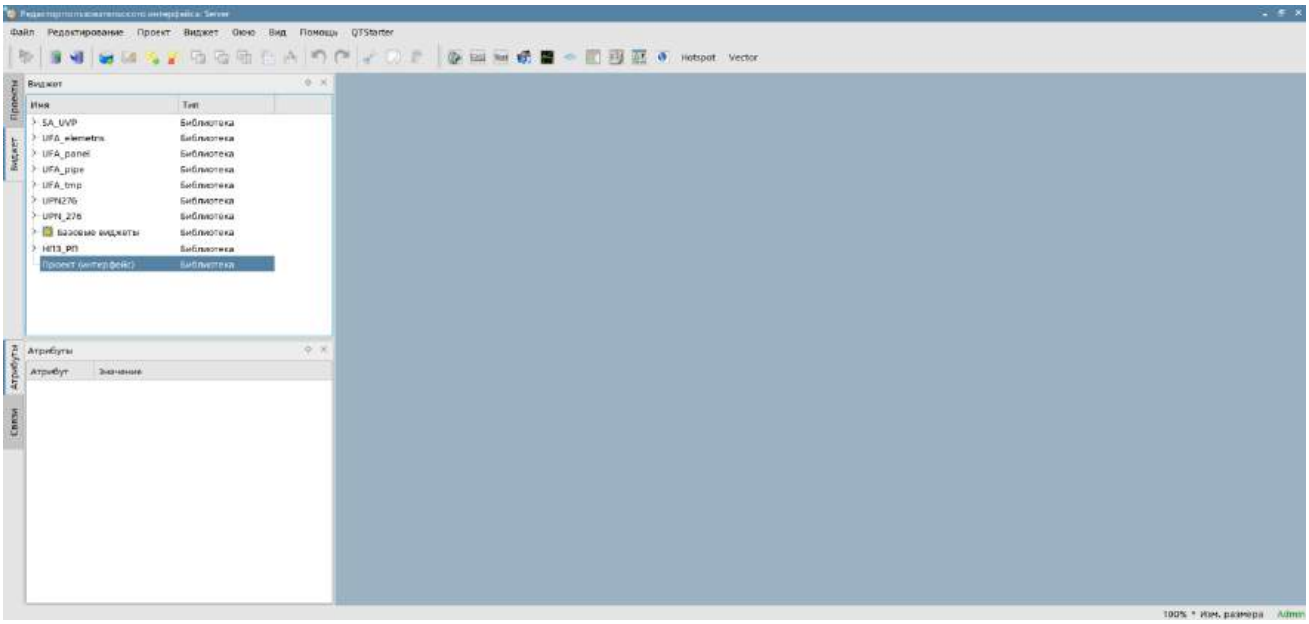


Рисунок 156

Выбрать «Свойства визуального элемента» из всплывающего списка или нажать сочетание клавиш «Ctrl+P» и в появившемся окне (рисунок 16) раскрыть список «БД контейнера». Выбрать базу данных (БД) для хранения библиотеки визуальных элементов. Элемент списка должен иметь имя вида

PostgreSQL.<Имя БД>.<Имя библиотеки>

где *<Имя БД>* соответствует «ID» ранее созданной БД «adku»;

<Имя библиотеки> соответствует «ID» настраиваемой библиотеки «PROJECT_UI».

После выбора элемента нажать кнопку «Закрыть».

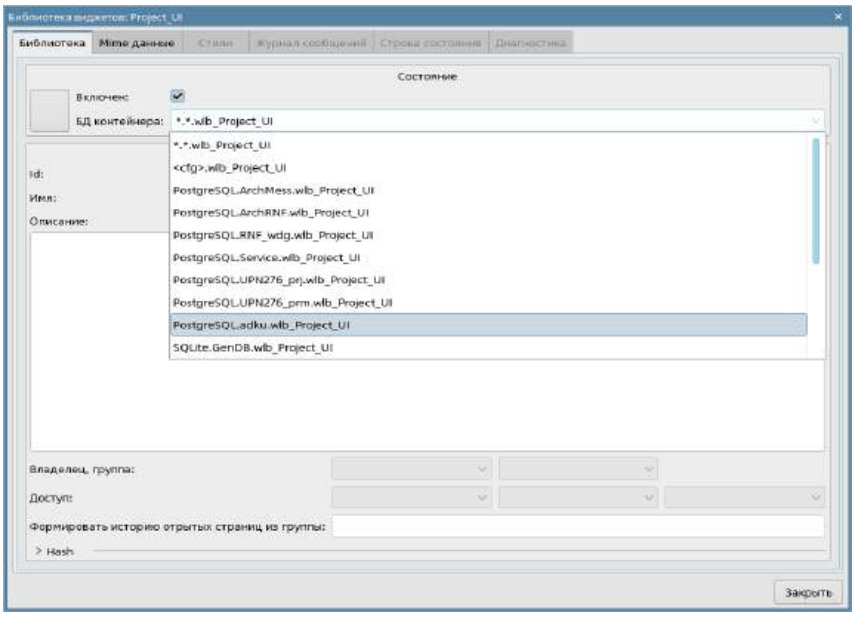


Рисунок 157

Выбрать библиотеку в списке виджетов и сохранить, нажав на пиктограмму  окна конфигуратора, либо выбрав правой кнопкой мыши «Сохранить в БД», либо нажав на клавиатуре сочетание клавиш «Ctrl+S». В появившемся диалоговом окне подтверждения сохранения (рисунок 158) нажать кнопку «Принять».

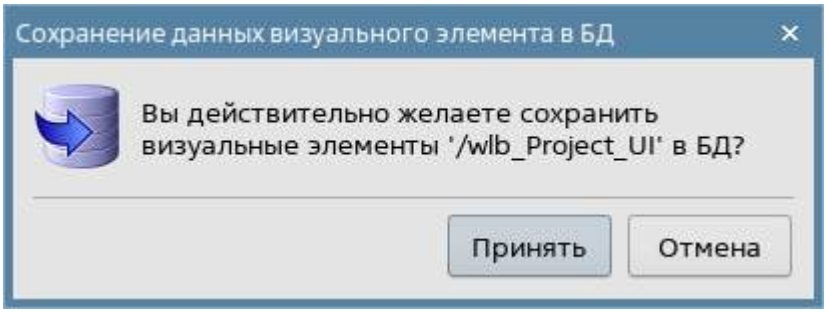


Рисунок 158

Для разделения по функциональным группам аналогичным образом необходимо создать библиотеку элементов с ID «Project_EL» и именем «Проект (элементы)» и библиотеку видеокадров с ID «Project_VF» и именем «Проект (ВК)». В первую библиотеку поместим элементы видеокадров для использования в проекте (линии, элементы отображения аналоговых значений и т.п.), во втором – создадим видеокадры (мнемосхемы, предназначенные для отображения технологического процесса).

12.5 Создание элемента в библиотеке «Проект (интерфейс)»

СКАДА позволяет создавать новые элементы на основе базовых шаблонов или копированием уже существующих элементов. В библиотеке «Проект (интерфейс)» создадим элемент «Главное окно», в котором в дальнейшем будет осуществляться переключение по видеокадрам и отображаться текущая дата.

12.5.1 Создание виджета на основе базовых шаблонов.

Для создания нового видеокадра необходимо, выделив библиотеку «Проект (интерфейс)», выбрать пункт «Библиотека: originals» → «Группа элементов» в контекстном меню созданной библиотеки. В диалоге ввода имени указать идентификатор «Main» и имя «Главное окно», а затем подтвердить изменения. В основе любого видеокадра и страницы должен лежать элемент «Группа элементов».

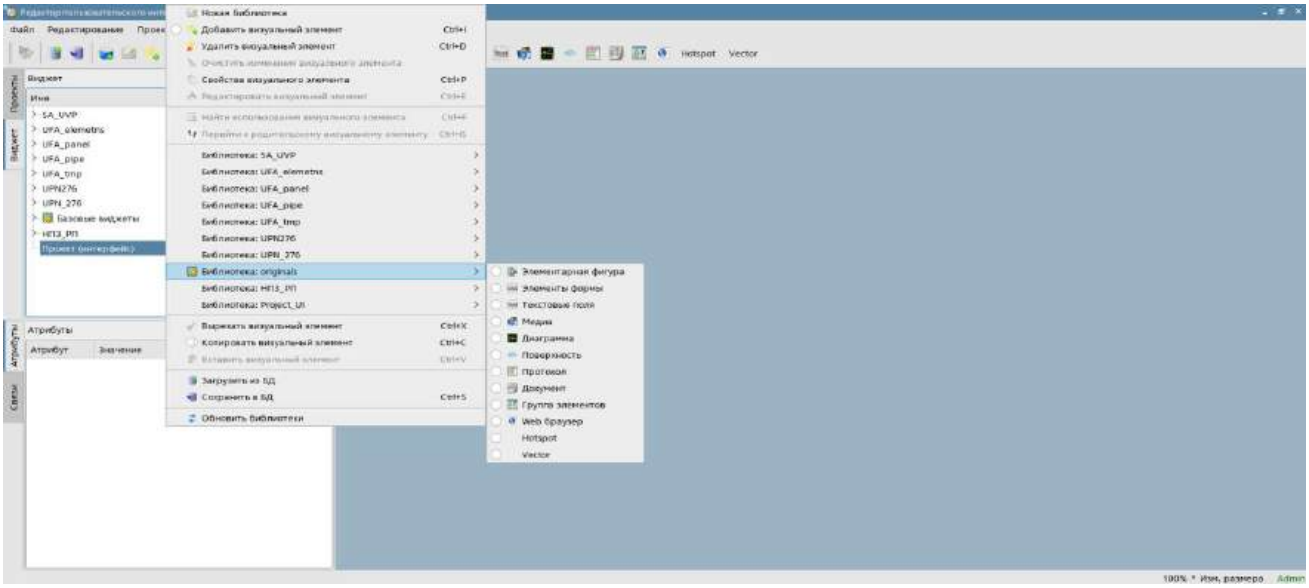


Рисунок 159

Сразу после создания элемента нового видеокадра можно установить его базовые свойства. Свойства или атрибуты любого визуального элемента можно указать в панели инструментов «Атрибуты», предварительно выбрав нужный визуальный элемент или нажав «горячие клавиши» на клавиатуре «Ctrl+P». Выберем созданный Элемент «Главное окно» и установим для него следующие свойства:

- Геометрия: ширина - 500;
- Геометрия: высота - 500;
- Граница: ширина - 1;
- Граница: цвет - «black».

В результате получим пустой видеокадр (рисунок 160), готовый для добавления элементов на него. Для редактирования или просмотра вида видеокадра необходимо в контекстном меню видеокадра выбрать пункт «Редактировать визуальный элемент» или нажать сочетание клавиш «Ctrl+E».

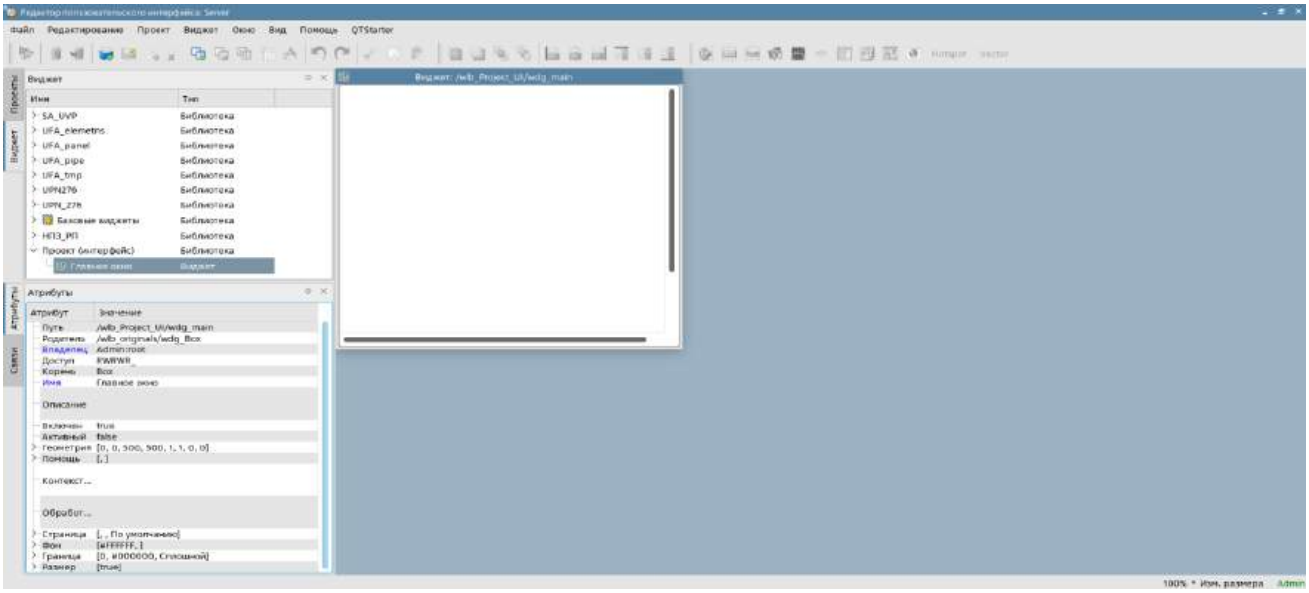


Рисунок 160

12.5.2 Создание элемента видеокадра копированием.

Для создания элемента копированием необходимо скопировать библиотеку из уже созданных библиотек. Например, можно выбрать в библиотеке виджетов «SA_UVP» → элемент «AI» (рисунок 161) и выбрать строку «Копировать визуальный элемент» контекстного меню (при нажатии правой кнопкой мыши) или нажать сочетание клавиш «Ctrl+C». Затем выбрать библиотеку «Проект (интерфейс)» и произвести вставку элемента - нажав сочетание клавиш «Ctrl+V».

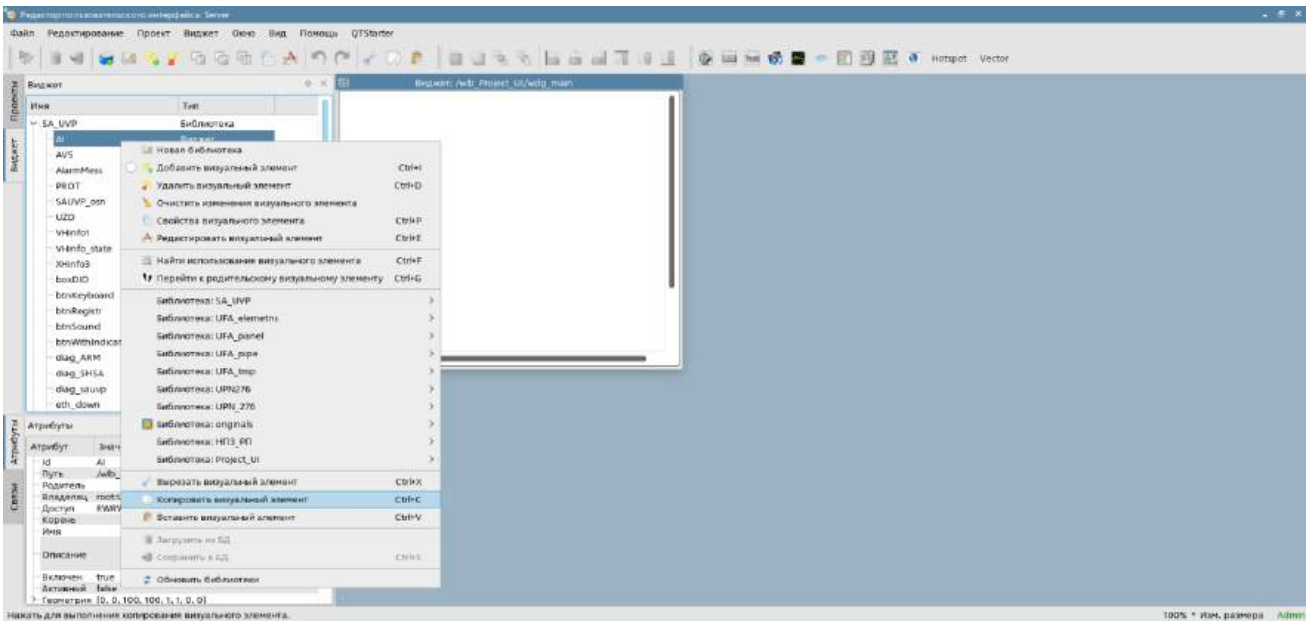


Рисунок 161

В появившемся окне (рисунок 162) изменить «ID» на «MAIN» (обязательно, для всех других элементов при копировании менять «ID» не нужно). Имя, если не будет указано, будет взято с копируемого элемента. Нажать кнопку «Принять».

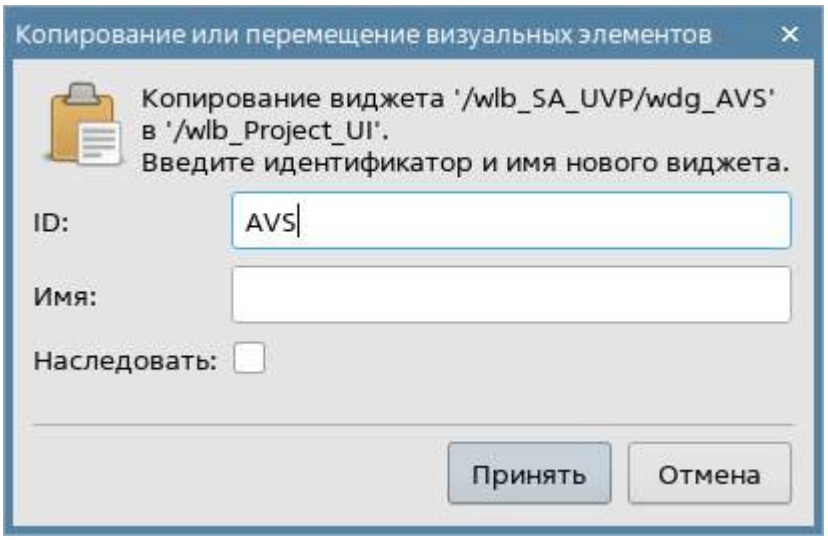


Рисунок 162

12.5.3 Создание базовых видеокадров.

В соответствии с 12.4 в библиотеке «Проект (ВК)» создадим пустой видеокадр с ID «VK» и именем «Пустой». Затем для него изменим свойства во вкладке «Атрибуты»: «Геометрия» (ширина, высота) и «Фон». Таким образом, можно настроить базовый размер видеокадра и цвет фона. Обратим внимание, что измененные поля у унаследованных виджетов подсвечиваются синим цветом для удобства отслеживания изменений и последующей их «очистки» (отката) нажатием правой кнопкой «мыши» по измененному атрибуту.

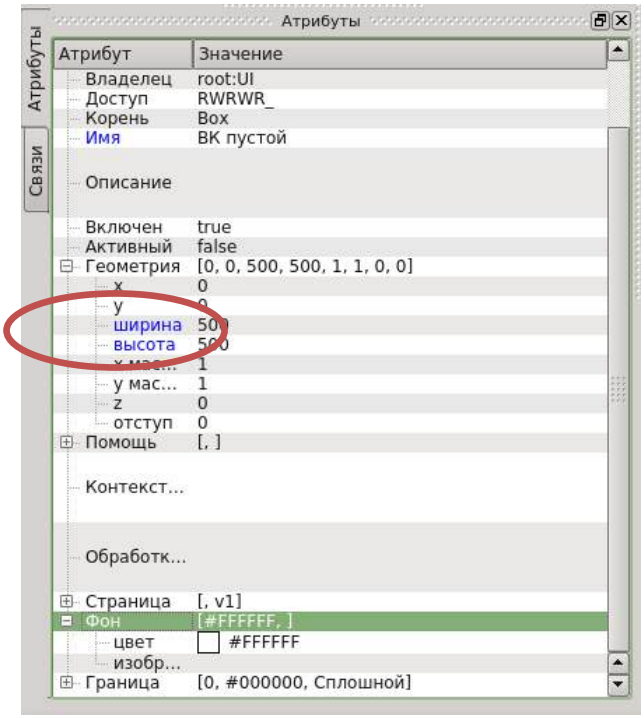


Рисунок 163

Скопировав элемент «ВК пустой» способом, аналогичным описанному в 12.5.2, в библиотеку «Проект (ВК)», указав при этом в качестве ID - «VK1», а имени - «Видеокадр1» получаем видеокадры с одинаковыми характеристиками. Создадим еще один видеокадр с ID «VK2» и именем - «Видеокадр 2». Обязательно сохранить изменения (см. 12.2.3).

12.6 Создание элементов видеокадров.

Элементами видеокадра (мнемосхемы) являются:

– статические элементы (надписи, рисунки, линии и прочие фигуры), т.е. элементы, пиктограмма которых остается неизменной;

– динамические элементы представляют собой пиктограммы, отображающие состояние механизмов, процессов, аналоговых и дискретных параметров, т.е. их состояние меняется с течением времени;

– сервисные элементы, такие как поля перехода.

Далее рассмотрим заполнение видеокадров различными элементами.

12.6.1 Добавление элементов отображения аналогового сигнала.

Добавим на «Видеокадр1» библиотеки «Проект (ВК)» элементы отображения значения аналогового параметра для четырёх сигналов. Для помещения на видеокадр элемента отображения аналогового сигнала необходимо выбрать «Видеокадр 1», а затем в меню окна выбрать пункт «Виджет» → «Библиотека: Main» → «Отобр аналог»; после чего появится курсор с образом этого элемента, который нужно подвести в желаемую область видеокадра и нажать левую кнопку мыши. В момент добавления появится диалог с запросом имени нового элемента. Добавим подобным образом пять элементов, которые назовём: «A1_Ti», «A1_To», «A2_Ti», «A2_To», «C». Добавленные элементы можно впоследствии расположить как нужно, просто выделяя и перетаскивая «мышью», либо указать координаты элемента (заполнить поля атрибута «Геометрия»). Масштабирование элемента производится с помощью одновременного нажатия левой кнопки «мыши» и клавиши “Ctrl”. Выравнивание элементов по горизонтали или вертикали, а также перемещение на более низкий уровень мнемосхемы осуществляется путем выделения элементов и нажатия на одну из кнопок панели инструментов  или из меню «Виджет» → «Вид»).

После выполнения подобных манипуляций у нас должен получиться видеокадр с видом, похожим на представленный на рисунке 164.

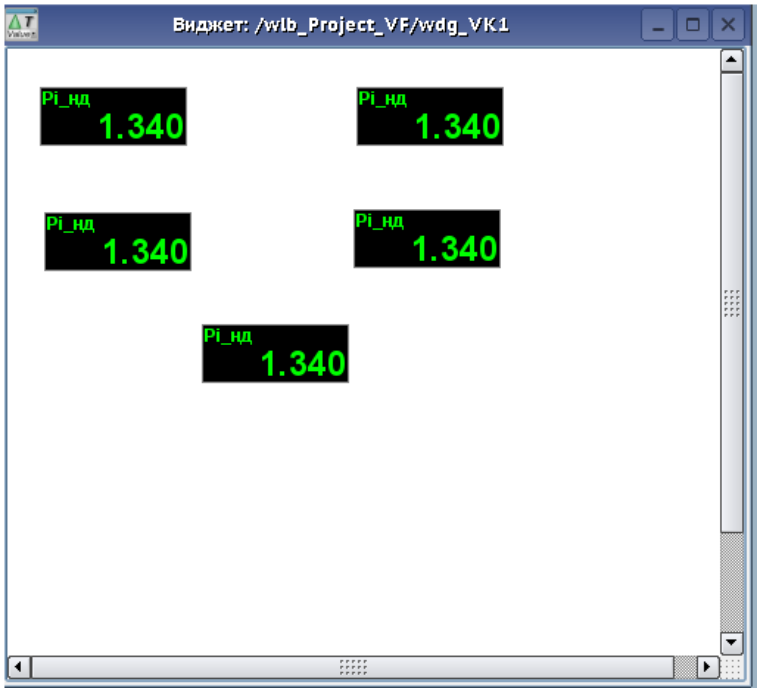


Рисунок 164

На этом процедуру создания мнемосхемы будем считать законченной. Сохраним изменения в соответствии с 12.2.3

12.6.2 Создание комплексного элемента.

Создание нового комплексного элемента, включающего в себя комбинацию различных базовых примитивов, может осуществляться в несколько этапов. В качестве примера рассмотрим задачу, создания виджета «Кран с заглушкой», изменяющего цвет в зависимости от своего состояния.

12.6.2.1 Создание видеокадра на основе примитива «Элементарная фигура».

Элемент будем создавать в созданном ранее базовом виджете «Видеокадр2». Для этого кликаем правой кнопкой «мыши» по пункту названия библиотеки «Видеокадр 2» и выбираем пункт «Библиотека: originals»→ «Элементарная фигура», как это показано на рисунке 165.

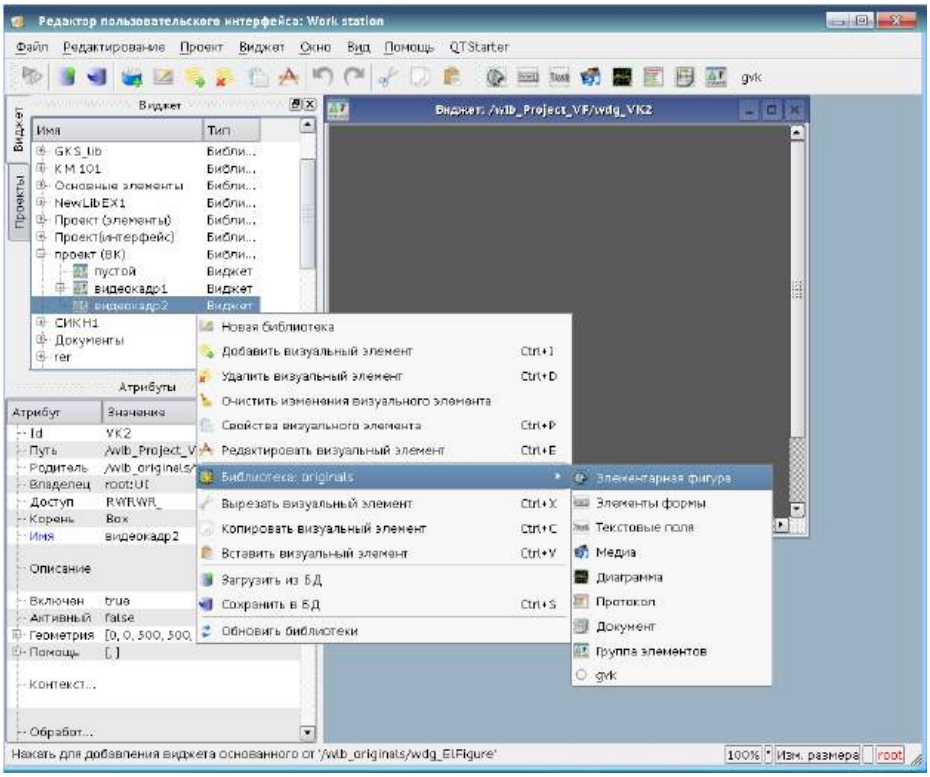


Рисунок 165

Для нового элемента указываем идентификатор «left», имя наследуется автоматически.

После подтверждения ввода появится объект нового виджета с именем «left». Выберем его в списке виджетов библиотеки «Проект (ВК)» и откроем для редактирования посредством контекстного меню нового элемента или нажав сочетание клавиш «Ctrl+P».

Теперь нарисуем визуальное представление виджета «left» (левая сторона крана). Эту процедуру можно проделать двумя нижеописанными способами:

– нарисовать желаемое изображение «мышью», 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 рисунке 166, либо двойным нажатием левой кнопки«мыши» на теле виджета;

– вручную заполнить поле «Список элементов», введя перечень необходимых элементов и координат точек.

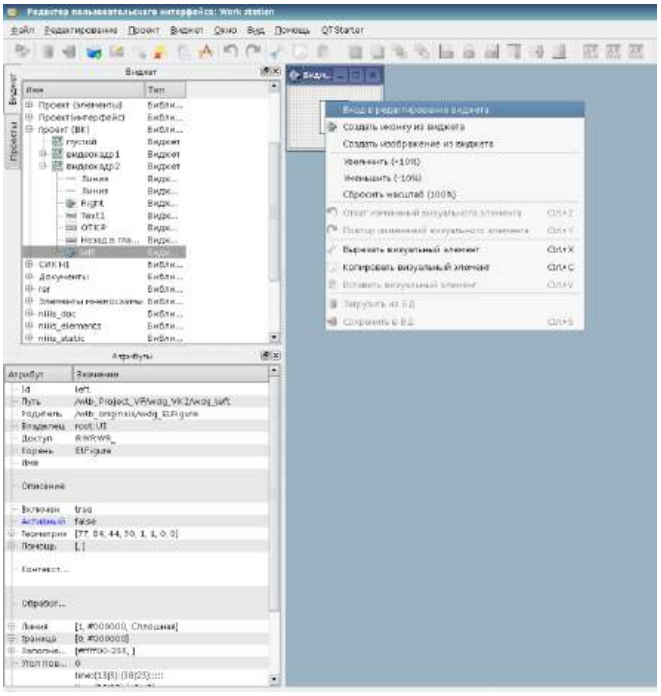


Рисунок 166

В нашем примере воспользуемся вторым способом, как это показано на рисунке 165. Для этого в поле значений атрибута «Список элементов» атрибутов введем нижеприведенный перечень и нажмем «Ctrl»+ «Enter»:

line:(20|10):(45|30)

line:(45|30):(20|50)

line: (20|50):(20|10)

fill: (20|10):(45|30):(20|50)::

где line – отрисовка линии;

fill – заполнение цветом фигуры.

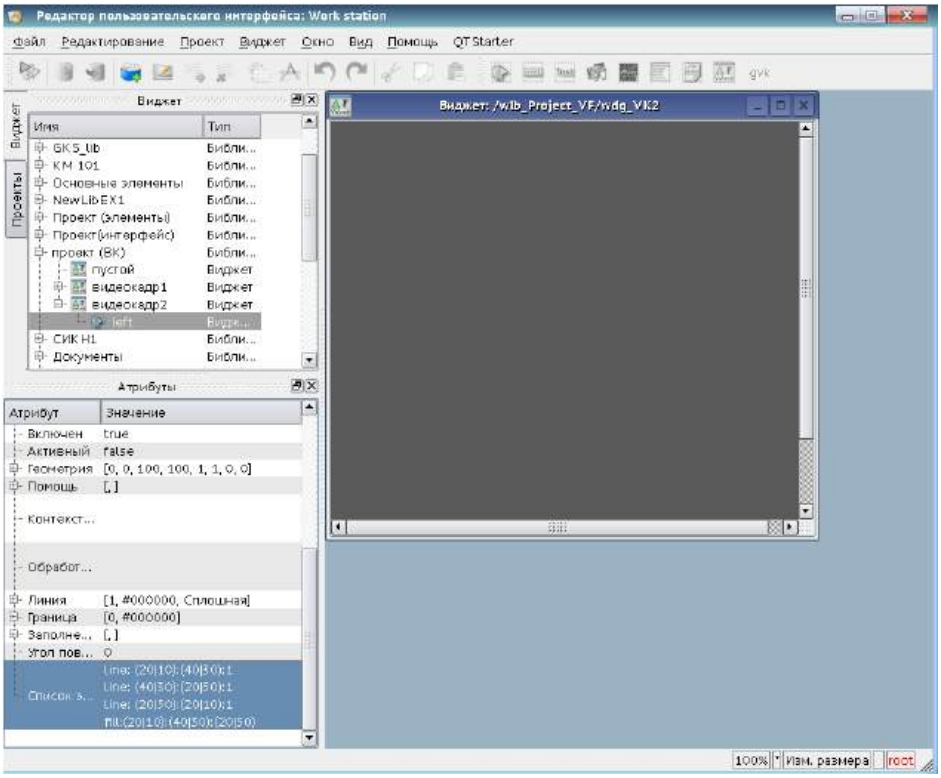


Рисунок 167

После чего установим значение атрибута «Цвет» заполнения “#ffff00”. Цвет можно задавать, как с помощью названий цветов, так и выбрав из палитры в формате #RRGGBB (#RRGGBB-AAA, где AAA - прозрачность).

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 рисунке 168.

Рисунок 168

Создадим иконку для нашего виджета, которая будет видна в дереве виджетов. Для этого в поле редактирования элемента необходимо кликнуть правой кнопкой мыши и из всплывающего меню выбрать строку «Создать иконку из виджета» (рисунок 169).

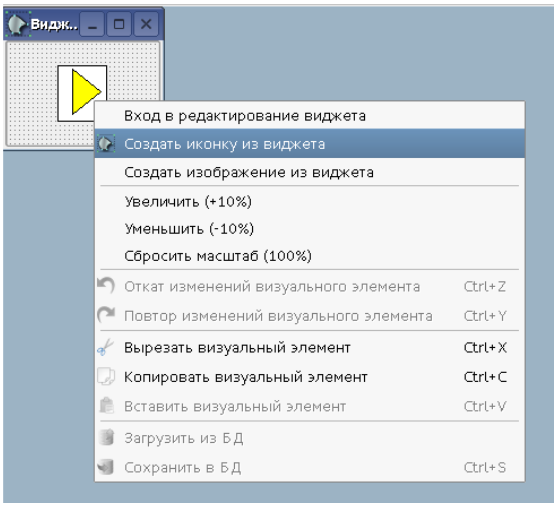


Рисунок 169

Сохраним сделанные изменения.

Аналогичным образом создадим виджет правой стороны крана. Для этого откроем в библиотеке «Проект(ВК)» - «Видеокадр 2» и создадим элемент как описано ранее или путем копирования элемента «left».

Рассмотрим второй вариант. Для этого выберем элемент «left» нашей библиотеки и нажмем сочетание клавиш «Ctrl+C», далее вставим скопированный элемент, нажав сочетание клавиш «Ctrl+V», и в поле ID зададим «right». Далее необходимо исправить значение атрибута «Угол поворота» на 180. В результате получится зеркальное отражение левой части крана (рисунок 170).

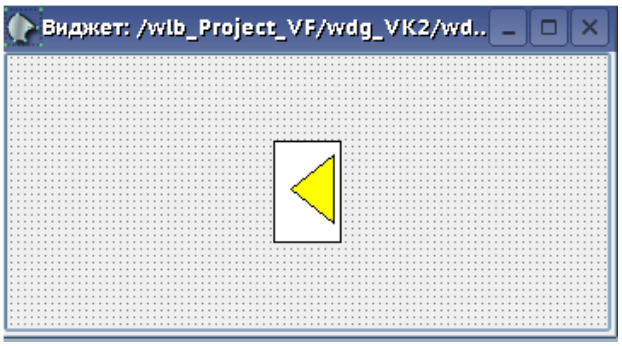


Рисунок 170

Сохраним сделанные изменения.

Далее необходимо совместить два элемента в окне виджета «VK2» (рисунок 171).

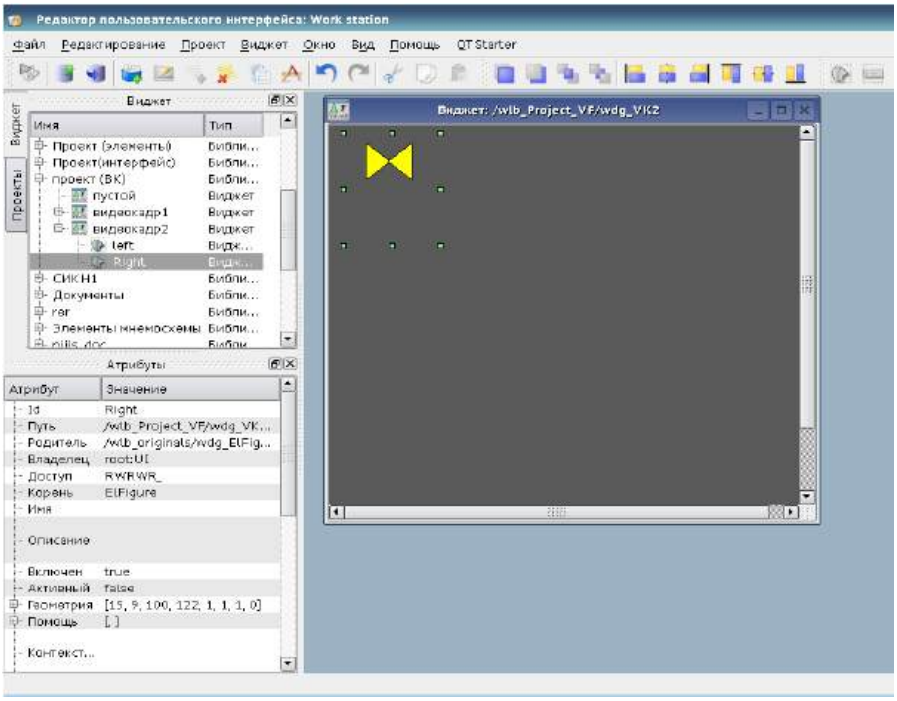


Рисунок 171

Теперь добавим на наш виджет линии. Для этого перетащим мышкой из библиотеки «Проект (элементы)» элемент Линия на наш видеокадр. Имя и ID виджета можно оставить без изменений. Установим линию на свое усмотрение. Сохраним изменения.

Далее можно добавить текстовое поле в наш виджет-контейнер «VK2», основанное на примитиве «Текст», с целью добавления названия крана. Для этого в библиотеке «Проект (ВК)» выделим виджет «Видеокадр 2» и нажмем на панели визуальных элементов на иконку примитива «Текст», как это показано на рисунке 172. В результате появится диалог ввода идентификатора и имени вновь создаваемого элемента. Согласимся с предложенным вариантом.

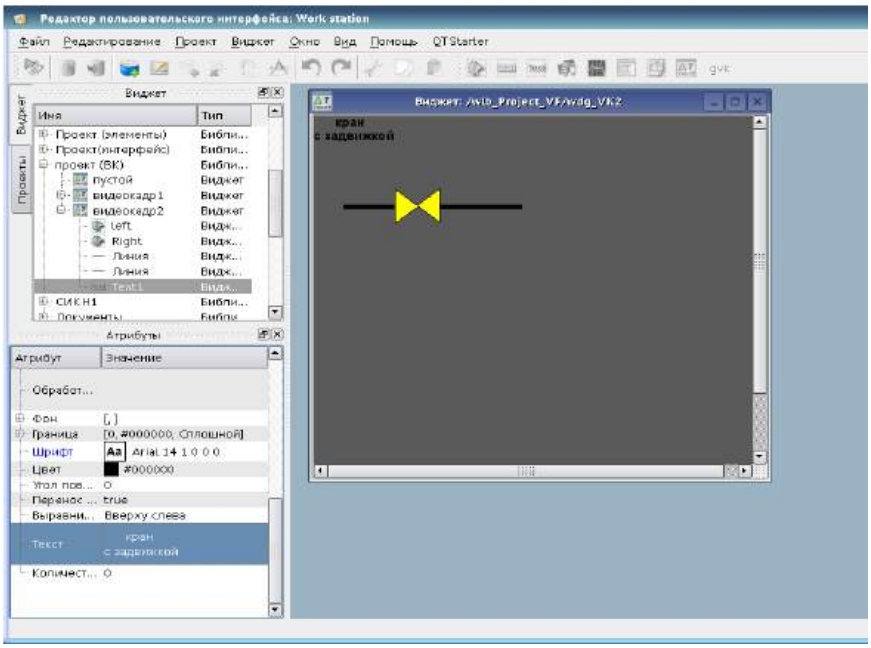


Рисунок 172

Изменим размер и установим усиление шрифта для этого элемента, нажав левой кнопкой мыши на значок ключа в поле «Шрифт» (рисунок 173). В поле значения атрибута «Текст» введем название «Кран с задвижкой».

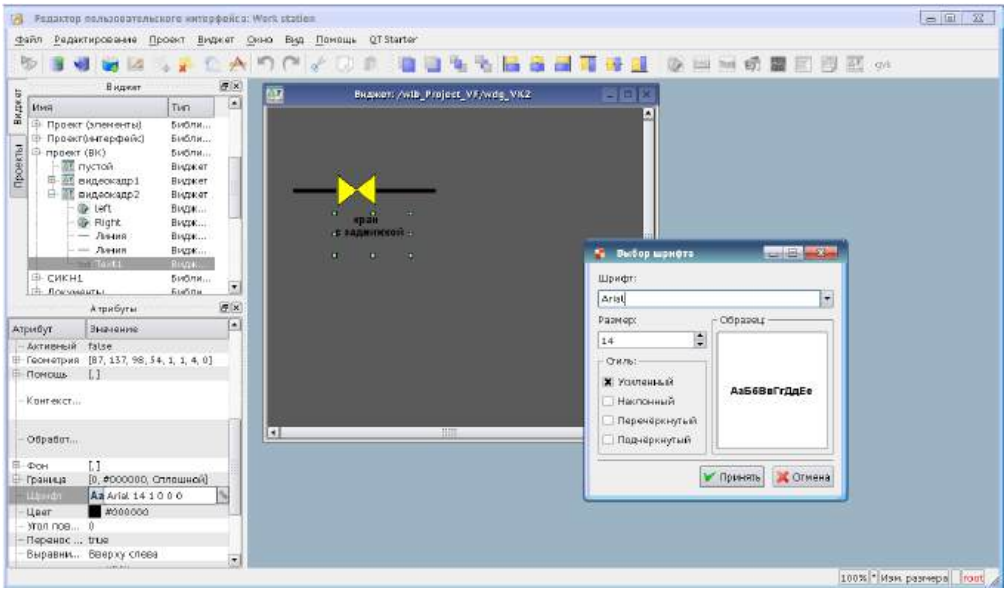


Рисунок 173

Сохраним видеокадр.

12.6.2.2 Добавление логики обработки виджета.

Для добавления логики обработки виджета «Видеокадр 2» (VK2) откроем диалог редактирования свойств этого визуального элемента, нажав сочетание клавиш «Ctrl+P», и перейдём на вкладку «Обработка». На этой вкладке мы увидим дерево атрибутов виджета и поле текста программы, для обработки атрибутов. В качестве примера будем изменять цвет заливки крана с желтого на зеленый в зависимости от его состояния: открыт/закрыт.

Для решения нашей задачи нужно добавить два атрибута: ON и OFF (рисунок 174). Для добавления атрибута необходимо развернуть корневой элемент «+», выбрать любой элемент внутри и нажать кнопку «Добавить атрибут». После добавления каждого атрибута задать для него все поля, как показано на рисунке.

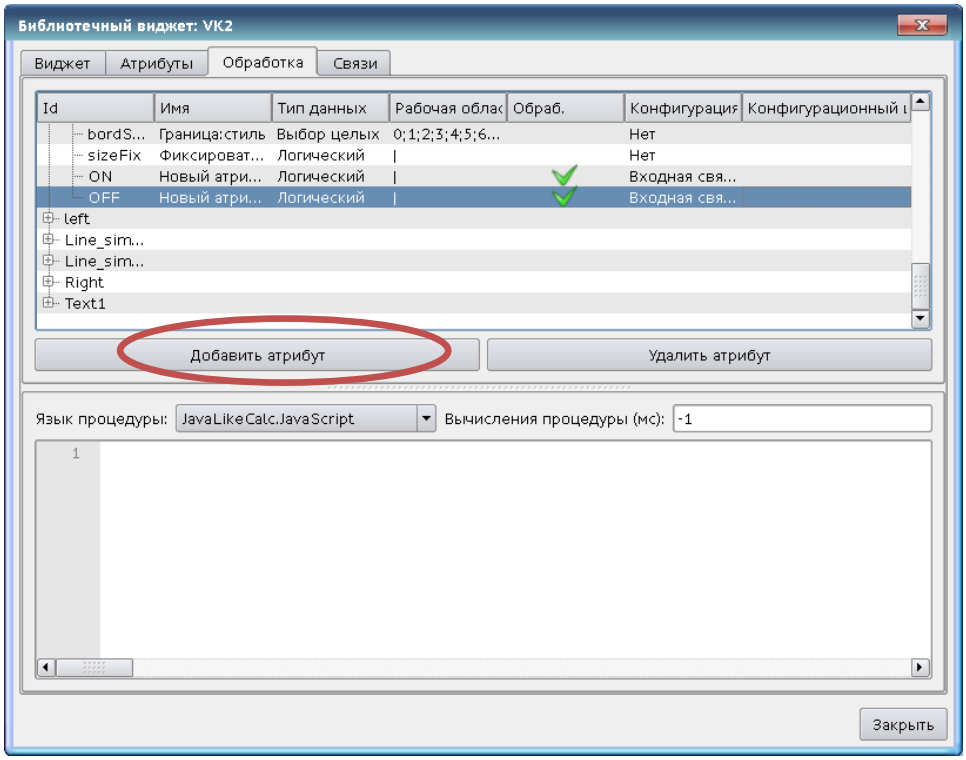


Рисунок 174

Далее для элементов «left» и «right» изменить конфигурацию для поля «FillColor» на «Постоянная» и установить флаг «true» в поле «Обработка» (рисунок 175).

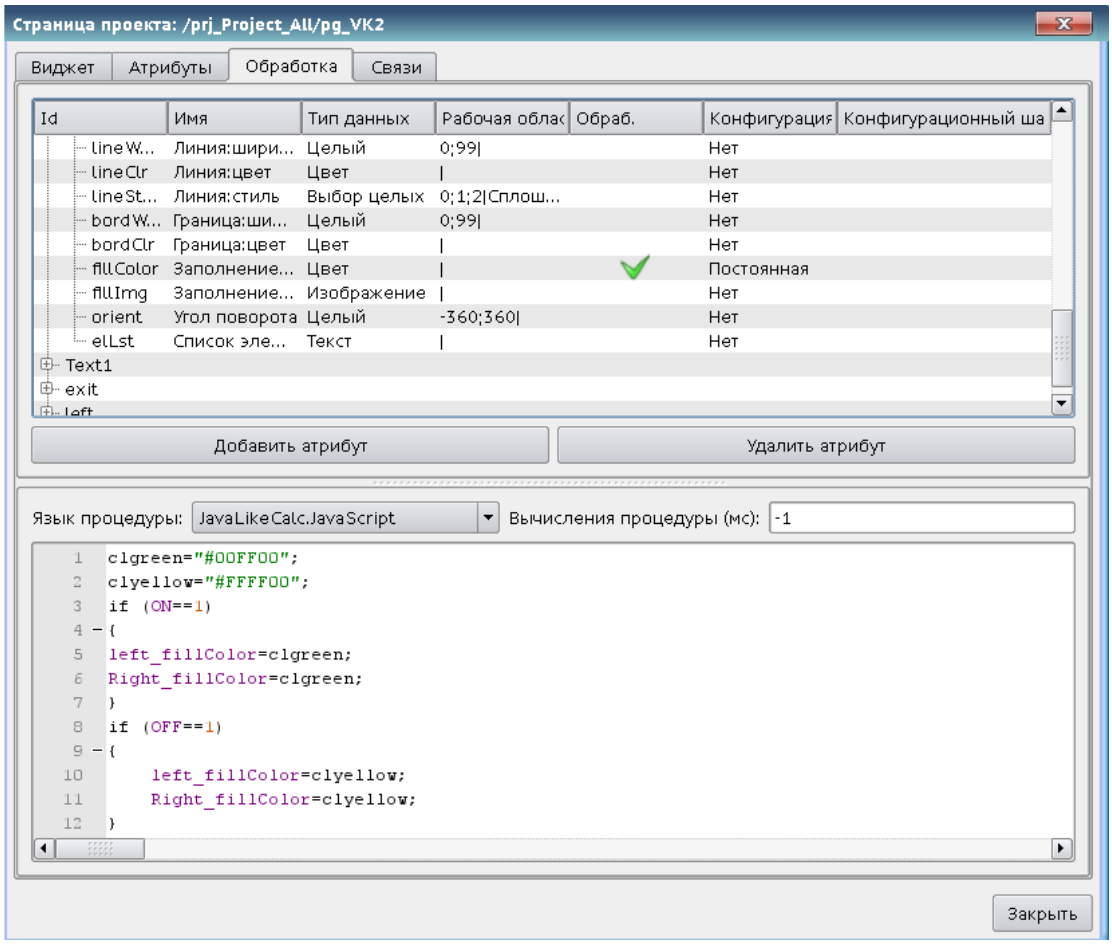


Рисунок 175

В завершение установим язык пользовательского программирования для процедуры в «JavaLikeCalc.JavaScript» и напишем программу обработки этого виджета:

```
nSingSpesial.FLibSys;
clgreen=”#00FF00”;
clyellow=”#FFFF00”;
if (ON==1)
{
  left_fillColor=clgreen;
  right_fillColor=clgreen;
}
If (OFF==1)
{
  left_fillColor=clyellow;
  right_fillColor=clyellow;
}
```

Нажмем на кнопку «Принять», после чего программа будет сохранена.

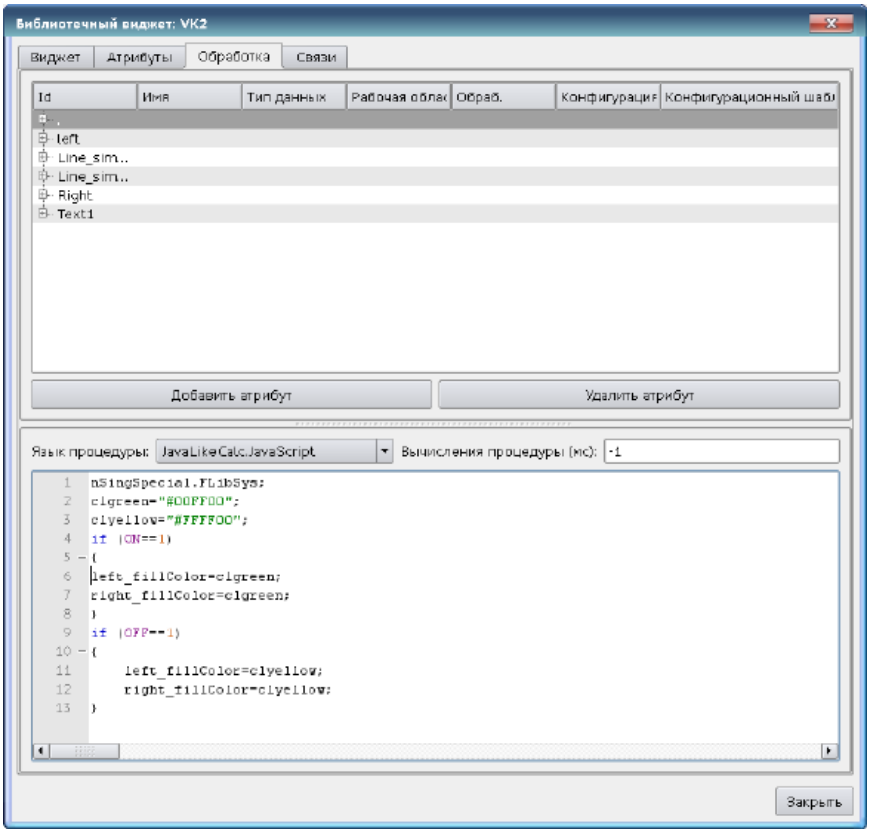


Рисунок 176

12.6.2.3 Создание сервисных кнопок

В Главном окне интерфейса пользователя поместим сервисные кнопки для перехода по мнемосхемам, созданным ранее, и поле редактирования текущей даты.

Для редактирования визуального представления «Главного окна» выберем библиотеку «Проект (интерфейс)» → «Главное окно» и нажмем сочетание клавиш «Ctrl+E». В появившемся окне создадим кнопку из примитива библиотеки originals. В качестве ID укажем «VK1» (соответствует ID виджета «Видеокадр 1»), имя укажем «Видеокадр 1». Размер и положение кнопки определим произвольно. Изменим атрибуты кнопки:

тип элемента – кнопка;

активный – true;

шрифт – усиленный, 14;

обработка событий: ws_BtPress::open://pg_VK1

Сохраним произведенные изменения.

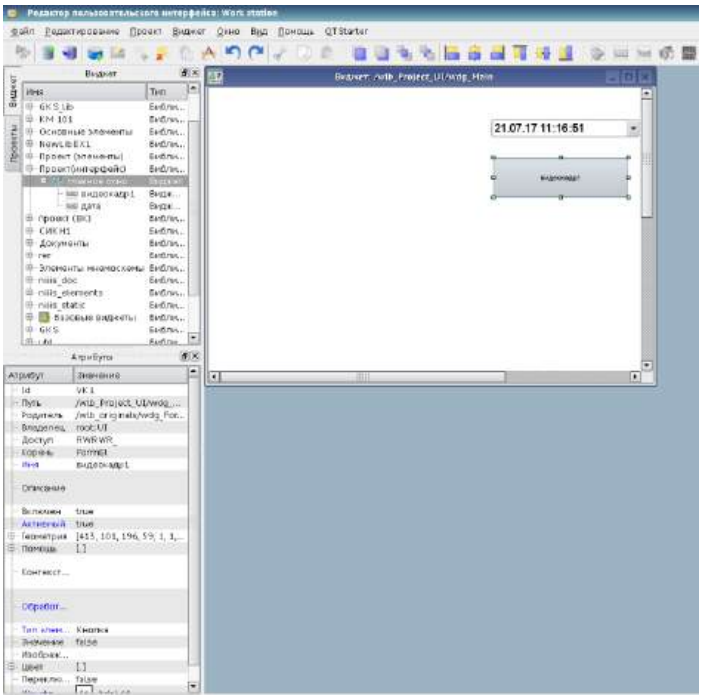


Рисунок 177

Скопируем виджет кнопки «Видеокадр 1» в библиотеку «Главное окно». Зададим ID – «VK2», имя – «Видеокадр 2». Все значения атрибутов у кнопки будут установлены как у копируемой. Необходимо будет только внести изменения в поле Обработка событий: ws_BtPress::open://pg_VK2. Сохраним изменения.

В «Главном окне» создадим еще одно поле «Текущая дата». Поле скопировано из примитива «Основные элементы», ID и имя оставлены без изменений.

На этом будем считать работу над виджетом «Главное окно» законченным, поэтому сохраним виджет и закроем окно редактирования.

Теперь создадим сервисную кнопку возврата в главное окно для мнемосхем, созданных ранее. Для этого откроем для редактирования библиотеку «Проект (ВК)» → «Видеокадр 1» и создадим кнопку с ID «Exit» и названием «Возврат в главное окно». Установим для нее атрибуты:

тип элемента – кнопка;

активный – true;

шрифт – усиленный, 13.

Сохраним изменения.

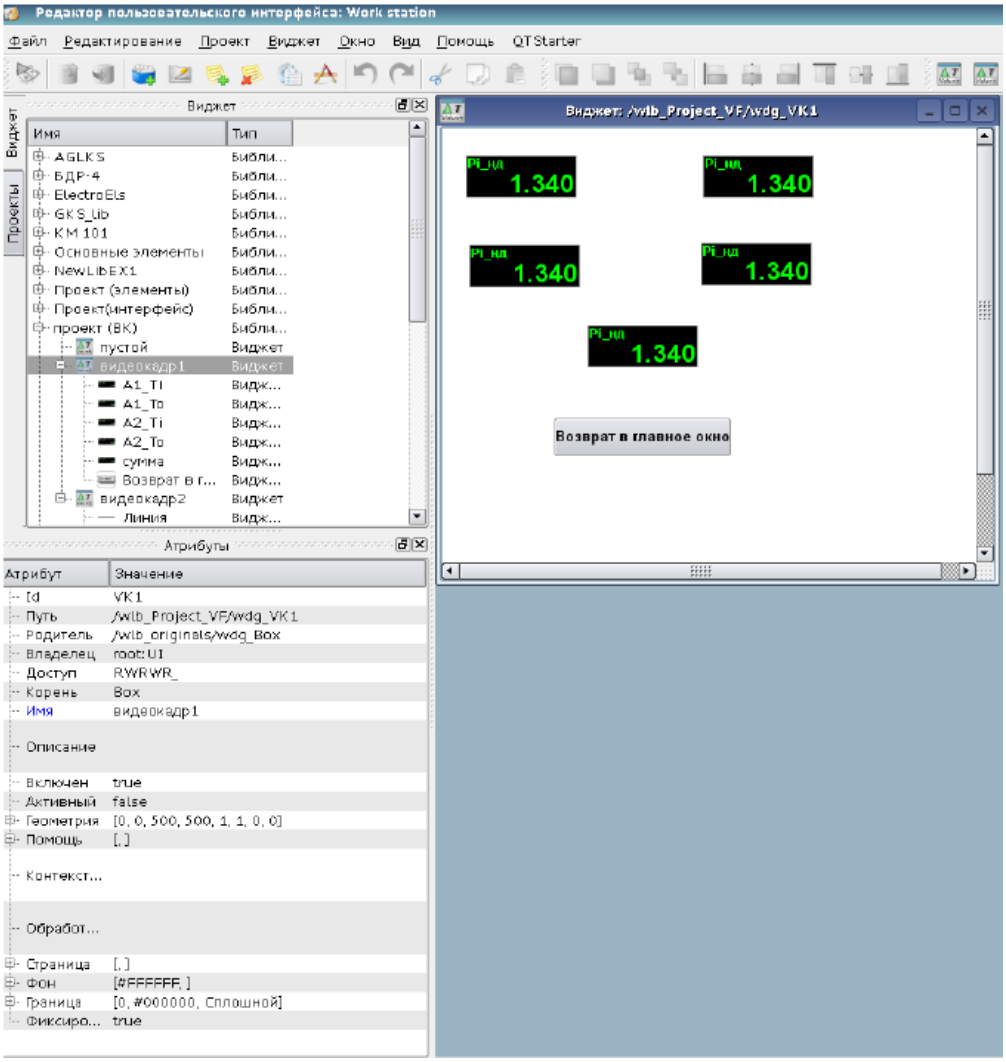


Рисунок 178

И скопируем эту кнопку на «Видеокадр 2», ID укажем такой же как и для «Видеокадр1» – «exit». Сохраним сделанные изменения.

12.7 Создание проекта

Для проверки работоспособности созданных нами видеокадров в рамках СКАДА создадим проект на базе наших библиотек элементов.

В визуальной компоненте выбрать в верхнем меню пункт «Проект» и в появившемся контекстном меню выбрать пункт «Новый проект» (рисунок 179).

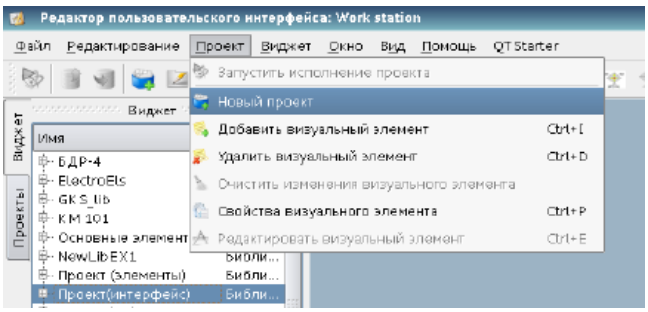


Рисунок 179

В появившемся окне указать ID (идентификатор) и имя библиотеки в соответствии с рисунком 180 и нажать кнопку «Принять».

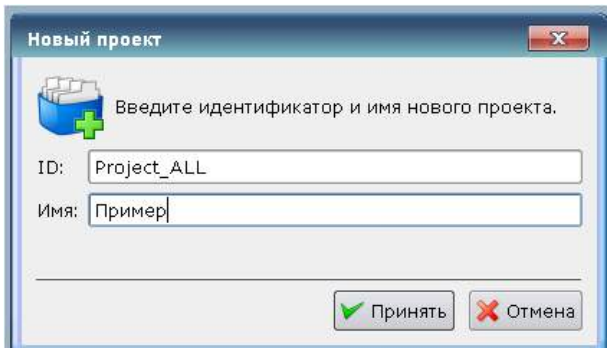


Рисунок 180

Выбрать левую вертикальную вкладку «Проекты» и в появившемся списке выбрать элемент «Проект». Открыть для редактирования свойства элемента, нажав сочетание клавиш «Ctrl+P», или выбрав из списка всплывающего меню и в появившемся окне раскрыть список «БД контейнера». В этом списке необходимо выбрать БД для хранения библиотеки визуальных элементов. Элемент списка должен иметь имя вида «PostgreSQL.<Имя БД>.<Имя библиотеки>». Где <Имя БД> соответствует ID ранее созданной БД - «PROJECT_DB», <Имя библиотеки> соответствует ID настраиваемой библиотеки «PROJECT_ALL». После выбора элемента нажать кнопку «Закрыть» (рисунок 181).

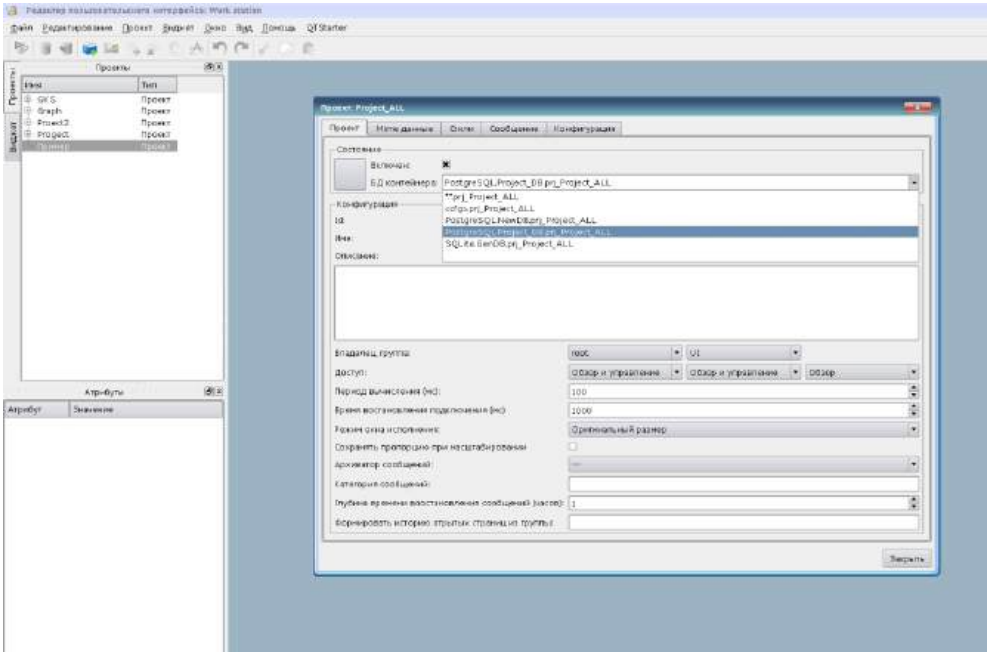


Рисунок 181

Сохранить проект в соответствии с 12.2.3.

12.7.1 Настройка окна визуального представления

Для настройки окна визуального интерфейса выбрать в меню пункт «Вид» и в появившемся списке (рисунок 182) оставить выбранными созданные нами библиотеки: «PROJECT_UI», «PROJECT_EL», «PROJECT_VF».

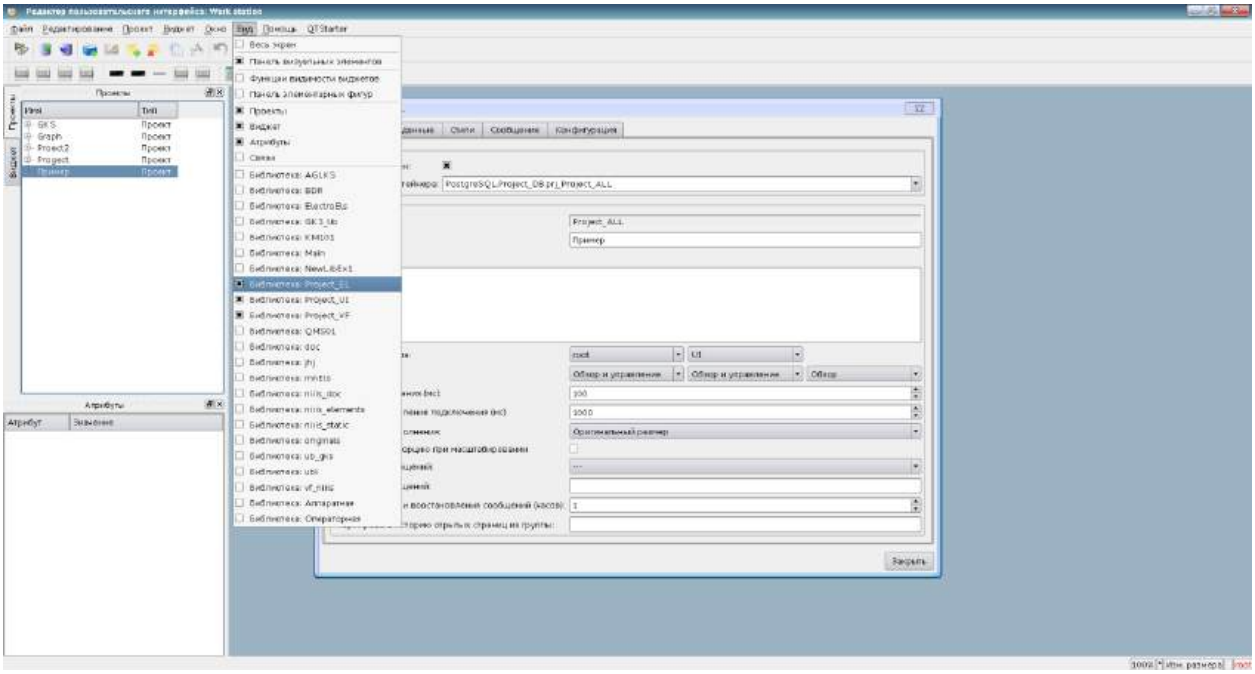


Рисунок 182

Выбрать левую вертикальную вкладку «Проекты» и в появившемся списке выбрать элемент «Проект».

12.7.2 Добавление элемента в проект.

На панели инструментов окна визуальной компоненты (рисунок 183) нажать на кнопку, добавляющую в выбранный проект элемент «Главное окно». Всплывающая подсказка данной кнопки содержит текст вида: «Добавление виджета основанного от ’/wlb_<ID_библиотеки>/wdg_<ID_элемента>’». Так как «Главное окно» имеет ID - «MAIN», а содержащая его библиотека ID - «Project_UI», то следует искать кнопку с описанием «Добавление виджета основанного от ’/wlb_Project_UI/wdg_Main’».

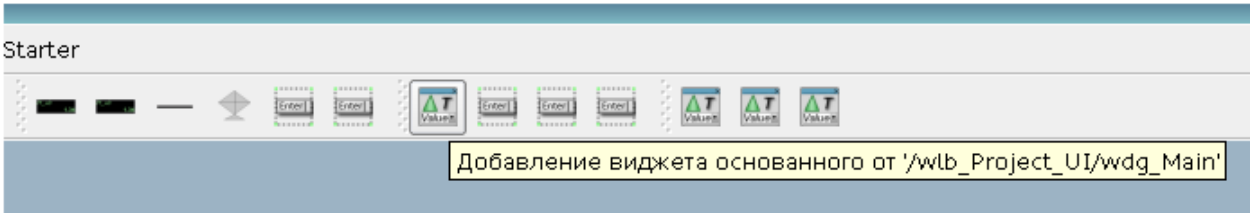


Рисунок 183

В появившемся окне задать в поле ввода ID - «MAIN», в поле «Имя» - «Главное окно».

Выбрать левую вертикальную вкладку «Проекты» и в появившемся списке выбрать элемент «Проект», затем содержащийся в нём элемент «Главное окно». Открыть свойства виджета, нажав сочетание клавиш «Ctrl+P», или из всплывающего меню. В появившемся окне поставить галочку для состояния страницы – «включен», затем раскрыть список «Тип страницы» и выбрать «Контейнер и шаблон». Тип страницы «Контейнер и шаблон» совмещает в себе функции шаблона и контейнера. После чего закрыть окно свойств.

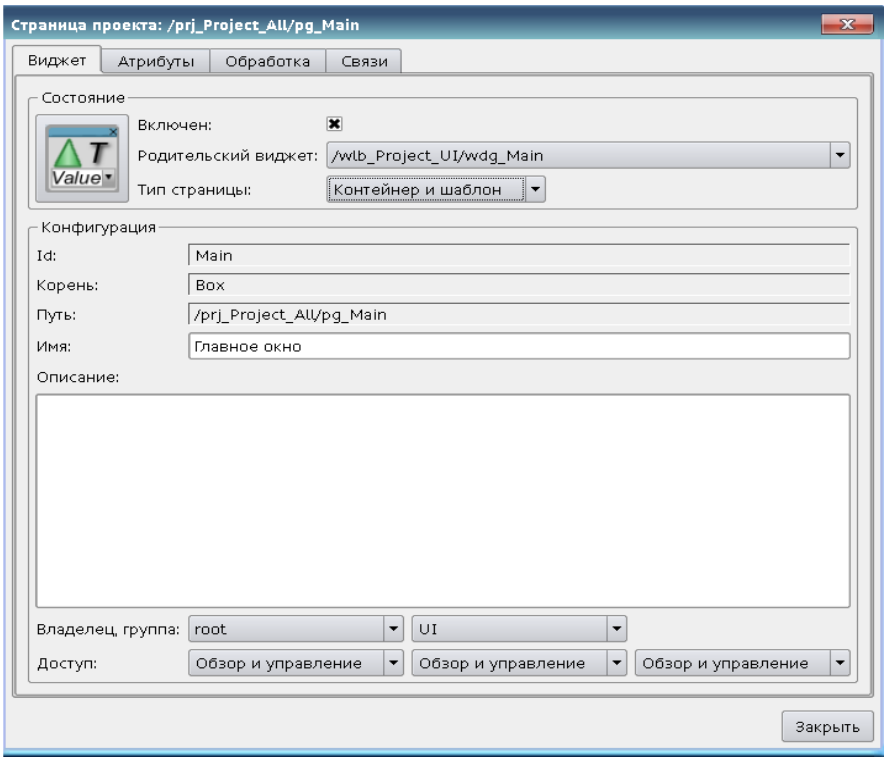


Рисунок 184

Если открыть «Главное окно» для редактирования, то мы увидим окно уже созданного нами виджета с идентичным названием в библиотеке «Проект(интерфейс)».

12.7.3 Подключение мнемосхем и источников данных к проекту.

Для подключения к Проекту наших мнемосхем необходимо в окружении Проекты выделить наш Проект и выбрать на верхней панели инструментов добавление визуального элемента, основанного от ’/wlb_Project_VF/wdg_VK1’. Создаваемому элементу присвоим ID «VK1» и имя «Видеокадр1». Точно так же добавим «Видеокадр 2», основанный на ’/wlb_Project_VF/wdg_VK2’ с ID “VK2”. Сохраним все видеокадры в проекте.

Далее поочередно открываем Видеокадры для редактирования («Ctrl+E») и выбрав кнопку «Возврат в главное окно» напишем обработку событий: ws_BtPress::open://pg_Main.

После чего сохраняем видеокадры в проекте. В результате наших действий дерево проектов имеет следующий вид (рисунок 185).

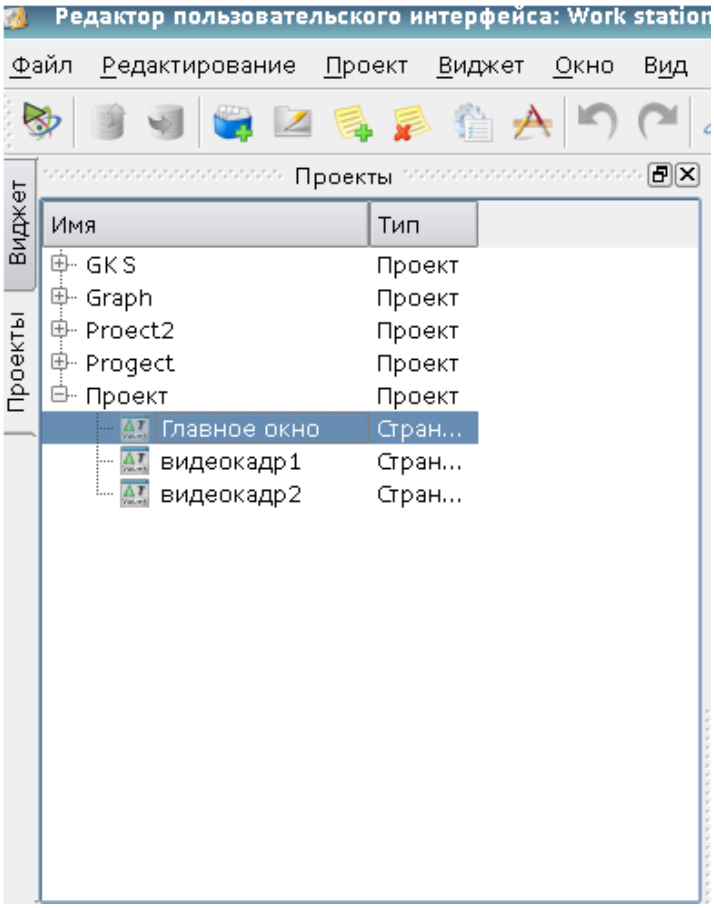


Рисунок 185

Далее для «Видеокадра1» зададим адреса параметров для отображения в виде аналогового значения. Для этого, находясь в проекте, откроем свойства визуального элемента «Видеокадр 1» (Ctrl+P) и в окне Связи установим значения Параметров pName и pVal. Имена параметров (pName) определим как имена отображаемых аналоговых сигналов: A1_1, A1_2, A2_1, A2_2, C. А в полях значений параметров (pVal) укажем адреса переменных созданных в 1.3 на Логическом уровне в контроллере Primer. Например: «prm:/LogicLev/primer/A1_T1/const» (при этом строка значений составляется путем последовательного выбора из доступных элементов всплывающего меню). Установленные параметры показаны на рисунке 186.

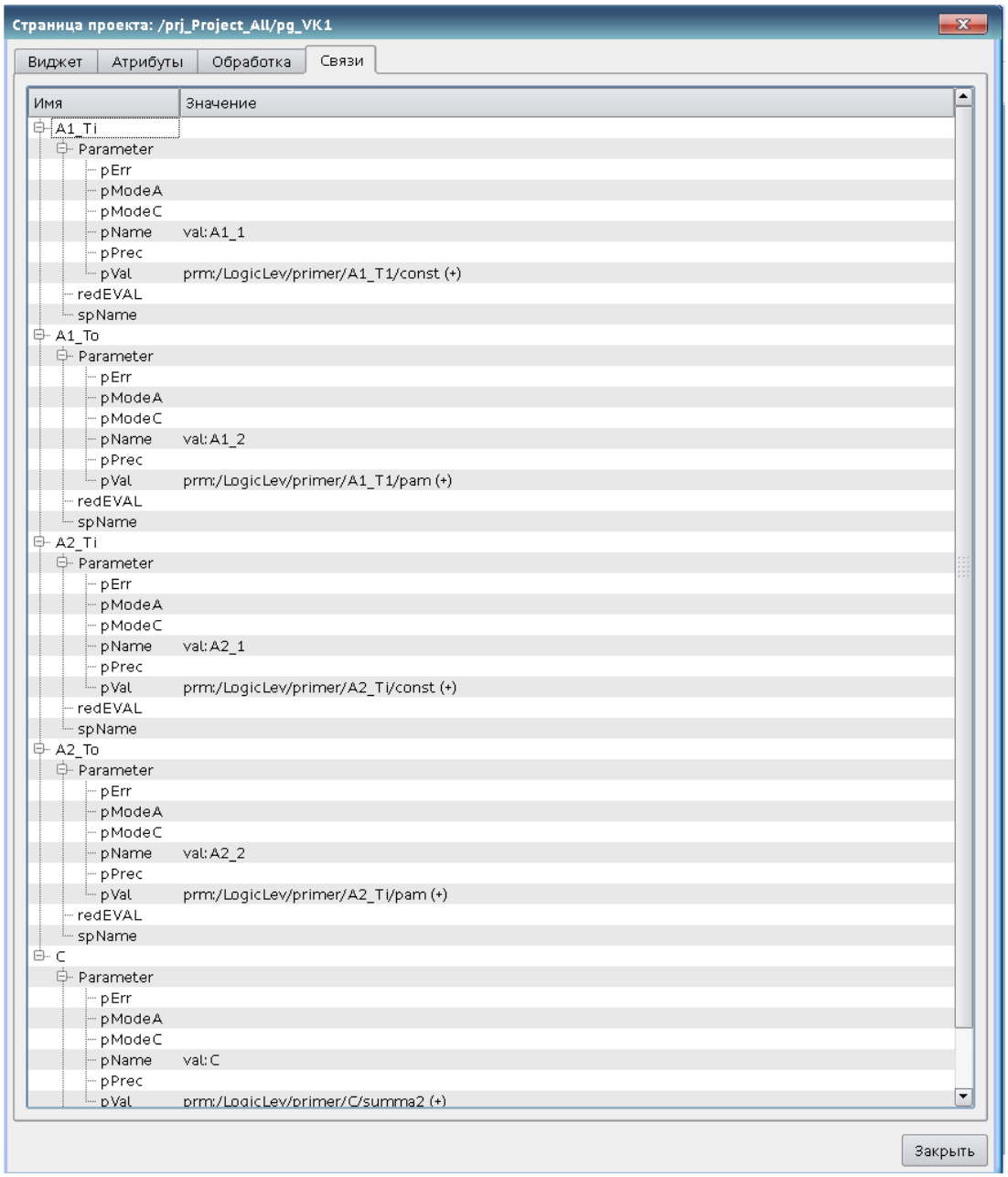


Рисунок 186

Закроем окно диалога свойств, сохраним нашу мнемосхему и проверим, что получилось.

12.7.4 Исполнение проекта

Для проверки работы нашего проекта запустим его на исполнение, нажав на иконку  .

При безошибочной конфигурации на экране появится окно, в котором можно установить дату, и две кнопки-перехода, при нажатии на которые отображаются 2 мнемосхемы (рисунок 187). Значения, отображаемые на Видеокадре 1 соответствуют значениям параметров контроллера «Primer» Логического уровня (рисунок 188).

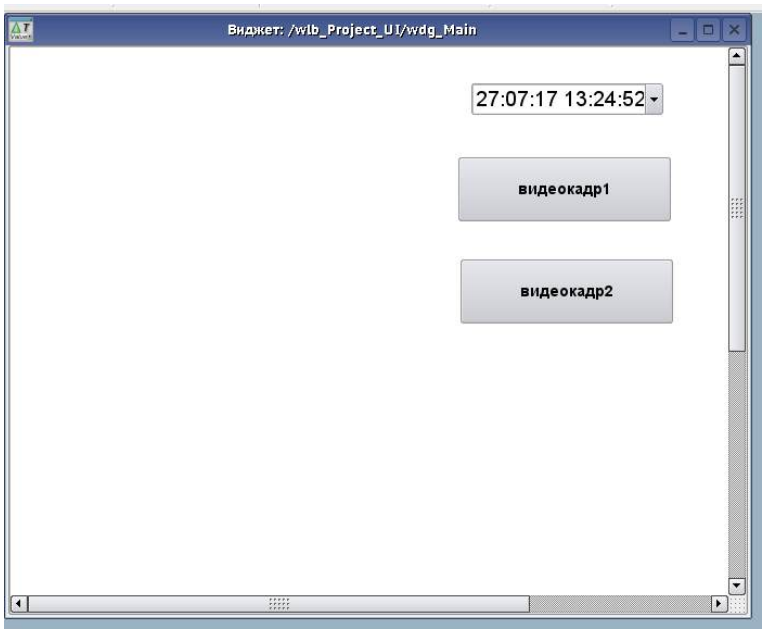


Рисунок 187

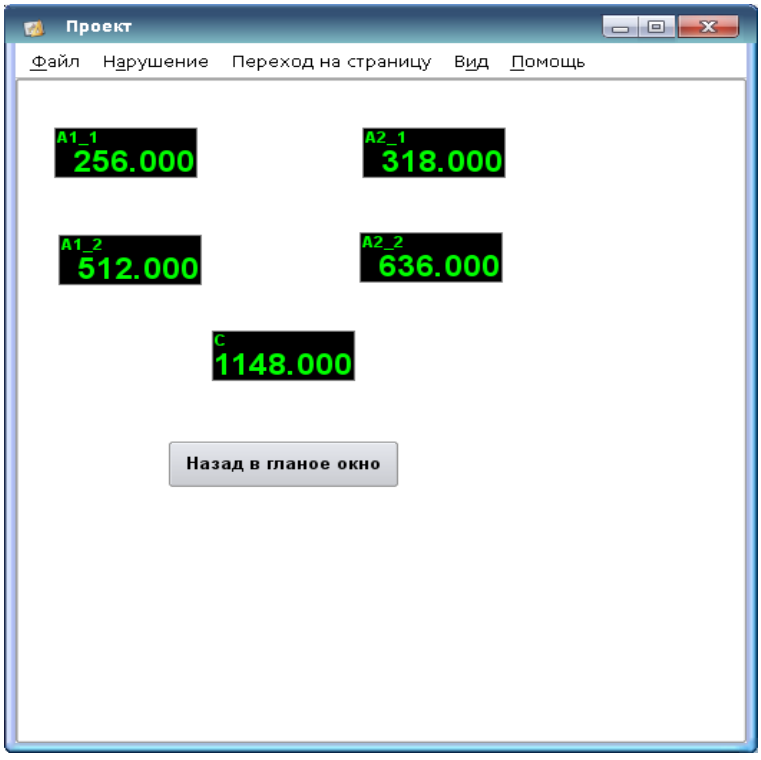


Рисунок 188

Кран на Видеокадре 2 изменяет свой цвет с желтого на зеленый при изменении значения атрибута ON на val:1 в окне «Связи» этого виджета (рисунок 189).

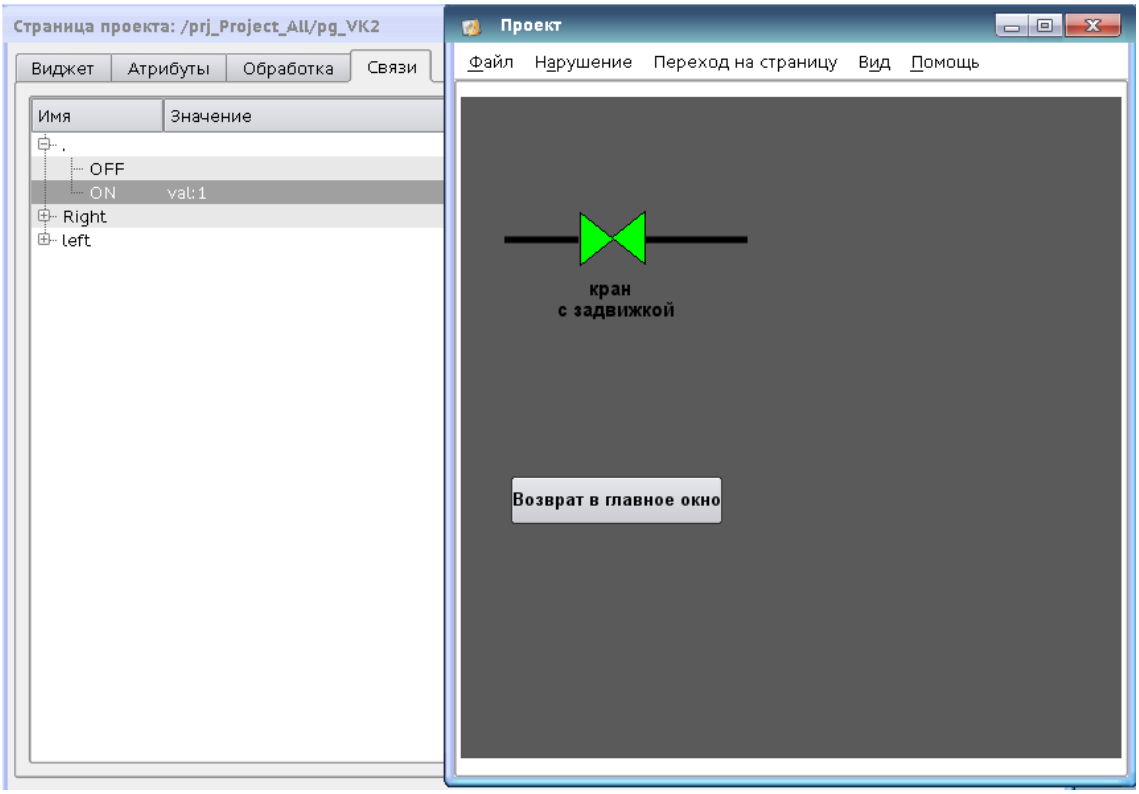


Рисунок 189

13. НАСТРОЙКА РАЗЛИЧНЫХ ТИПОВ ИСТОЧНИКОВ ДАННЫХ

13.1 Модуль источника данных ModBUS

13.1.1 Конфигурирование сбора данных по протоколу ModBUS

Создадим объект контроллера для опроса по протоколу ModBUS и получим эти данные, тем самым фактически реализовав задачу опроса реальных данных (от настоящего внешнего устройства наша конфигурация будет отличаться адресом устройства, адресами регистров ModBUS и возможно интерфейсом взаимодействия).

Для добавления нового контроллера необходимо открыть в конфигураторе страницу модуля «Сбор данных» → «Модуль» → «ModBUS» и в контекстном меню пункта "ModBUS" нажать "Добавить" (рисунок 190).

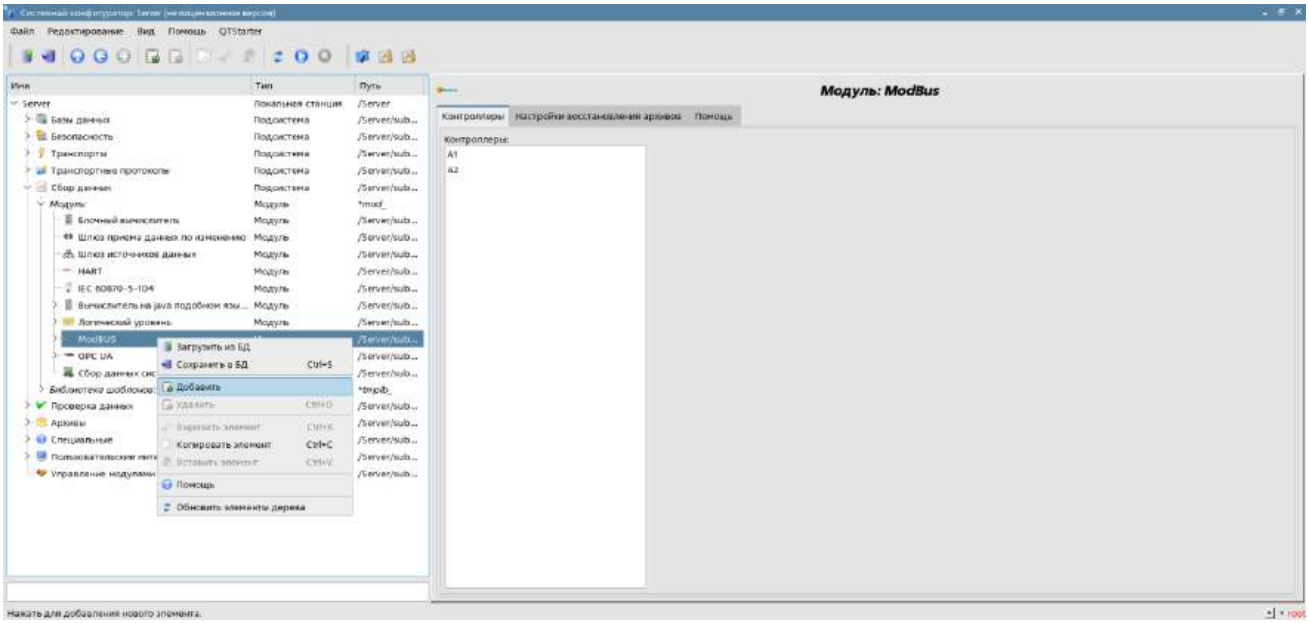


Рисунок 190

В результате появится окно диалога (рисунок 191) с предложением ввести идентификатор и имя нового контроллера (для написания имени и идентификатора действуют ранее описанные правила). Введем идентификатор "primer" и имя "primer".

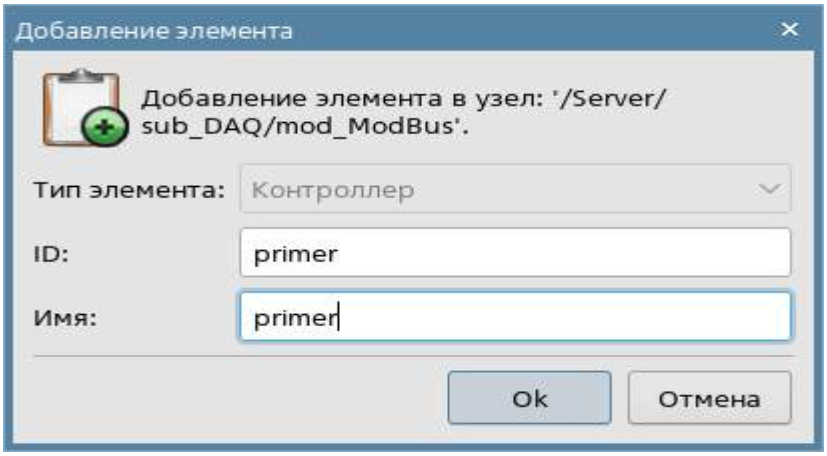


Рисунок 191

После подтверждения появится объект нового контроллера. Выбрав который можно провести его настройку.

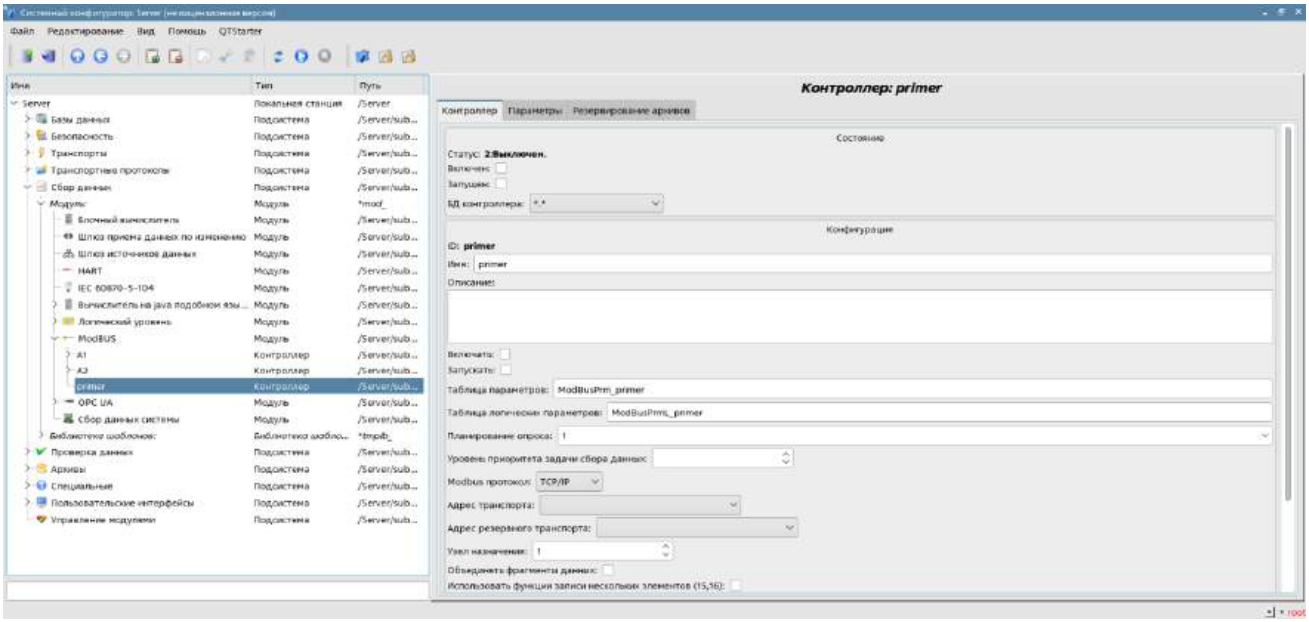


Рисунок 192

Перед конфигурацией связи со своим контроллером предварительно необходимо, из документации на контроллер, выяснить настройки его сетевых интерфейсов и протоколов, а также получить таблицу назначения внешних и внутренних сигналов контроллера на номера регистров "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 контроллера можно изменить в данном окне, а изменение идентификатора возможно только путем вырезания (Ctrl+X) и вставки (Ctrl+V) объекта с его переименованием.

– "Описание" может содержать развёрнутую характеристику и назначение объекта контроллера.

– "Включать" и "Запускать" указывают на состояние в которое следует переводить объект контроллера при запуске СКАДА. Установим оба поля.

– "Таблица параметров" — содержит имя таблицы БД, в которой будет храниться конфигурация параметров данного контроллера. Оставим по умолчанию.

– «Таблица логических параметров» - содержит имя таблицы БД, в которой будет храниться конфигурация логических 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.

– "Уровень приоритета задачи сбора данных" — указывает насколько приоритетная данная задача (от -1 до 99). Приоритеты выше нуля имеют смысл только при запуске СКАДА от привилегированного пользователя. Оставим это поле без изменений.

– "ModBUS протокол" — указывает на вариант протокола ModBUS. Возможны варианты протокола "TCP/IP", "RTU" и "ASCII". Выберем "TCP/IP". Варианты протоколов "RTU" и "ASCII" нужно устанавливать в случае связи с контроллером посредством последовательных интерфейсов (например,"RS-485").

– "Адрес транспорта" — указывает на исходящий транспорт подсистемы "Транспорты", который используется для соединения с контроллером. При выборе "TCP/IP" будет необходим транспорт в модуле Sockets, а при выборе “RTU”, “ASCII” и последовательного интерфейса – транспорт в модуле Serial. Создание исходящего транспорта в "Sockets" описано далее. Если транспорт уже создан, то достаточно выбрать его из выпадающего списка.

– "Адрес резервного транспорта" — позволяет выбрать резервный исходящий транспорт подсистемы "Транспорты", который будет использоваться в случае отсутствия ответа от основного транспорта.

– "Узел назначения" — указывает узел источника данных или контроллера в сети ModBUS. В нашем случае - "1".

– "Объединять фрагменты данных" — включает объединение несмежных фрагментов регистров в один блок запроса, до 100 регистров, вместо

генерации отдельных запросов. Позволяет уменьшить общее время опроса. Установим эту опцию.

– "Использовать функции записи нескольких элементов (15,16)" — вместо функций одноэлементной записи будут использованы многоэлементные. Оставим неизменным.

– "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.

– "Асинхронная запись" — установленный флаг реализует асинхронный способ передачи данных;

– "Время восстановления" — указывает на время в секундах, через которое, в случае отсутствия связи, повторять попытку восстановить соединение.

– "Максимальный размер блока запроса (байты)" — устанавливает максимальный размер блока групповых запросов регистров и битов, в байтах. Полезен для некоторых контроллеров с подобным ограничением. Оставим неизменным.

Сохраним наши изменения в БД (см. 12.2.3).

Далее необходимо создать Выходной транспорт в модуле "Sockets" «Транспорты» → «Сокеты» посредством контекстного меню (рисунок 193) с названием контроллера: "primer" и именем "primer".

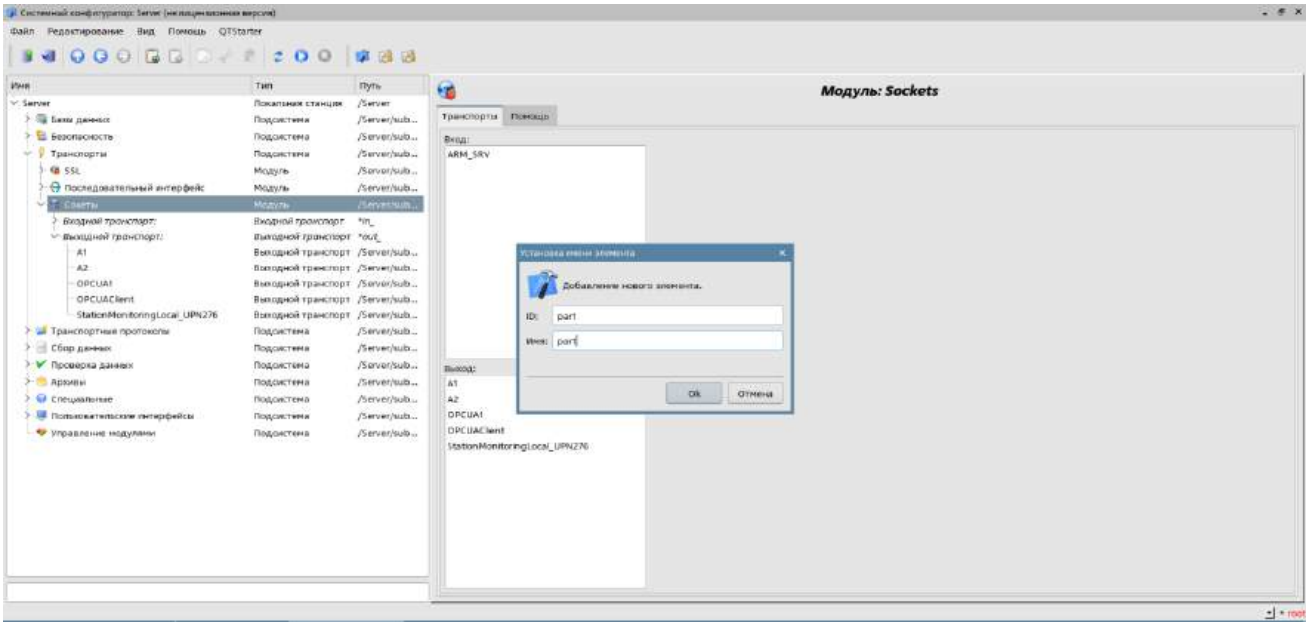


Рисунок 193

Страница конфигурации полученного исходящего транспорта приведена на рисунке 194. 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.

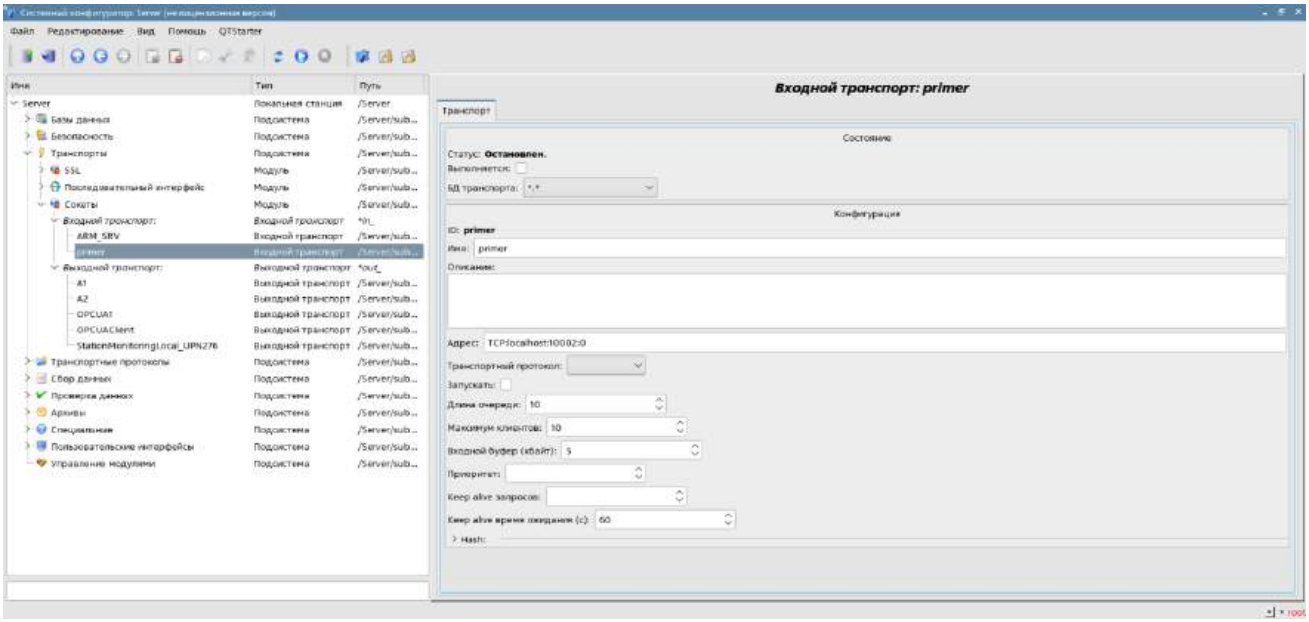


Рисунок 194

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 СКАДА.

– "Временные интервалы" — указывают продолжительность ожидания ответа от удалённой станции. Ознакомиться с форматом записи можно из всплывающей подсказки. Оставим значение неизменным.

Транспорты других типов создаются аналогичным образом, а их конфигурация их отличается обычно только форматом записи адреса и таймаутов. Для транспорта модуля "Serial" в поле адреса указывается путь к последовательному устройству,

скорость, и формат. Для переходников *USB →Serial* адрес нужно узнать в операционной системе, например, консольной командой "$ dmesg", сразу после подключения переходника.

Сохраним объект транспорта и вернёмся к конфигурационному полю "Адрес транспорта" объекта контроллера, где выберем адрес "Sockets.primer". На этом настройка объекта контроллера закончена, включим его: установив флаг "Включен".

Объект "Параметр" контроллера позволяет описать перечень данных, получаемых у контролера и передать их в окружение СКАДА.

Для добавления нового объекта параметра контроллера "primer" необходимо открыть в конфигураторе страницу контроллера и нажав правую кнопку мыши выбрать пункт «Добавить». Для нового объекта ввести идентификатор "par1" и имя "par1".

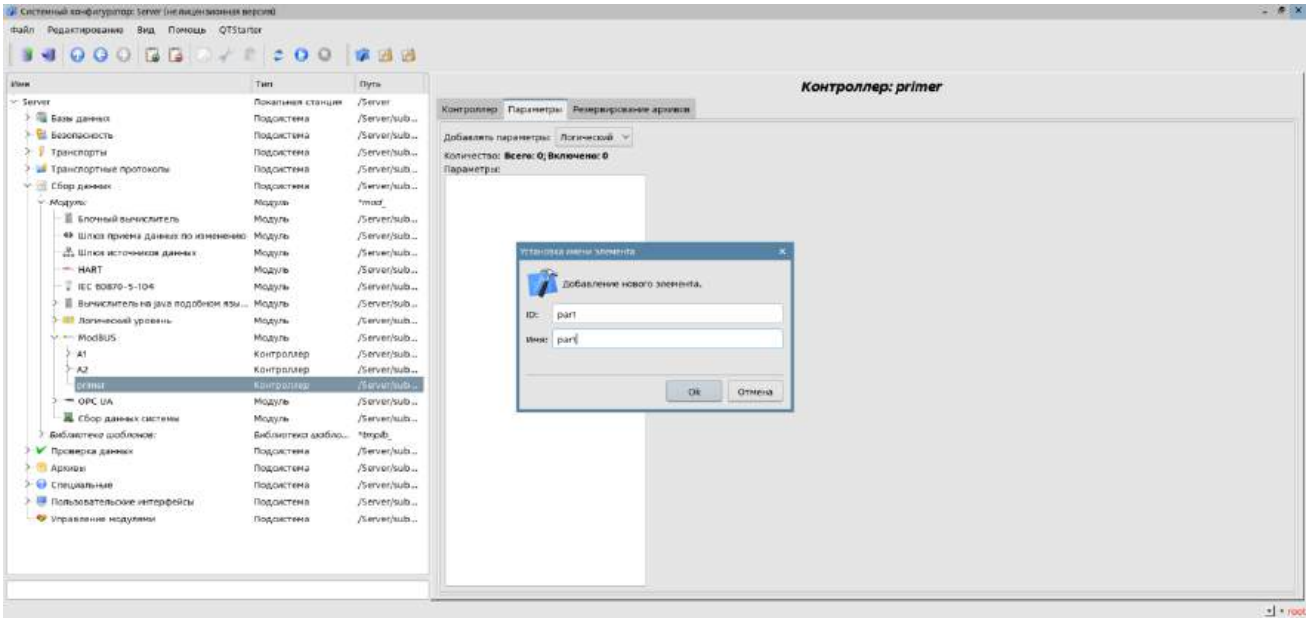


Рисунок 195

Страница конфигурации полученного параметра приведена на рисунке 196. Эта страница содержит раздел состояния и оперативного управления. В поле "Тип" содержится идентификатор типа параметра, в нашем случае тип "Стандартный" (std). Параметр мы можем включить, установив флажок напротив соответствующего поля. Включенный параметр участвует в процессе обмена с контроллером.

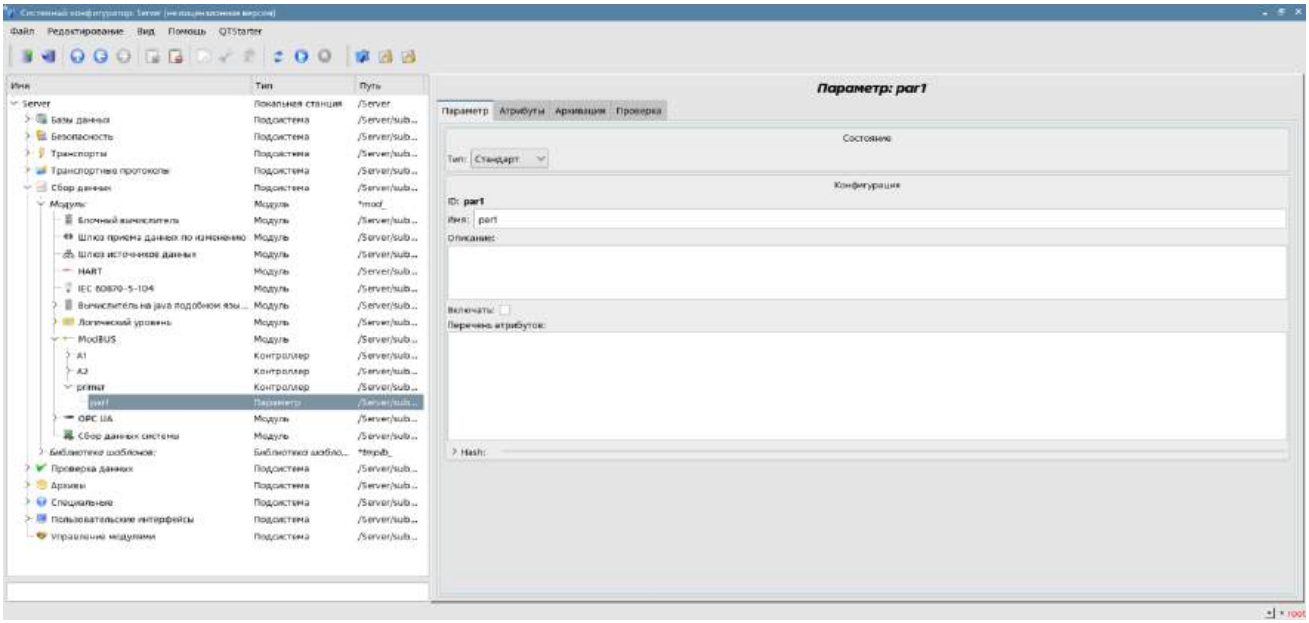


Рисунок 196

В разделе "Конфигурация" непосредственно содержится конфигурация объекта параметра:

– "Идентификатор" и "Имя" содержат названия, введенные при создании объекта.

– "Описание" — может содержать развёрнутую характеристику и назначение объекта.

– "Включать" — указывает на то, в какое состояние переводить объект при запуске СКАДА. Установим поле.

– "Перечень атрибутов" — содержит конфигурацию атрибутов параметров в соотношении их с регистрами и битами "ModBUS". Ознакомиться с форматом записи можно из всплывающей подсказки (рисунок 197).

```
Список конфигурации атрибутов. Список формируется строками в формате: "{dt}:{numb}:{rw}:{id}:{name}".
Где:
  dt - ModBus тип данных (R-регистр[3,6(16)], C-бит[1,5(15)], RI-регистр входа[4], CI-бит входа[2]);
       R и RI могут быть расширены суффиксами: i2-Int16, i4-Int32, f-Float, b5-Bit5, s-Строка;
       Начните с символа '#' для комментирования строки;
  numb - адрес ModBus устройства (десят., шестн., или восьмеричн.) [0...65535];
  rw - режим чтения/записи (r-чтение; w-запись, rw-запись и чтение);
  id - идентификатор создаваемого атрибута;
  name - имя создаваемого атрибута.
Примеры:
  "R:0x300:rw:var:Variable" - доступ к регистру;
  "C:100:rw:var1:Variable 1" - доступ к биту;
  "R_f:200:r:float:Float" - получить вещественное из регистров 200 и 201;
  "R_i4:300,400:r:int32:Int32" - получить int32 из регистров 300 и 400;
  "R_b10:25:r:rBit:Reg bit" - получить бит 10 из регистра 25;
  "R_s:15,20:r:str:Reg blk" - получить строку, блок регистров, из регистра 15 и размером 20.
```

Рисунок 197

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: "par2" с именем "par2". 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 флажок "Запущен". В результате в поле "Статус" отразится результат подключенного обмена данными.

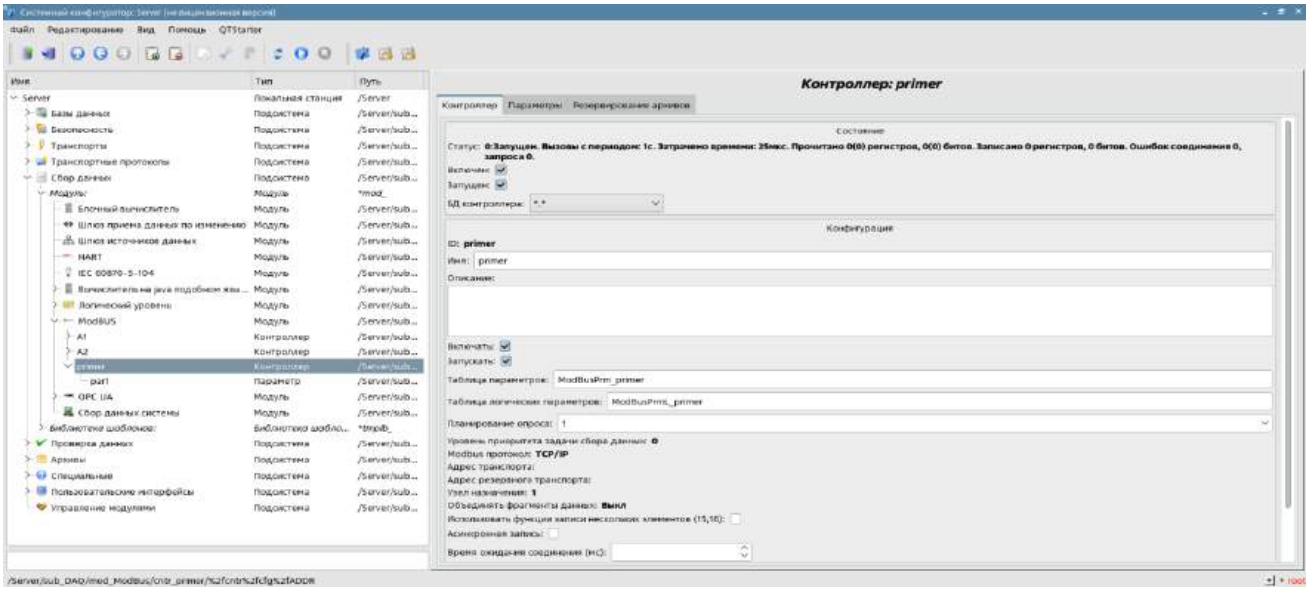


Рисунок 198

В случае успешного обмена с физическим контроллером на вкладке "Атрибуты" наших параметров отобразятся эти данные. Поскольку опрос производится регулярно и с периодичностью в секунду, то мы можем наблюдать их изменение, нажимая кнопку "Обновить текущую страницу" на панели инструментов.

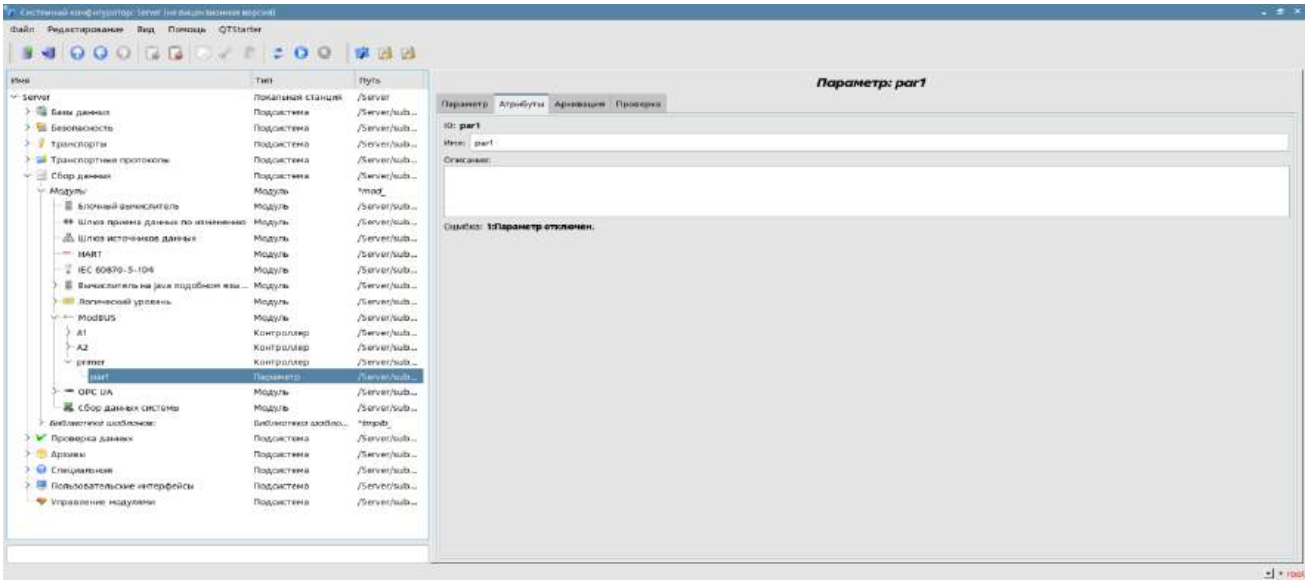


Рисунок 199

13.1.2 Конфигурирование обработки данных, полученных по протоколу ModBUS

Для возможности использования полученных в предыдущем пункте данных на «Логическом уровне» подсистемы «Сбор данных» необходимо создать объект библиотеки шаблонов ("Сбор данных"→"Библиотека шаблонов"→"ADCU") или воспользоваться уже имеющимися шаблонами, указывая адреса ModBus-регистров параметров. Далее описан первый способ.

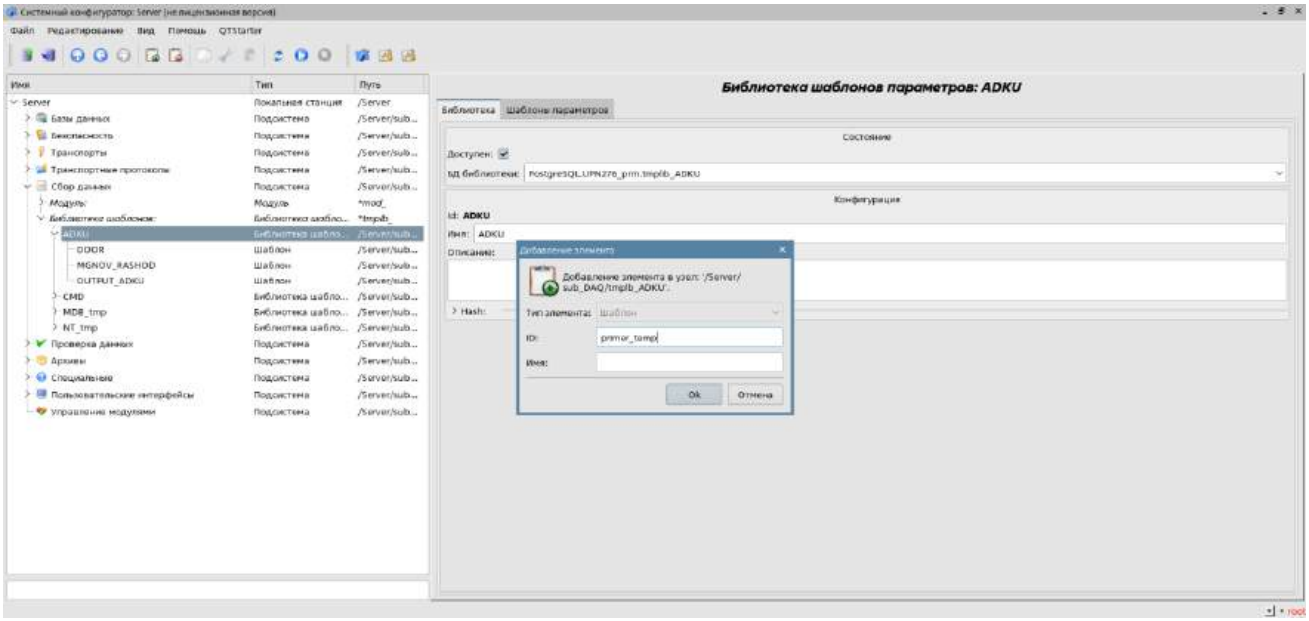


Рисунок 200

Основная конфигурация и формирование шаблона параметра сбора данных осуществляется во вкладке "IO" шаблона (рисунок 201). 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", "Cw" в "Полный доступ" для формирования трёх атрибутов: два только на чтение и один на полный доступ, у результирующего параметра сбора данных.

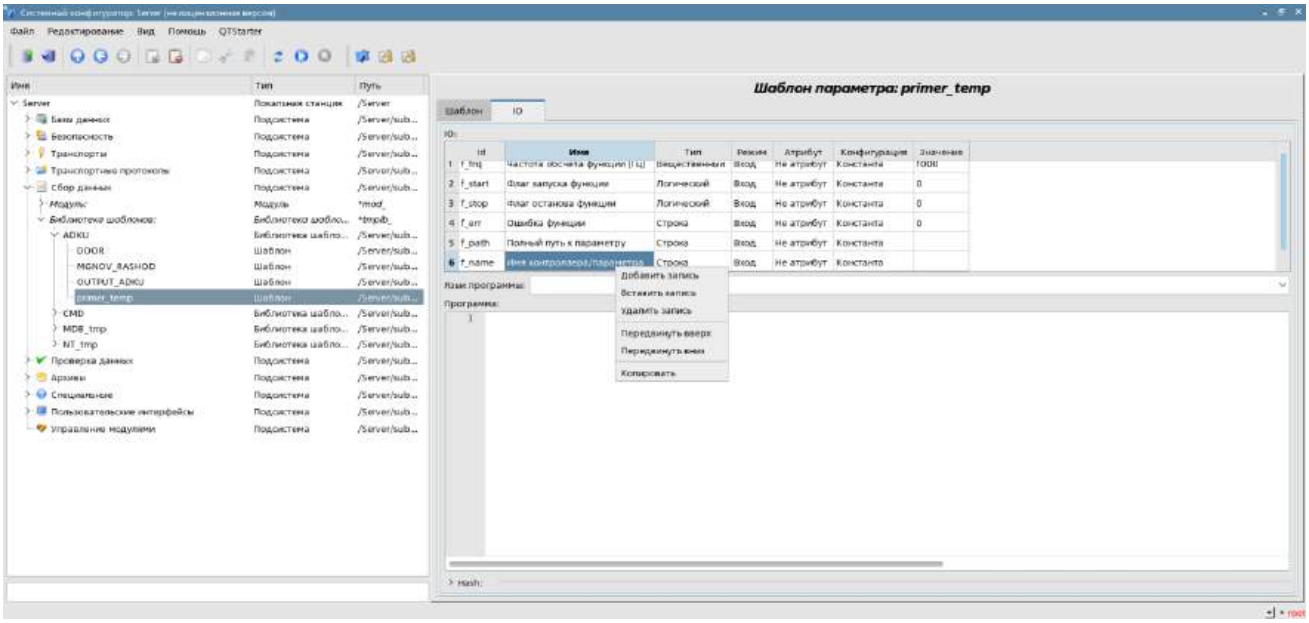


Рисунок 201

Язык программы установим в "JavaLikeCalc.JavaScript", а в поле описания программы введем следующий код:

Ti=150*TiCod/65536;

To=100*ToCod/65536;

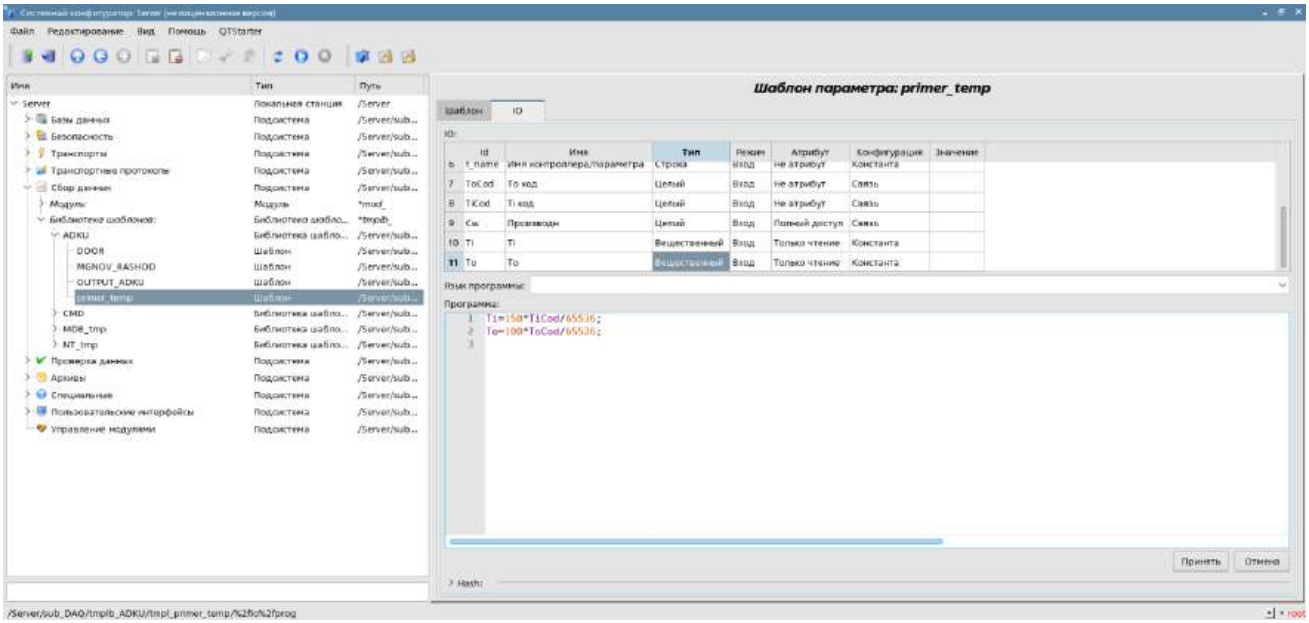


Рисунок 202

Полученный шаблон необходимо сохранить и сделать шаблон доступным, установив флажок напротив соответствующего поля. Доступные шаблоны могут подключаться к параметрам контроллеров сбора данных, а параметры будут выполнять вычисления по этому шаблону. В поле "Использовано" отображается число объектов, которые используют данный шаблон для вычисления образа параметра.

Далее необходимо создать контроллер на «Логическом уровне» и связать его параметры с параметрами шаблона (последовательный выбор типа источника, элемента выбора, параметра, атрибута – алгоритм описан в 12.3).

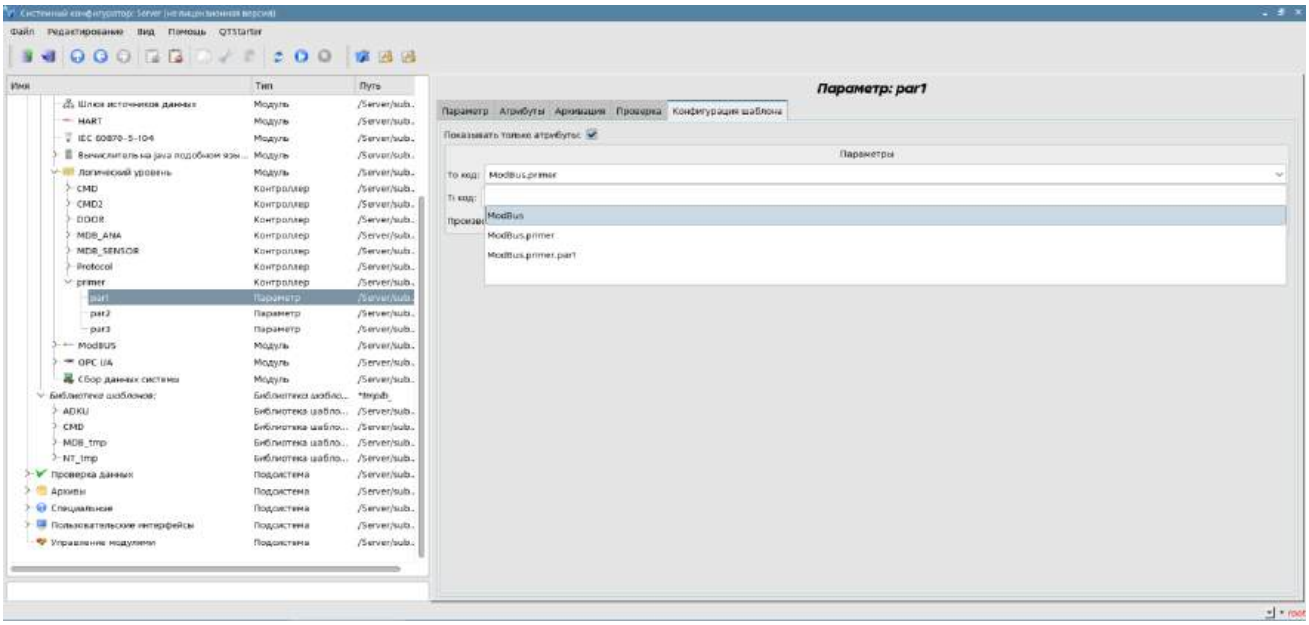


Рисунок 203

После сохранения сделанных изменений, включения и запуска контроллера в поле «Статус» раздела «Состояние» отразится результат обработки данных (рисунок 204), а у параметров контроллера отразятся вычисленные значения атрибутов (рисунок 205).

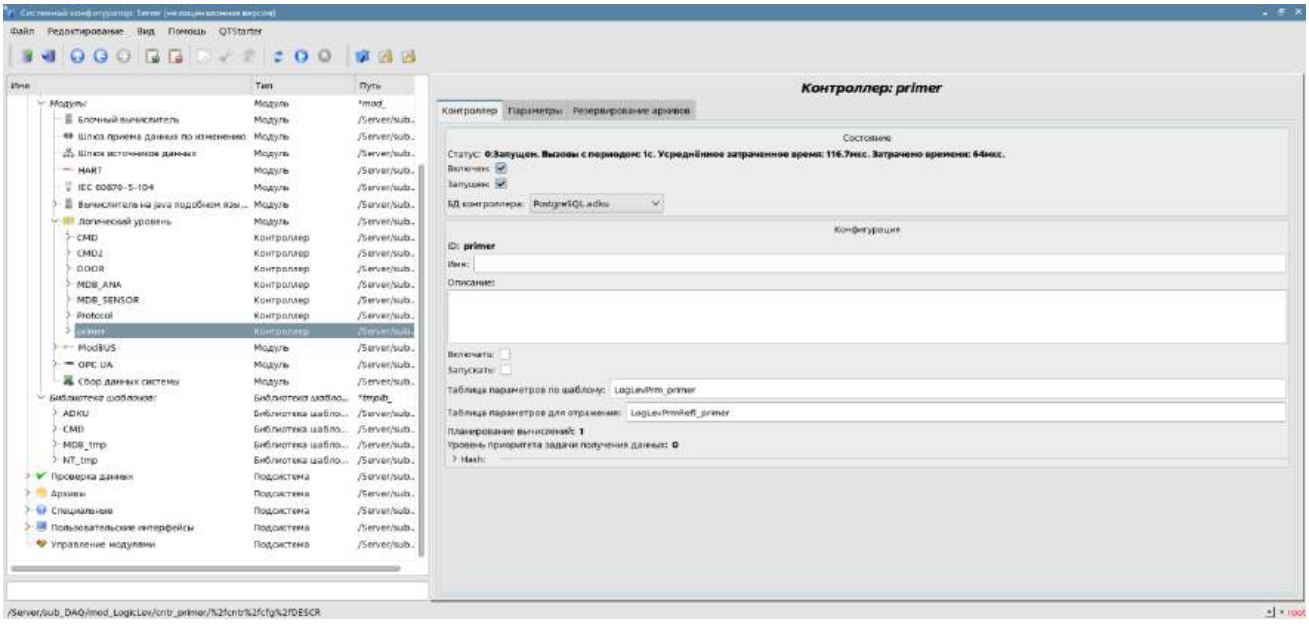


Рисунок 204

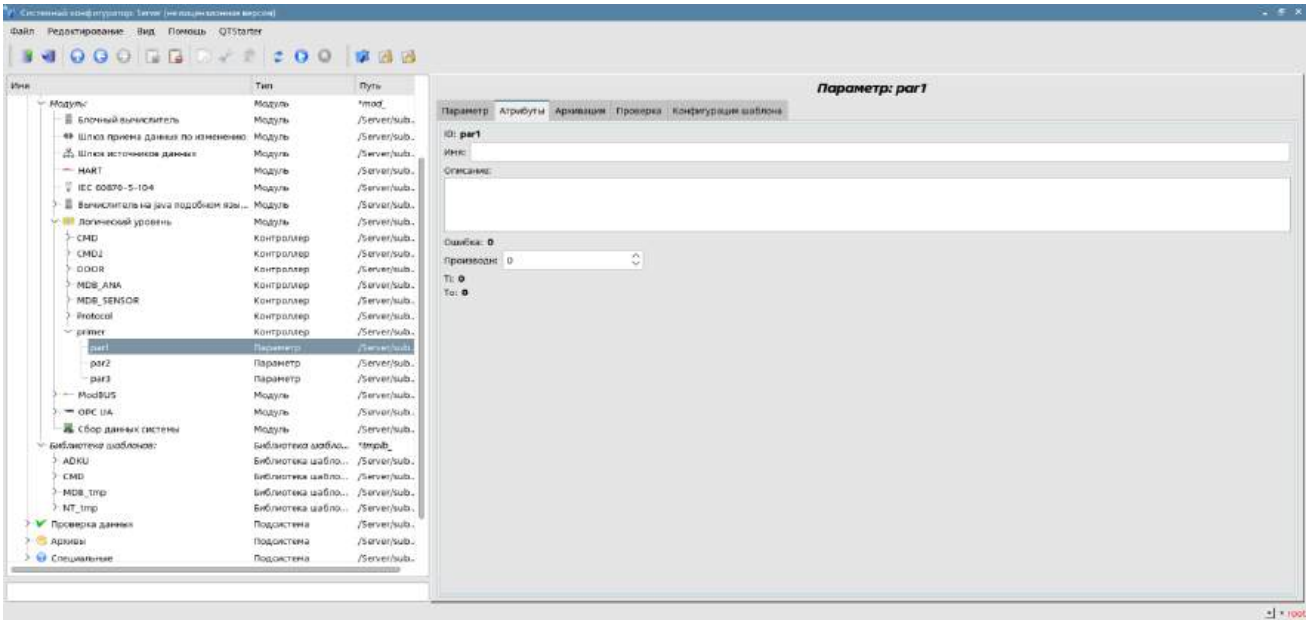


Рисунок 205

На этом конфигурация обработки данных считается законченной.

13.2 Модуль шлюз источников данных DAQGate

13.2.1 Назначение модуля

Основной функцией данного модуля является отражение данных подсистемы «Сбор данных» с серверов на АРМ. В своей работе модуль использует собственный протокол системы (Self System).

Модулем реализуется следующие функции:

– отражение структуры параметров подсистемы «Сбор данных» удаленного сервера на АРМ. Структура периодически при работе синхронизируется;

– доступ к текущим значениям атрибутов параметров и возможность их модификации. Значения атрибутов параметров обновляются с периодичностью исполнения локального контроллера на АРМ. Запросы на модификацию атрибутов транслируются на сервер;

– предоставление реализации механизма вертикального резервирования, а именно возможность отражения данных с нескольких серверов одного уровня на АРМ.

Использование схемы отражения данных представлено на рисунке 206.

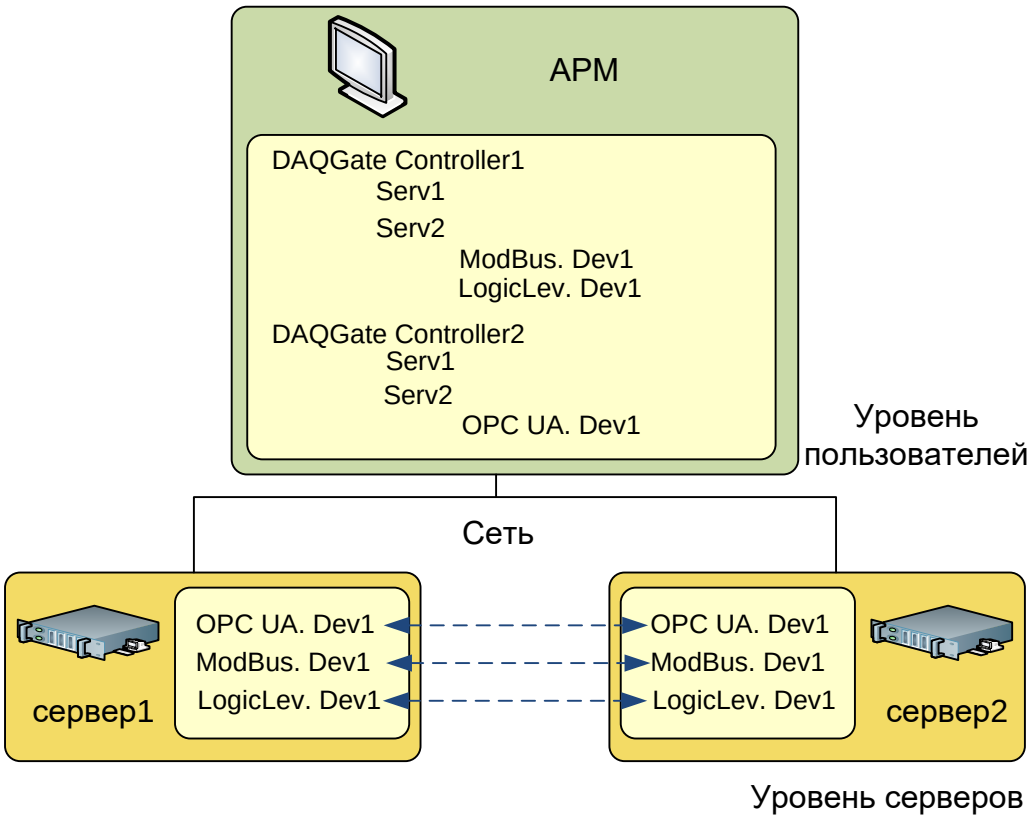


Рисунок 206

13.2.2 Конфигурирование передачи данных

Для подключения контроллера «Сбора данных» необходимо в левой части окна системного конфигуратора СКАДА раскрыть вкладку «Сбор данных» нажатием левой клавиши «мыши» на ней. В появившемся меню раскрыть вкладку «Модуль» и выбрать вкладку «Шлюз источников данных» (рисунок 207).

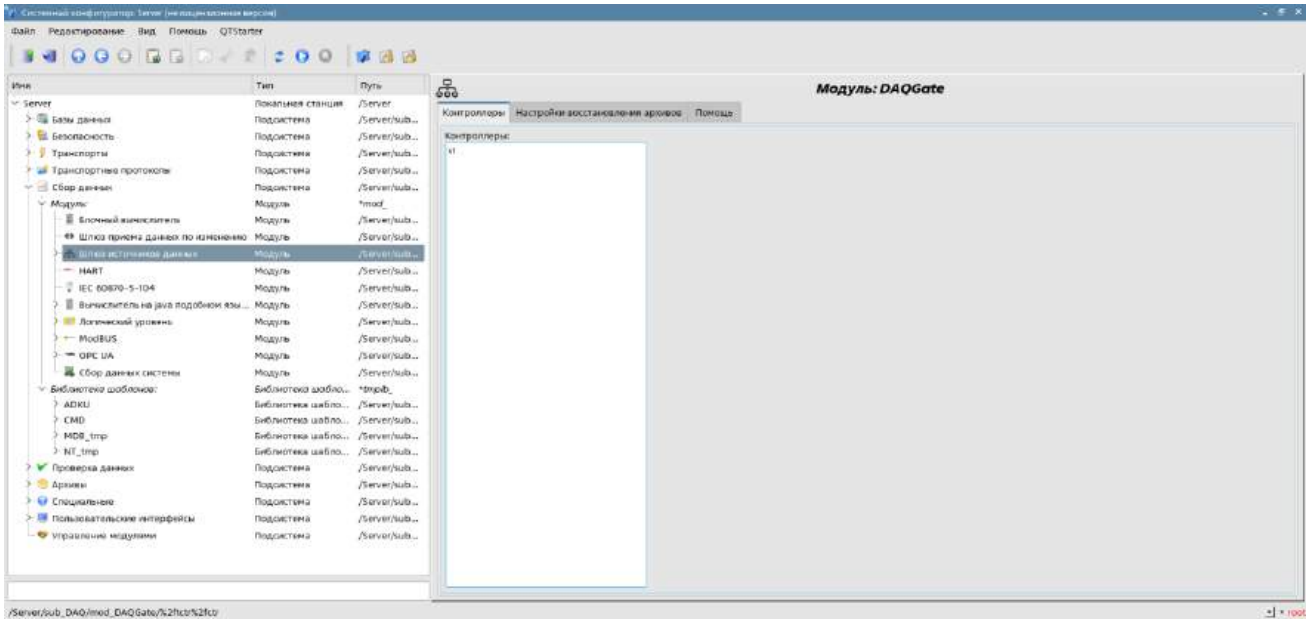


Рисунок 207

В правой части окна конфигуратора во вкладке «Контроллеры» нажать правую клавишу «мыши» и выбрать «Добавить» (рисунок 208). В диалоговом окне «Установка имени элемента» (рисунок 209) задать идентификатор (ID) шлюза сопряжения и его имя, например, «Gate». Нажать кнопку «ОК». В строке меню «Шлюз источников данных» появится вкладка «Gate».

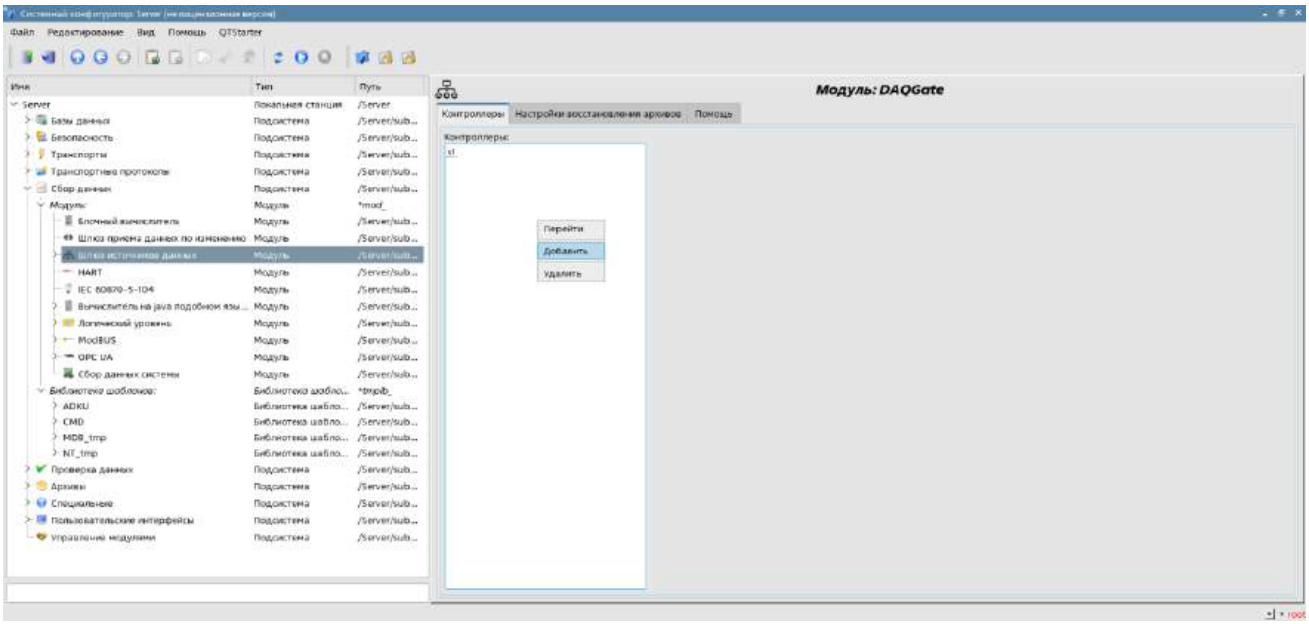


Рисунок 208

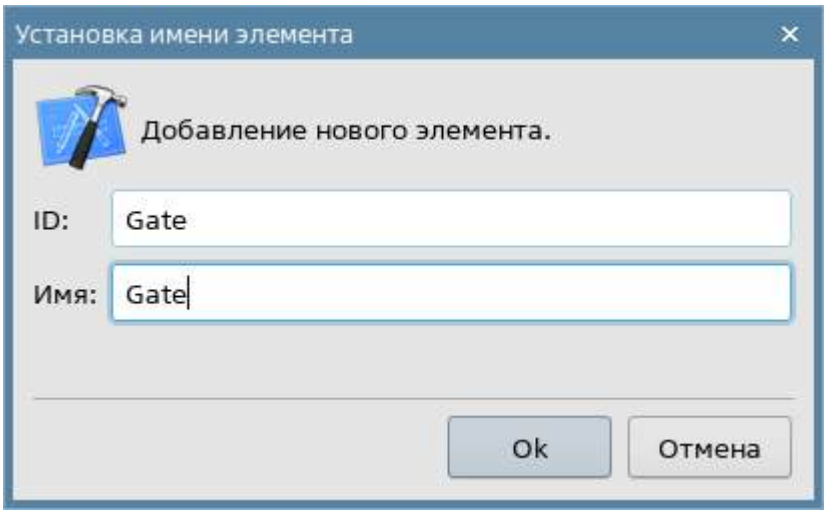


Рисунок 209

В левой части окна конфигуратора двойным нажатием левой клавиши «мыши» выбрать вкладку «Gate», при этом главное окно примет вид, изображенный на рисунке 210.

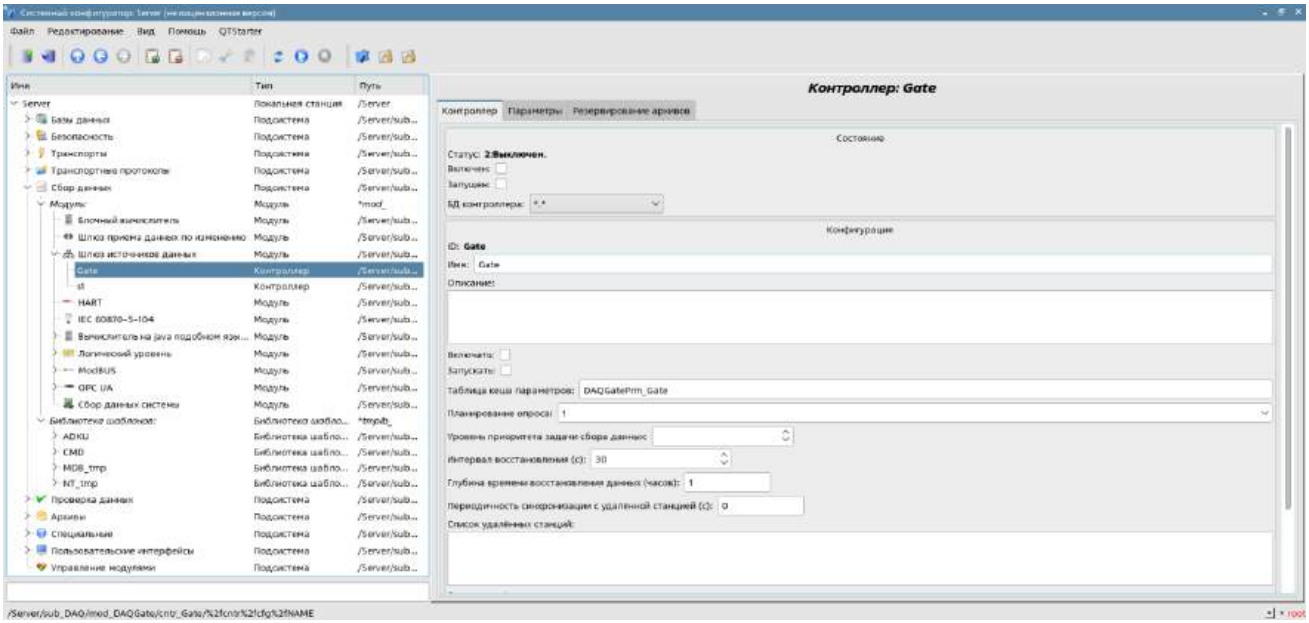


Рисунок 210

В окне «*Контроллер:Gate»* во вкладке «Контроллеры» в поле ввода «Описание» ввести информацию о подключаемом шлюзе.

Левой клавишей «мыши» установить отметки «Включать» и «Запускать». Это обеспечит автоматический запуск контроллера при следующем старте подсистемы обработки данных.

В поле ввода «Интервал восстановления» задать время восстановления соединения с подсистемой интеграции - 3 с.

В поле «Список удаленных станций» ввести имя подсистемы интеграции.

В поле «Список удаленных контроллеров и параметров» указать контроллеры системы интеграции, с которых будут передаваться данные. Имя контроллера должно совпадать с именем контроллера в шлюзе (рисунок 211).

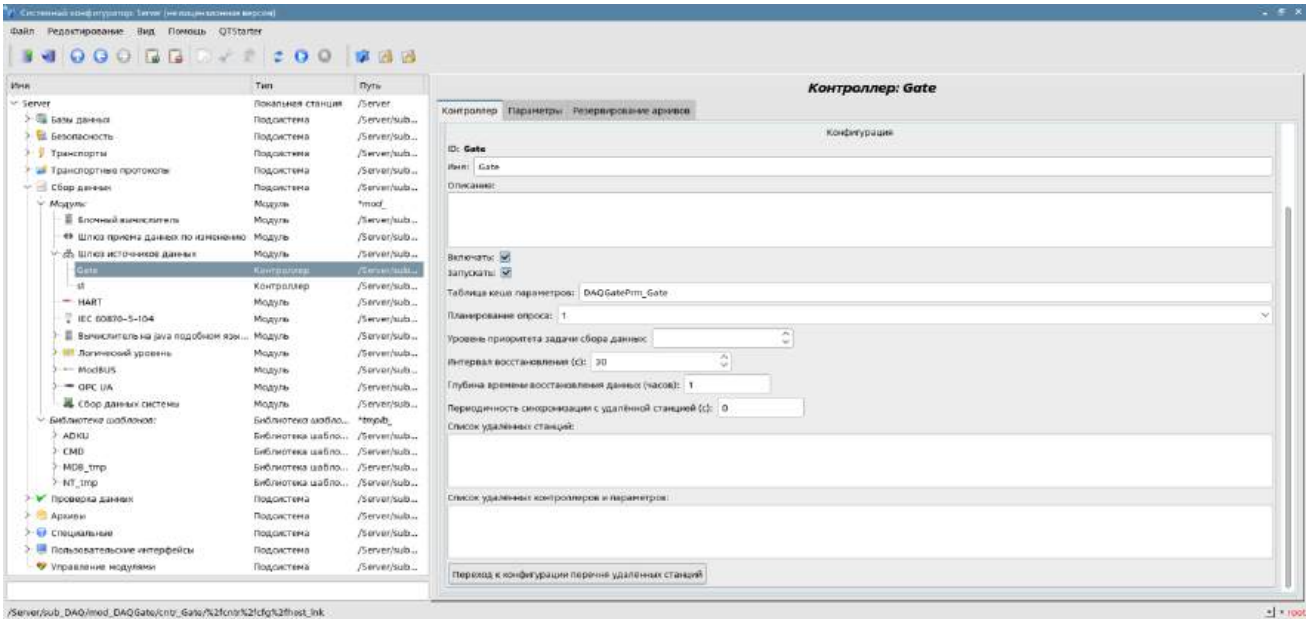


Рисунок 211

Левой клавишей «мыши» выбрать кнопку «Переход к конфигурации перечня удаленных станций». Основное окно программы примет вид, изображенный на рисунке 212.

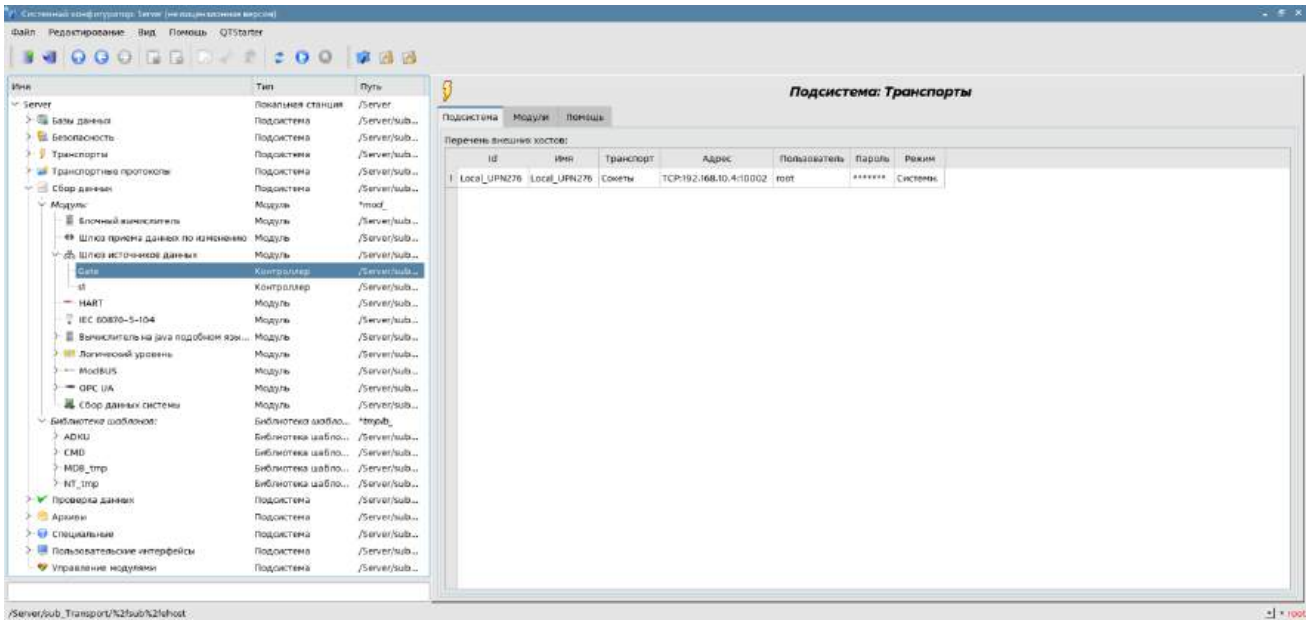


Рисунок 212

В правой части окна системного конфигуратора во вкладке «Подсистема» в столбце «Адрес» задать IP-адрес и порт, по которому будет осуществляться связь со шлюзами.

Сохранить конфигурацию в базе данных.

При следующем запуске подсистемы связь с настроенной подсистемой интеграции будет установлена автоматически.

13.3 Модуль источника данных SNMP

Данный модуль предоставляет возможность собирать информацию и вносить изменения у различных устройств по протоколу SNMP.

SNMP (англ. SimpleNetworkManagementProtocol — простой протокол сетевого управления) — стандартный интернет-протокол для управления устройствами в IP-сетях на основе архитектур TCP/UDP. К поддерживающим SNMP устройствам относятся маршрутизаторы, коммутаторы, серверы, рабочие станции, принтеры, модемные стойки и другие. Протокол используется для контроля подключенных к сети устройств на предмет условий, которые требуют внимания администратора. Он состоит из набора стандартов для сетевого управления, включая протокол прикладного уровня, схему БД и набор объектов данных.

SNMP предоставляет данные для управления в виде переменных, описывающих конфигурацию управляемой системы. Эти переменные могут быть запрошены (а иногда и заданы) управляющими приложениями.

Доступные через SNMP переменные организованы в иерархии. Эти иерархии описываются базами управляющей информации (базы MIB, от англ. ManagementInformationBase).

Базы MIB используют иерархическое пространство имен, содержащее идентификаторы объектов (OID-ы). Каждый OID состоит из двух частей: текстового имени и SNMP адреса в цифровом виде. Базы MIB выполняют вспомогательную роль по переводу имени объекта из текстового формата в формат SNMP (цифровой). Так как структура объектов на устройствах разных производителей не совпадает, без базы MIB практически невозможно определить цифровые SNMP адреса нужных объектов.

На рисунке 213 представлено окно настройки контроллера подсистемы Сбор данных по протоколу SNMP.

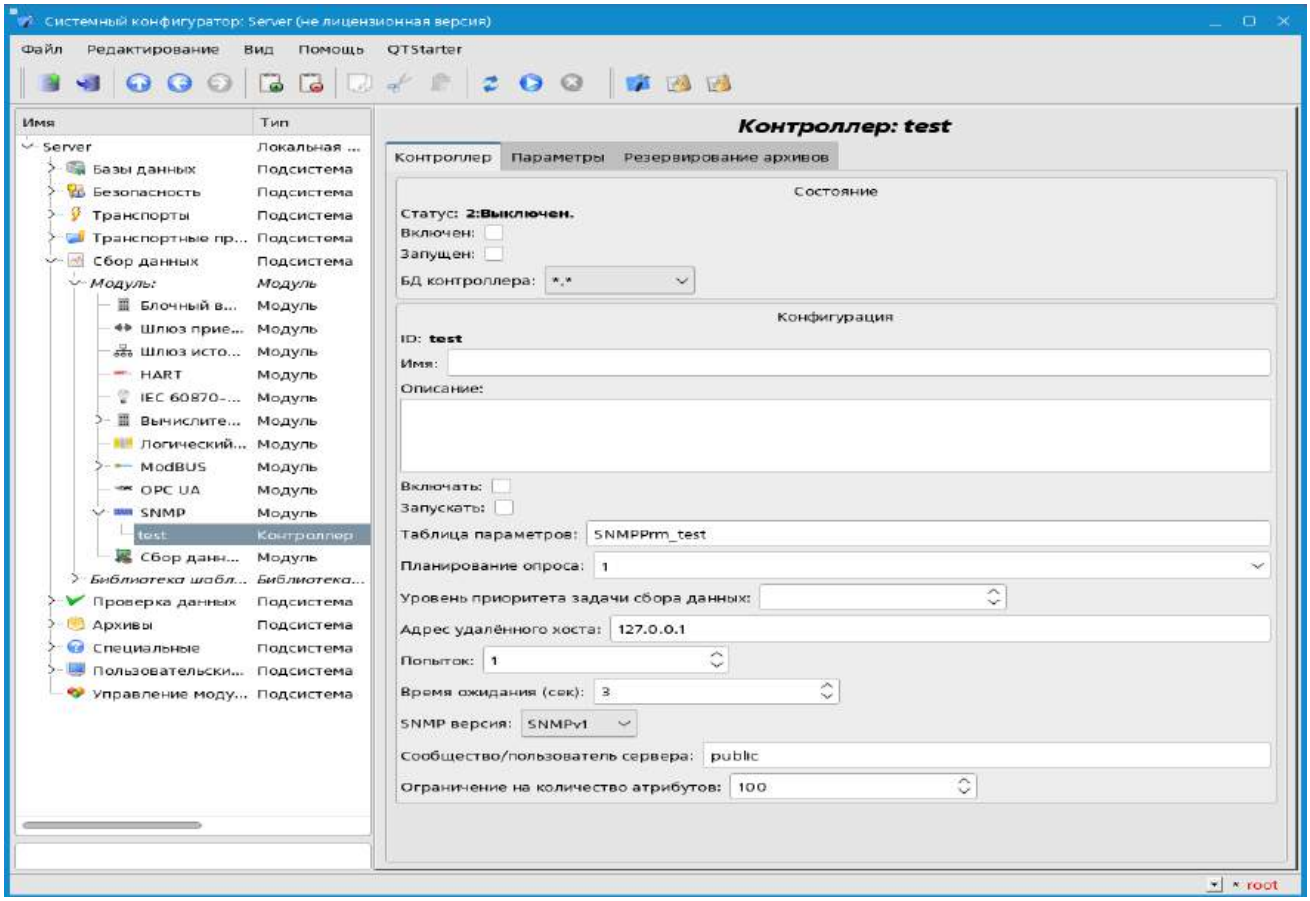


Рисунок 213

С помощью вкладки «Контроллер» 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: "Включать" и "Запускать";

– имя таблицы для хранения конфигурации параметров контроллера;

– планирование опроса;

– уровень приоритета задачи сбора данных (-1 – 99);

– адрес удаленного хоста – SNMP хост агента в IP адресе или доменном имени хоста, также можно установить порт «localhost:161»;

– количество попыток установить соединение;

– время ожидания (сек);

– SNMP версия – SNMPv1, SNMPv2c, SNMPv2u, SNMPv3. Для версии SNMPv3 становятся активными поля настройки уровня безопасности и авторизации/приватности;

– сообщество/пользователь сервера;

– ограничение на количество атрибутов (по умолчанию 100).

Для опроса параметров устройства по протоколу SNMP необходимо добавить контроллер модуля SNMP, соответствующий данному устройству. В поле «Адрес удаленного хоста» указать IP адрес опрашиваемого устройства. В атрибуте «SNMP версия» указать версию протокола SNMP.

На рисунке 214 приведен пример настройки контроллера модуля SNMP для ИБП.

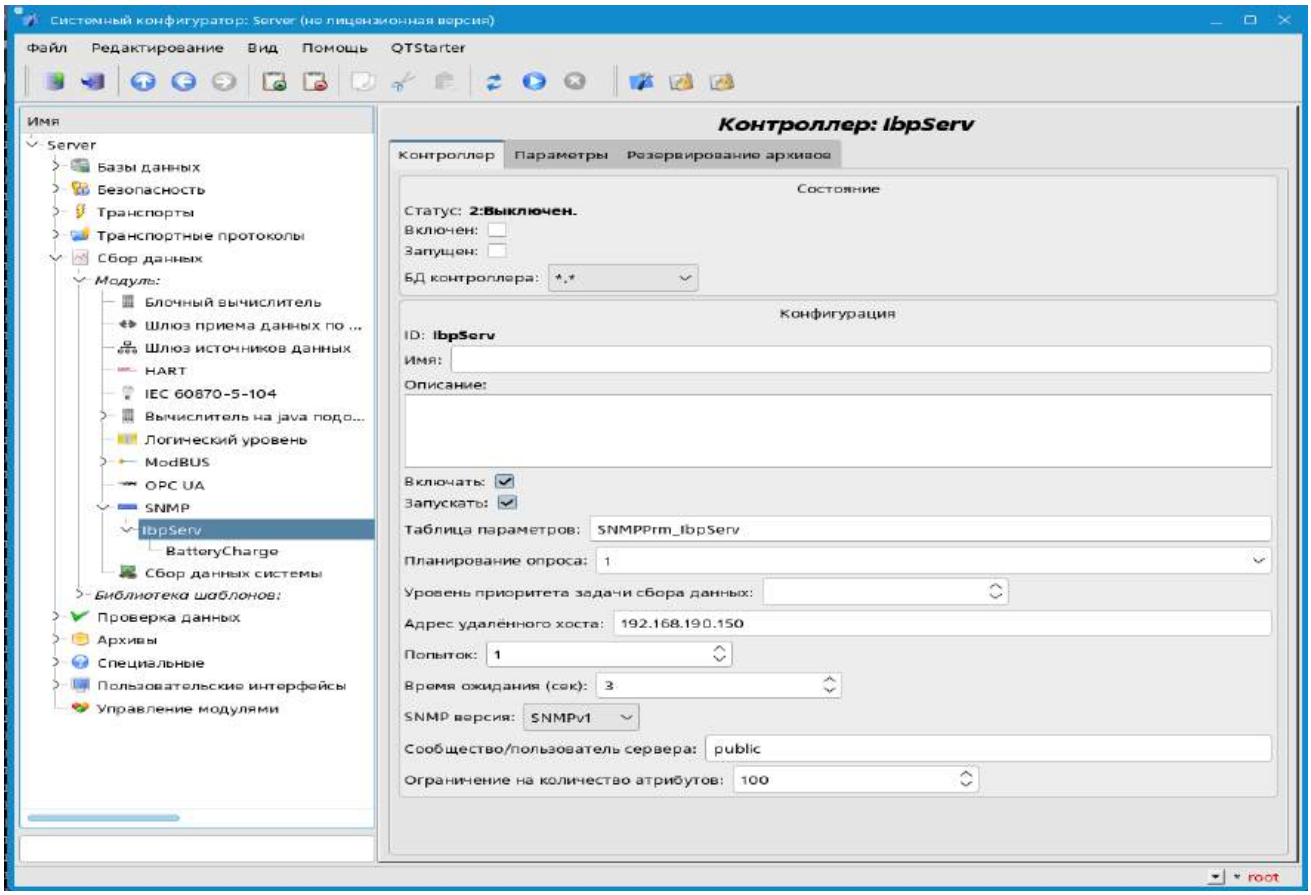


Рисунок 214

В созданном контроллере необходимо добавить требуемое число параметров. Основным атрибутом параметра является «Список OID (разделение след. строкой)».

В нем необходимо указать OID опрашиваемого параметра. На рисунке 215 приведен пример параметра модуля SNMP для заряда батареи ИБП.

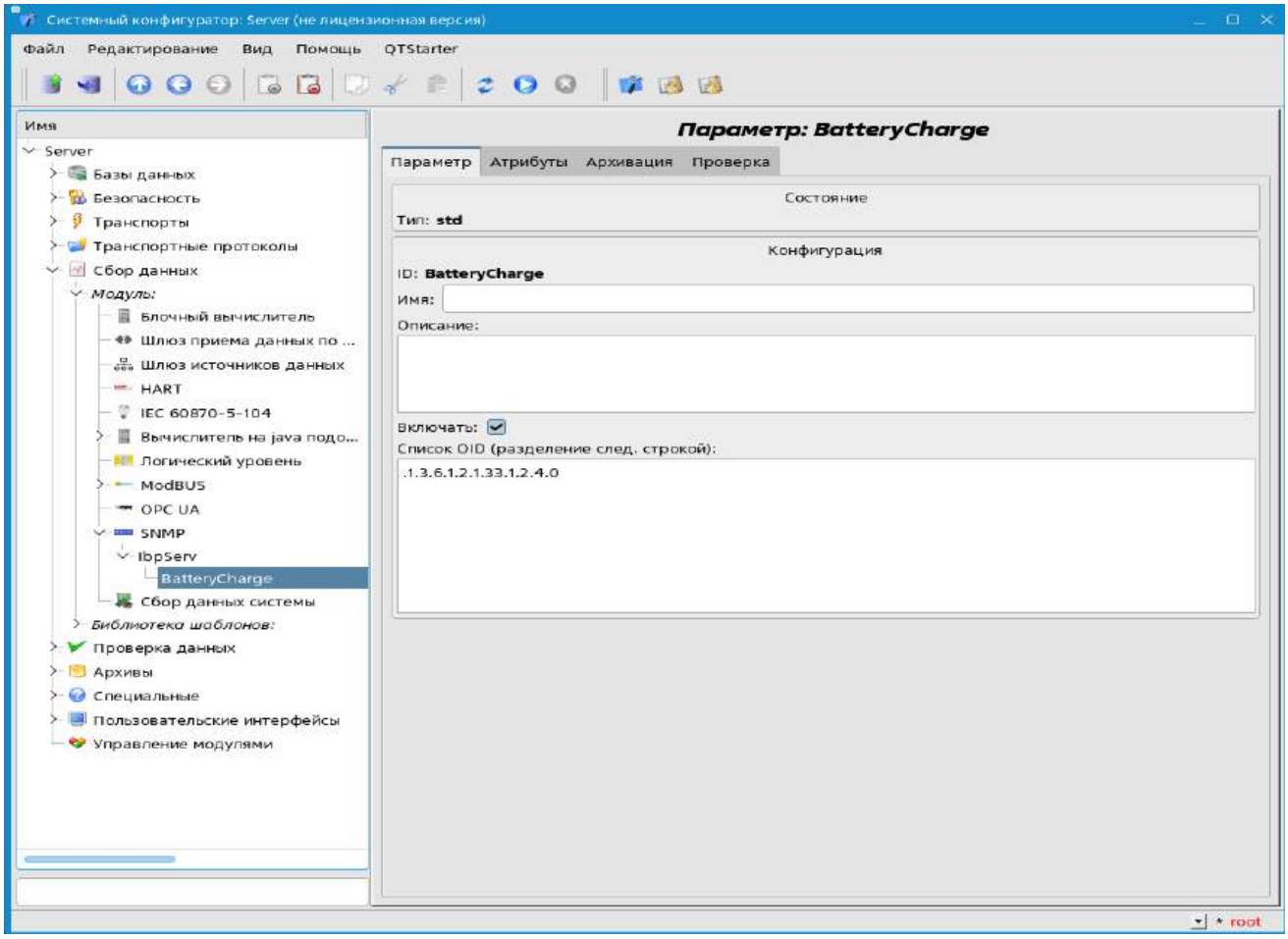


Рисунок 215

13.4 Модуль источника данных IEC 60870-5-104

Протокол IEC 60870-5-104 распространяется на устройства и системы телемеханики с передачей данных последовательными двоичными кодами для контроля и управления территориально распределенными процессами. Протокол регламентирует использование сетевого доступа по протоколу TCP/IP. Обеспечивает достаточно высокую функциональность при решении задач телеуправления, телесигнализации и телеизмерений, интеграции данных устройств в системы управления.

В основу протоколов положен обмен таблицами сигналов, причем типы данных, которыми осуществляется обмен, жестко фиксированы. Протокол не предусматривает возможность передачи сигналов реального времени.

Конфигурирование сбора данных по протоколу IEC 60870-5-104 осуществляется в окне конфигурирования контроллера (рисунок 216).

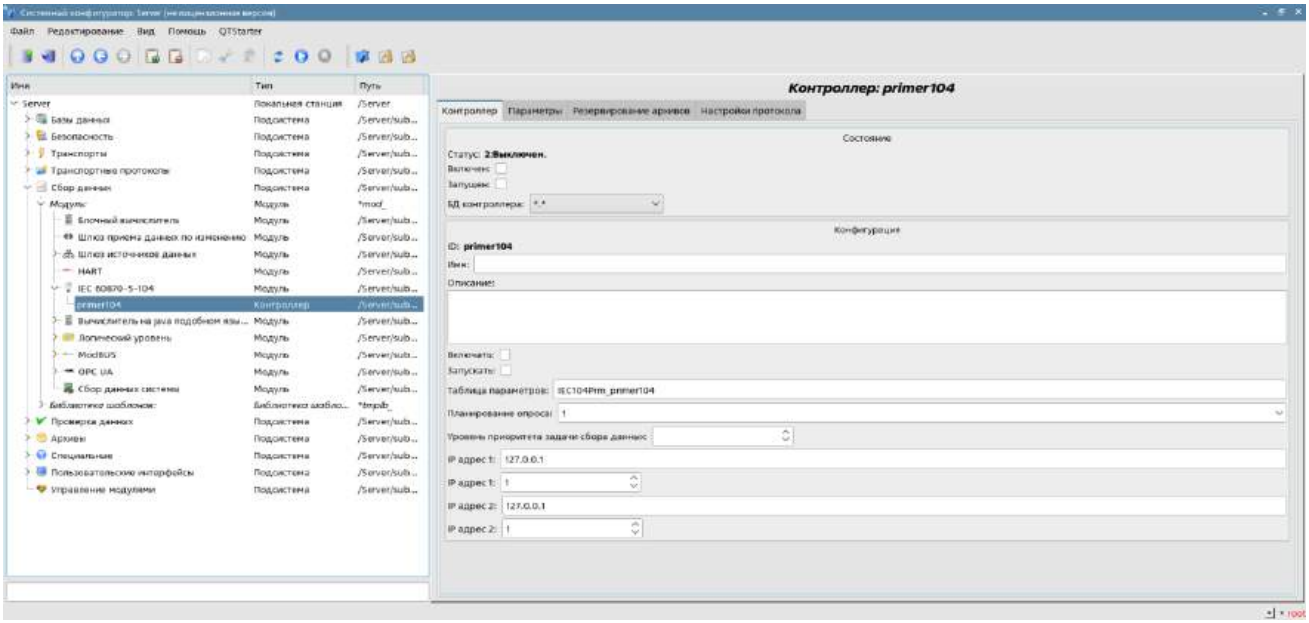


Рисунок 216

Индивидуальными настройками контроллера данного модуля являются поля настройки IP адресов, а также настройки протокола (рисунок 217).

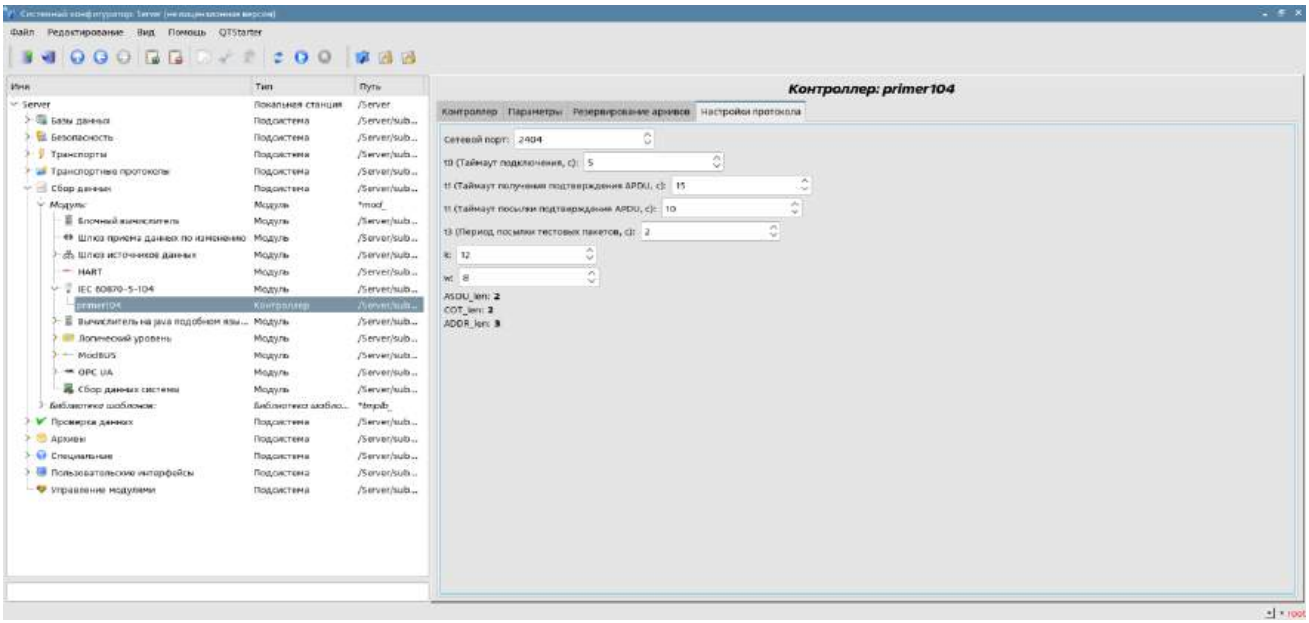


Рисунок 217

В СКАДА реализованы возможности настройки следующих параметров:

- сетевой порт – по умолчанию 2404;
- таймаут подключения (с);
- таймаут получения подтверждения APDU(с);
- таймаут посылки подтверждения APDU(с);
- период посылки текстовых пакетов(с);
- k- максимальное число неподтвержденных пакетов (значение от 2 до 99);
- w- количество пакетов до подтверждения (значение от 1 до 98).

Значения параметров устанавливаются в соответствии с сопроводительной документацией на контроллеры.

По умолчанию предустановлены следующие атрибуты:

длина адреса ASDU - ASDU_len: 2 байта;

длина причины передачи - COT_len: 2 байта;

общая длина адреса объекта - ADDR_len: 3 байта.

На рисунке 218 изображено окно конфигурирования параметра контроллера модуля IEC 60870-5-104.

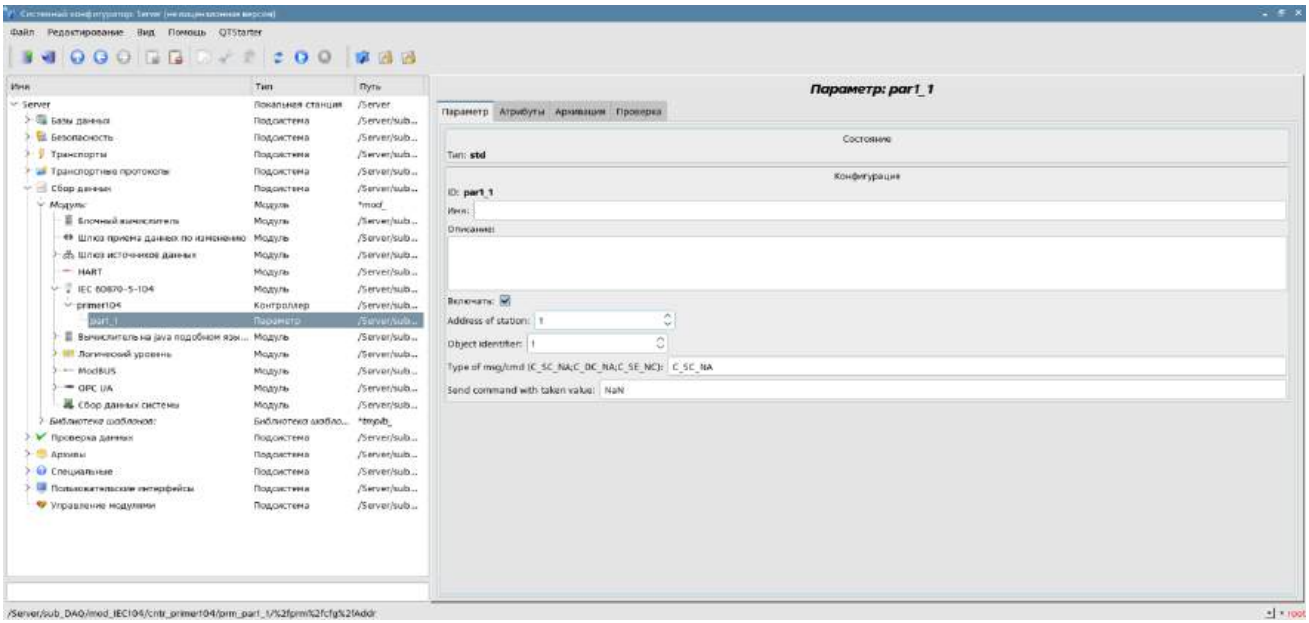


Рисунок 218

Индивидуальным атрибутом является поле для указания идентификатора объекта (object identifier), значение устанавливается в соответствии с документацией на контроллер.

13.5 Модуль источника данных OPC UA

В модуле OPC UA подсистемы «Сбор данных» производится настройка узла-клиента, сервер настраивается в модуле OPC UA подсистемы Транспортные протоколы (описание приведено в части 1 настоящего руководства оператора).

Для настройки клиента OPC UA необходимо создать контроллер в модуле OPC UA подсистемы «Сбор данных».

Для контроллера кроме стандартных настроек состояния и конфигурации доступны следующие поля:

– Период синхронизации с удаленной станцией (сек) - для отключения периодической синхронизации установить значение ноль;

– Конечный узел – указывается IP адрес сервера и порт соединения;

– Политика безопасности – позволяет выбрать алгоритм безопасности: нет, Basic128rsa15, Basic256. В зависимости от выбранного значения изменяется длина приватного ключа (REM);

– Режим безопасности сообщения – для выбора доступны следующие значения: нет, подпись, подпись&шифрование. Наиболее безопасным является подпись&шифрование;

– Сертификат (PEM) - указывается при необходимости, в соответствии с настройками сервера;

– Приватный ключ (PEM) - указывается при необходимости, в соответствии с настройками сервера;

– Аутентификация пользователь – имя пользователя;

– Аутентификация пароль – пароль;

– Ограничение количества атрибутов параметра: 100.

На рисунке 219 представлено окно настройки контроллера “testOPC”.

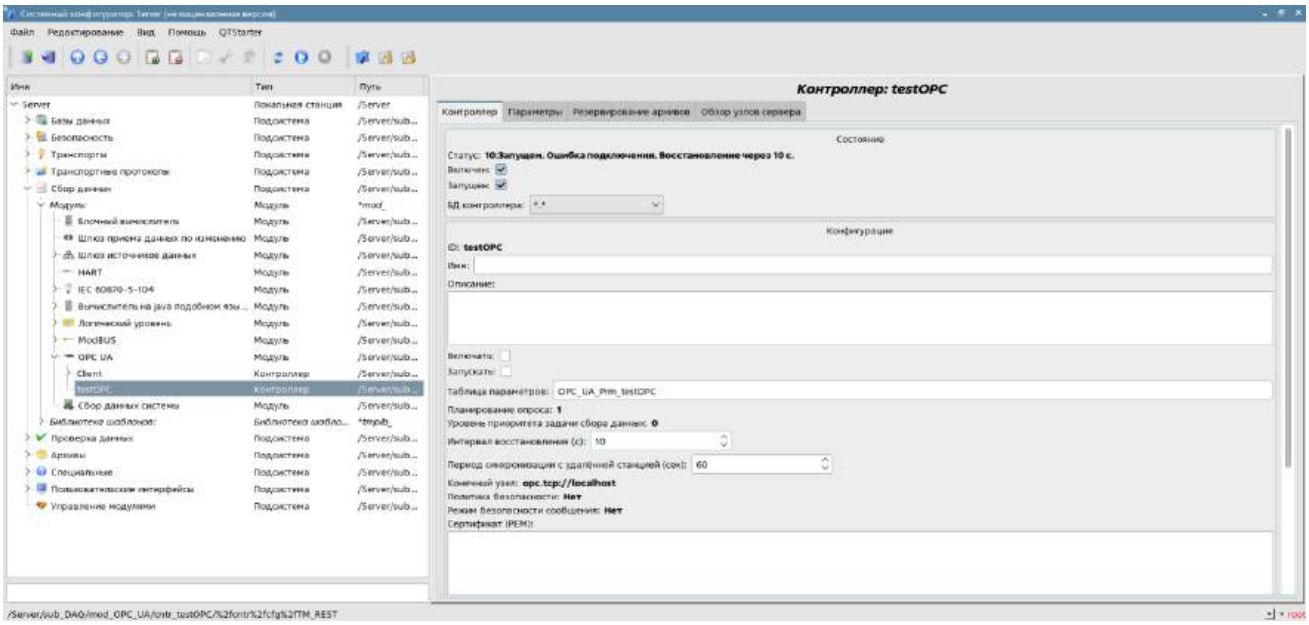


Рисунок 219

Добавляя параметры контроллера можно указать список узлов, представляющий из себя список переменных и контейнеров (объектов). Добавление узла производится последовательным выбором необходимого объекта в поле «Добавить узел» (рисунок 220). После включения параметра все переменные переносятся в перечень атрибутов параметра (рисунок 221) и автоматически заполняется поле список узлов. В подсказке к данному полю описан формат описания переменных (рисунок 222).

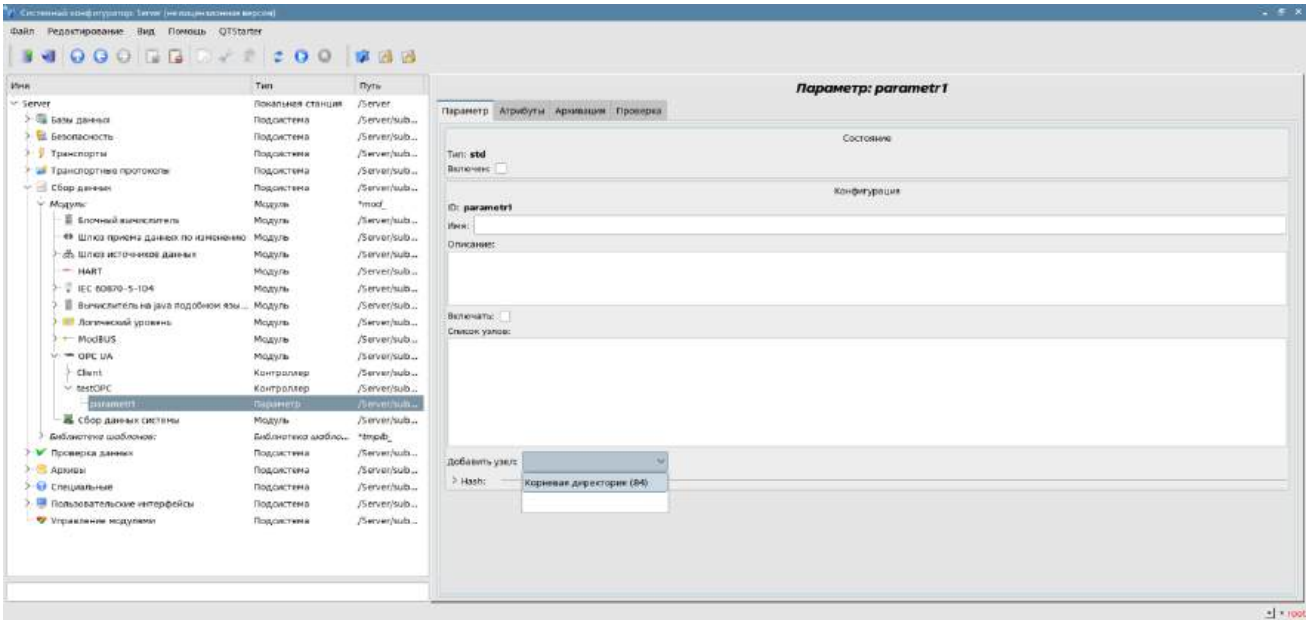


Рисунок 220

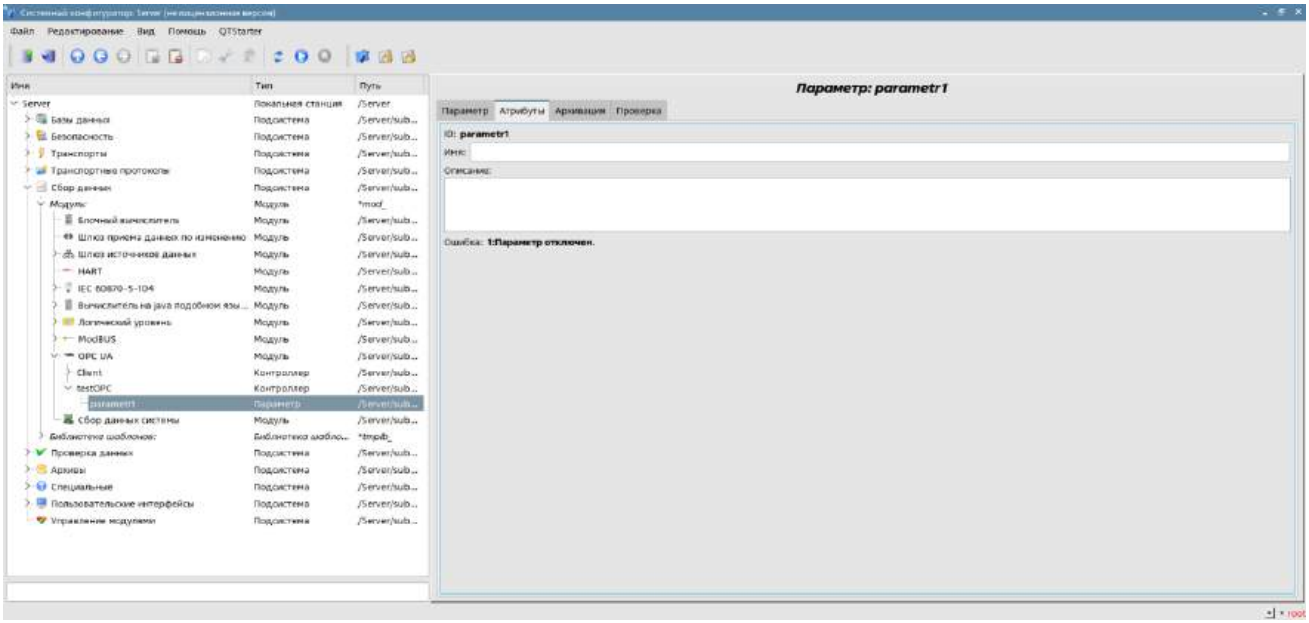


Рисунок 221

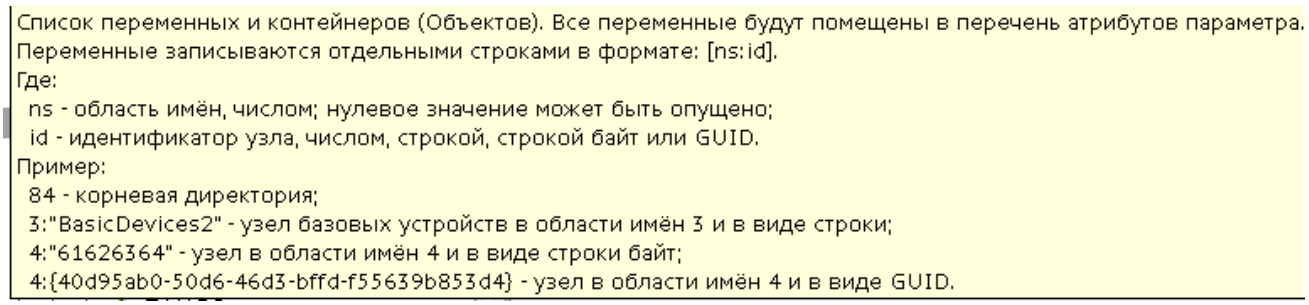


Рисунок 222

При подключении клиента также необходимо принять соглашение о сертификате сервера (согласиться в случае его отсутствия).

13.6 Модуль источника данных HART

Модуль HART предназначен для взаимодействия СКАДА с HART-устройствами.

Он позволяет конфигурировать параметры HART-устройств без использования HART-коммуникатора. Данные возможности важны, когда необходимо дистанционно с АРМ инженера настраивать параметры HART-устройств, находящихся в удаленных, или труднодоступных местах, позволяя конфигурировать, диагностировать и калибровать HART-устройства с АРМ инженера/оператора.

Алгоритм сбора данных модуля HART позволяет сравнивать и проверять полученные результаты собираемых данных со значениями, передаваемыми по каналу 4..20 мА на контроллер ТПТС.

Модуль HART использует механизм резервирования серверов СКАДА, что гарантирует непрерывную работу модуля HART и СКАДА в случае нештатного отключения одного из серверов, использует централизованный механизм архивирования СКАДА, что позволяет обеспечить архивирование переменных модуля HART поступающих от HART-устройств за определенный временной интервал.

При открытии модуля HART кроме стандартных вкладок «Контроллеры» и «Помощь» доступна вкладка «Настройка модуля», которая при установке соответствующего флага позволяет выводить в консоль запуска платформы отладочную информацию.

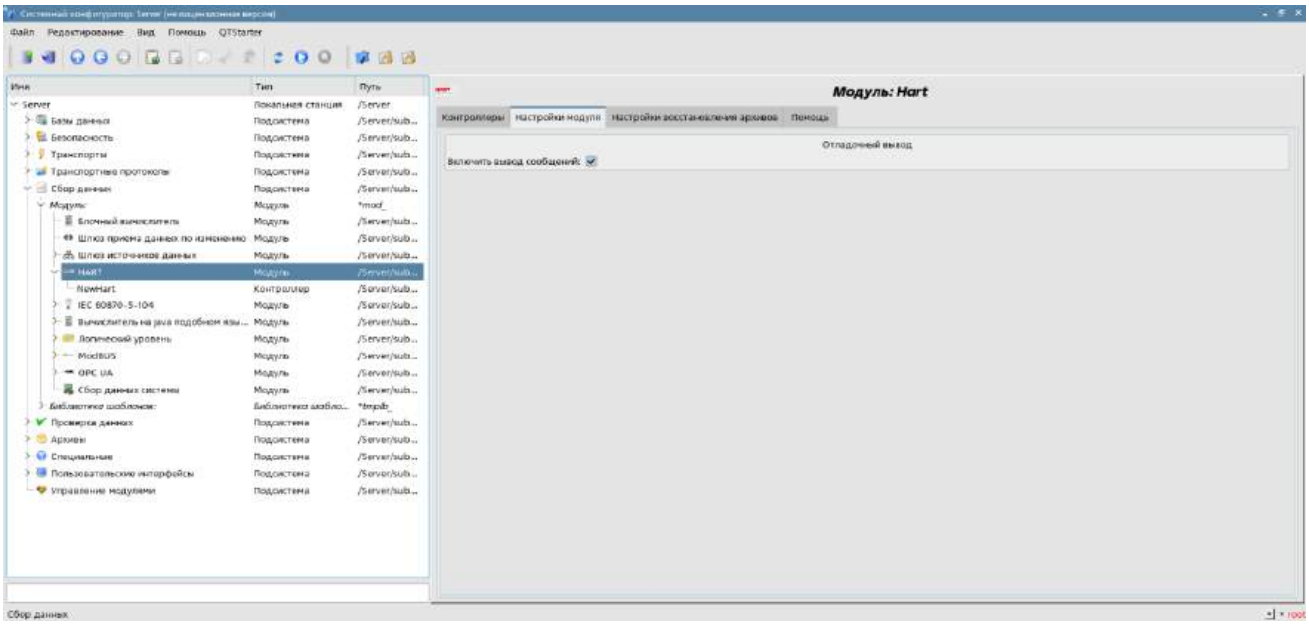


Рисунок 223

Для добавления нового контроллера в модуль HART необходимо выбрать пункт контекстного меню «Добавить» и ввести ID и имя добавленного устройства. На вкладке «Контроллер» созданного элемента отражаются настраиваемые параметры для подключения HART-устройства (рисунок 224).

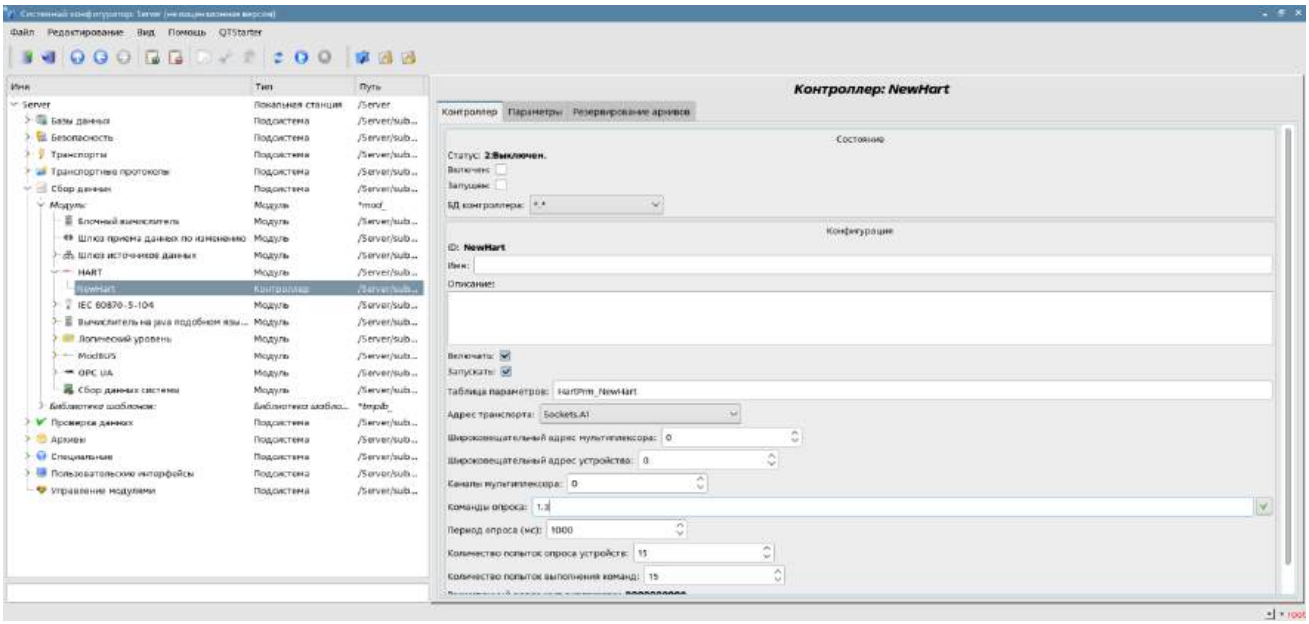


Рисунок 224

Индивидуальными настройками контроллера являются:

Таблица параметров - определяет имя таблицы, где будут храниться параметры контроллера;

Адрес транспорта – содержит адрес транспорта, созданного в модуле «Транспорты» - «Последовательный интерфейс» - «Выходной транспорт», на котором будет организовано взаимодействие с HART-устройством (создание транспорта описано в части 1 руководства оператора);

Широковещательный адрес мультиплексора - адрес подключаемого HART-мультиплексора, настраиваемый на корпусе прибора*;*

Широковещательный адрес устройства – по умолчанию у HART-устройств этот адрес нулевой*;*

Канал мультиплексора – номер одного из 16 или 32 каналов на плате MCR-S-MUX-TB мультиплексора, к которому подсоединено подключаемое HART-устройство;

Период опроса(мс)- период опроса датчика по заданному каналу мультиплексора;

Попытки опроса датчика – количество попыток опроса датчика в случае отсутствия ответа;

Попытки выполнения команды – количество попыток запроса выполнения команд у логического контроллера;

Расширенный адрес мультиплексора – значение в поле устанавливается автоматически в результате выполнения команды cmd0;

Расширенный адрес устройства – значение в поле устанавливается автоматически в результате выполнения команды mux0*;*

Команды для опроса – позволяет выбрать существующую команду для опроса (по введенному OID), при этом на вкладке «Атрибуты» отразятся атрибуты, соответствующие выбранной команде, а в поле «Описание» соответствующее описание команды. Для команд «mux1, mux2, mux3» допустимо вводить в данном поле только номера команд в диапазоне от 1 до 3 через «,» (например: «1,2,3»). В таблице 1 представлено описание поддерживаемых команд. Если команда не поддерживается HART-устройством, на передней панели мультиплексора загорается красный индикатор. Если команды для опроса не указаны, то опрос данного устройства производиться не будет.

Таблица 21

Идентификатор (OID)	Краткое описание команды
cmd0	Чтение уникального идентификатора HART-мультиплексора
mux0	Чтение уникального идентификатора HART-устройства
mux1	Чтение первичной переменной
mux2	Чтение первичной переменной как величины тока и в процентах от диапазона
mux3	Чтение четырех основных переменных и токового значения первичной переменной
mux14	Чтение информации сенсора первичной переменной
mux15	Чтение информации о выходном сигнале по первичной переменной
mux34	Запись значения демпфирования первичной переменной
mux35	Запись значения диапазона первичной переменной
mux36	Установка значения верхней границы диапазона сенсора
mux37	Установка значения нижней границы диапазона сенсора
mux38	Сброс флага изменения конфигурации
mux40	Вход/выход из режима фиксированного токового значения первичной переменной
mux41	Самотестирование прибора
mux43	Установка нуля первичной переменной
mux45	Подстройка нуля ЦАП первичной переменной
mux46	Подстройка коэффициента усиления ЦАП первичной

Идентификатор (OID)	Краткое описание команды
	переменной
mux47	Запись функции преобразования первичной переменной процесса
mux48	Чтение дополнительного статуса HART-устройства
mux108	Запись номера команды для монопольного режима
mux109	Управление режимом монопольной работы

Разработанные команды конфигурирования имеют следующие особенности:

- команда 34 (OID - mux34) задает величину демпфирования первичной переменной. Для ее отправки необходимо задать «Время демпфирования»;

- команда 35 (OID - mux35) записывает данные диапазона значений датчика. Для ее отправки необходимо ввести «Код единиц диапазона» (целое), и значения верхнего и нижнего предела диапазона (вещественное). Целые значения кодов единиц перечислены в приложении 3.

- команды 45, 46 (OID - mux45, mux46) задают верхнее и нижнее значение тока ЦАП (вещественные). Это команды из разряда калибровочных. В некоторых датчиках фиксировано значение 4..20мА;

- команда 40 (OID - mux40) задает фиксированное значение тока первичной переменной. Для этого в поле вводится вещественное значение для входа в режим фиксированного тока. Отправленное нулевое значение означает выход из режима фиксированного тока;

- команда 47 (OID - mux47) задает функцию преобразования первичной переменной. Значение 0 соответствует линейной функции. Могут быть заданы другие функции передачи, например, квадратный корень;

- команды 36, 37, 43 (OID - mux36, mux37, mux43) задают верхнюю границу, нижнюю границу и нуль диапазона. При нажатии на кнопку выполнения команды, выбранная величина устанавливается согласно текущему значению первичной переменной.

Для конфигурирования связи по HART-протоколу необходимо для контроллера создать параметр и выбрать в поле «OID» имя HART-команды. При этом, на вкладке «Атрибуты» можно убедиться что, в зависимости от введенного идентификатора команды - OID, вид окна будет отличаться (например, рисунки 225 - 230).

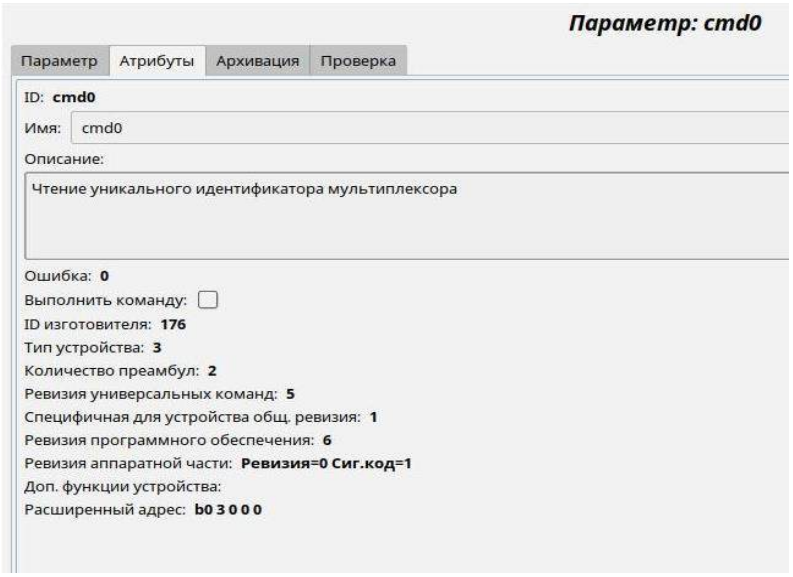


Рисунок 225

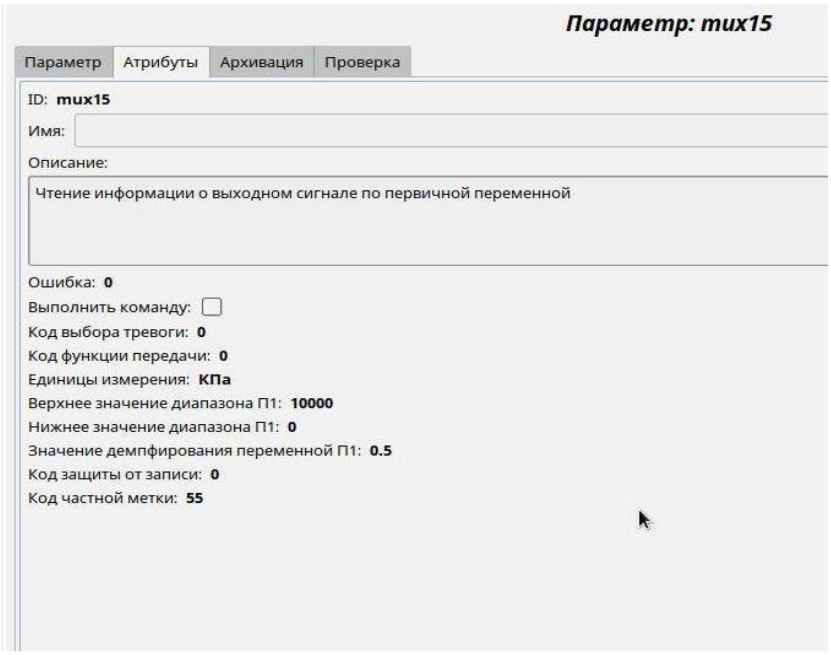


Рисунок 226

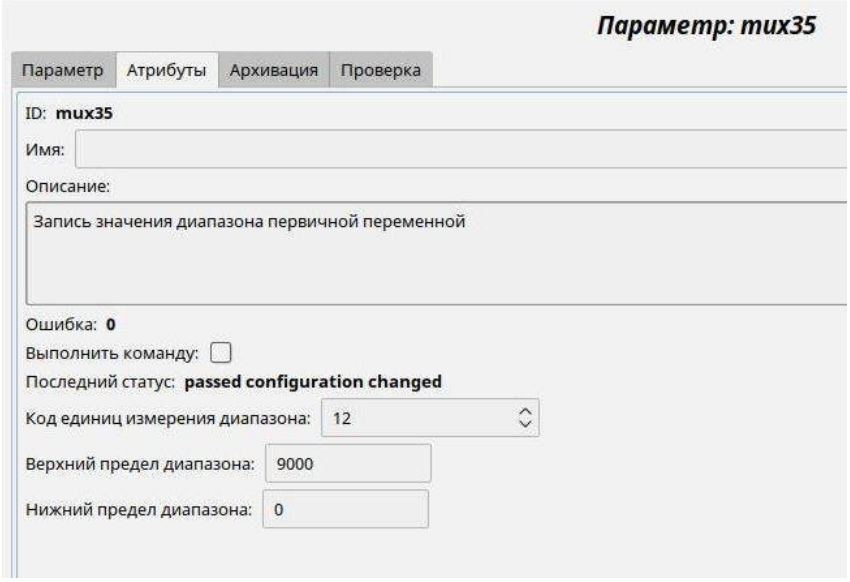


Рисунок 227

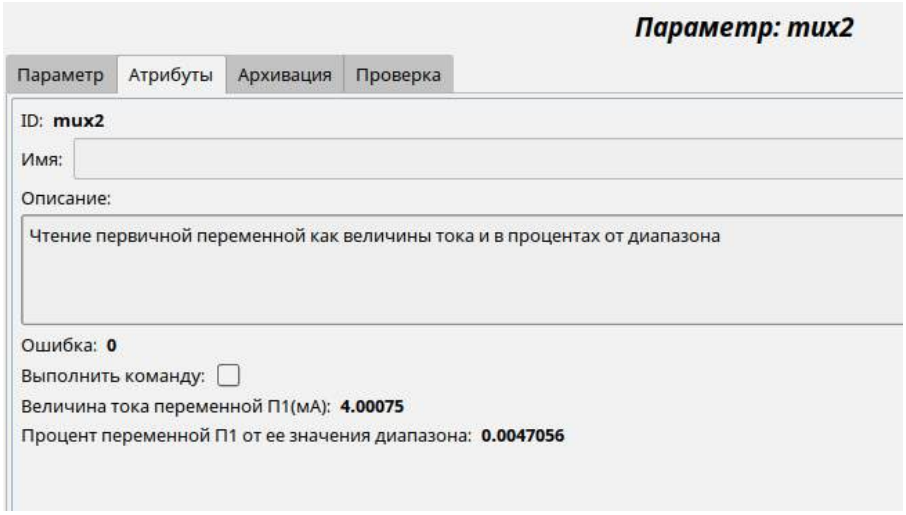


Рисунок 228

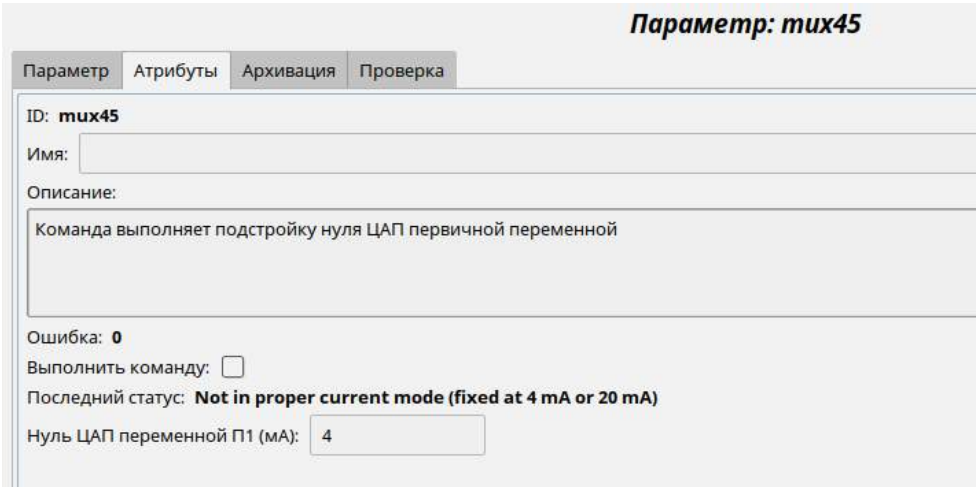


Рисунок 229

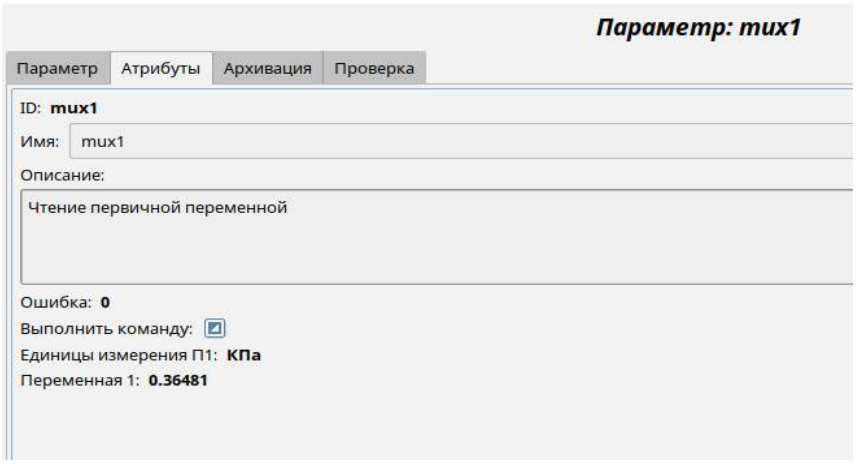


Рисунок 230

При выполнении команд конфигурирования HART-устройства (команды от 34 до 48) в поле «Последний статус» возвращается статус выполнения команды. Расшифровка статусов команд модуля представлена в приложении 4.

14. НАСТРОЙКА РЕЗЕРВИРОВАНИЯ

14.1 Резервирование модуля «Базы данных»

Для настройки резервирования модуля «Базы данных» необходимо в модуле «Транспорты» создать внешний хост на каждом из узлов. Для этого *на стороне основного узла*: в поле Id ввести имя резервного узла, а в поле «Адрес» ввести IP адрес резервного узла. *На стороне резервного узла*: в поле Id ввести имя основного узла, а в поле «Адрес» ввести IP адрес основного узла. В поле «Режим» выбрать «Пользов. и Системн.». На рисунке 231 приведен пример модуля «Транспорты» для резервного узла.

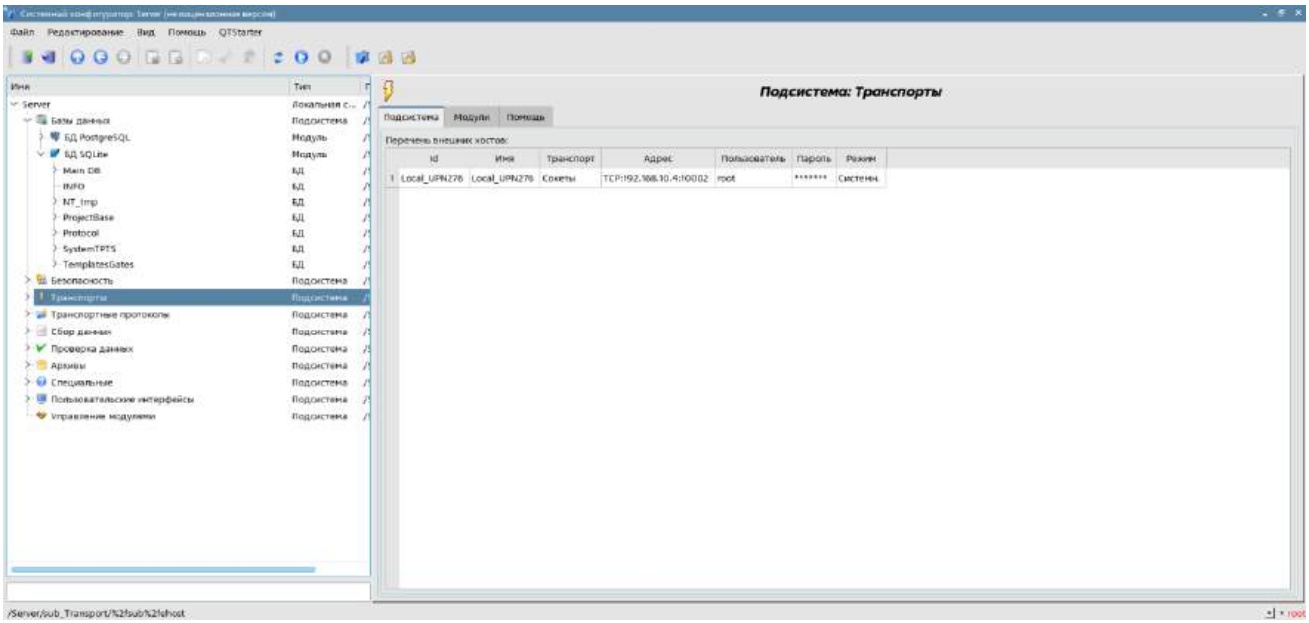


Рисунок 231

Далее для каждого узла выполнить следующие действия:

– в окне «Базы данных» на вкладке резервирование в таблице «Станции» добавить узел;

– *на стороне основного узла* - в поле Id ввести имя резервного узла, добавленного в транспорты, *на стороне резервного узла* - в поле Id выбрать имя основного узла, добавленного в транспорты;

– в таблице «Таблицы» создать по строке на каждую из таблиц, которые требуется резервировать;

– в поле «БД» необходимо ввести путь до базы данных, содержащей нужную таблицу;

– в поле «Таблица» ввести Id таблицы;

– в поле «Предп. исп.» выбрать то же имя узла, что и в таблице «Станции»;

– сохранить сделанные изменения.

На рисунке 232 приведен пример настройки модуля «Базы данных» для резервного узла.

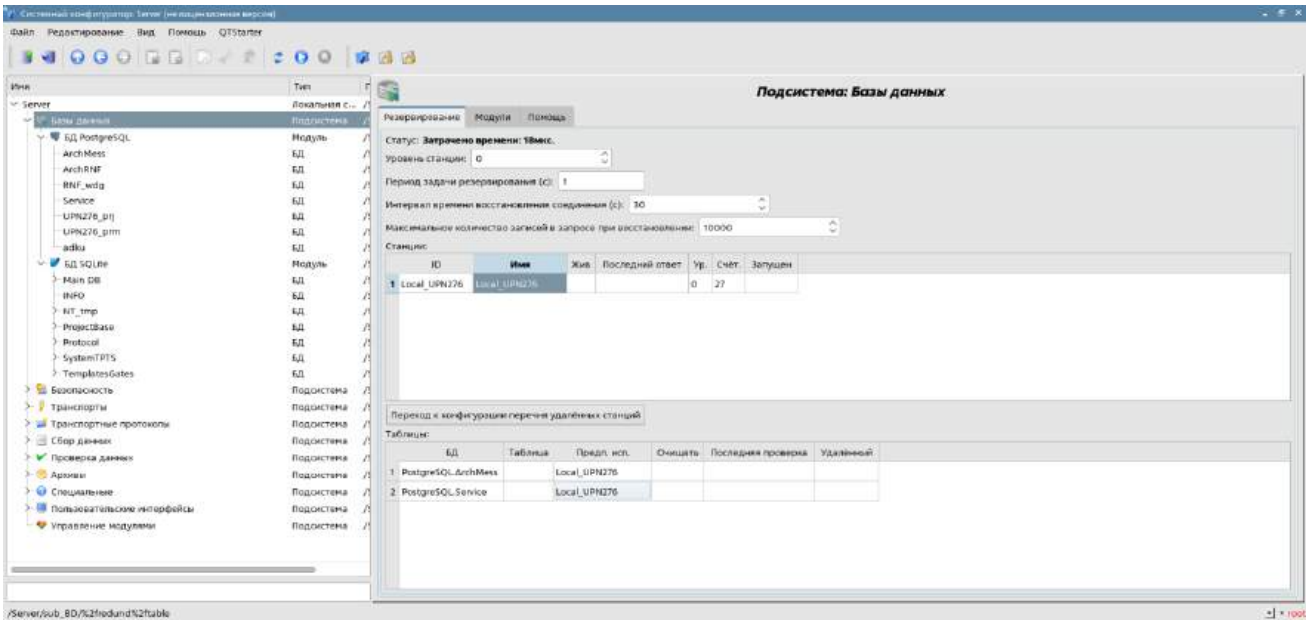


Рисунок 232

14.2 Резервирование модуля «Сбор данных»

Для работы механизма синхронизации данных на каждом сервере хранится список адресов серверов, от которых возможно получение данных. При восстановлении работоспособности сервера после его отказа каждый контроллер SCADA-системы (настроенный на синхронизацию данных) запускает отдельный поток, в котором выполняется загрузка из другого сервера необходимых архивных данных по каждому атрибуту. Если список состоит из нескольких серверов, то выбирается работающий сервер, содержащий аналогичный контроллер. В случае отсутствия подходящего сервера процесс синхронизации прекращается по данному контролеру. Начало запрашиваемого временного промежутка по каждому архиву атрибута выбирается исходя из последней метки времени поступивших данных в архив текущего сервера. Окончание запрашиваемого временного промежутка задается уже на сервере, с которым синхронизируют данные: указывается время начала процесса получения архива. Переданные данные атрибута, помещаются в архив, независимо от процесса архивации атрибута, то есть данные не помещаются в буфер архиватора, а сразу записываются в БД.

Для увеличения быстродействия процесса синхронизации передача данных между серверами выполняется через отдельный транспорт, не задействованный в

других операциях SCADA-системы. Кроме того, запись архивов в БД происходит по отдельным подключениям к ней для каждого контроллера SCADA-системы.

Для настройки резервирования модуля «Сбор данных» необходимо создать внешний хост в модуле «Транспорты» (как описано в пункте 14.1). Далее для каждого узла выполнить следующие действия:

– в поле «Резерв.» выбрать «Архивное»;

– в поле «Предп. исп.» выбрать то же имя узла, что и в таблице «Станции»;

– сохранить сделанные изменения.

На рисунке 233 приведен пример настройки модуля «Сбор данных» для резервного узла.

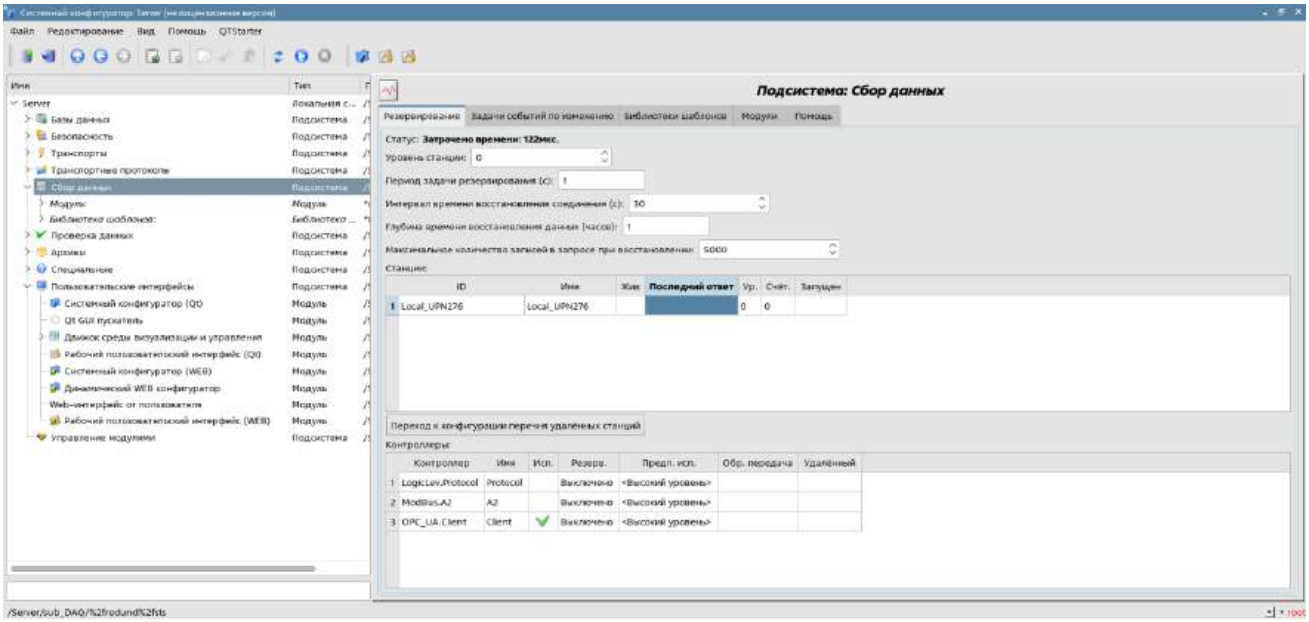


Рисунок 233

15. РЕКОМЕНДАЦИИ ПО РАБОТЕ С ПРОЕКТОМ

15.1 Редактирование графической части проекта АСУ ТП

При разработке графической части проекта все элементы, группы элементов и видеокадры необходимо создавать во вкладке «Виджеты», а затем во вкладке «Проект» подключать разработанные видеокадры. При внесении изменений в проект все изменения необходимо вносить в виджеты проекта (вкладка «Виджеты»). В этом случае изменения автоматически будут отражены в проекте. При редактировании виджетов проекта в окне «Проект» после перезагрузки изменения будут утрачены.

15.2 Настройка отображения проекта на удаленном АРМ

Для настройки удаленного подключения к проекту: в системном конфигураторе необходимо выбрать «Пользовательский интерфейс» → «Рабочий пользовательский интерфейс» → «Удаленные экраны». Данная вкладка предоставляет возможность конфигурирования вывода видеокадров на экраны удаленного АРМ (рисунок 234). Для этого выбрать в списке станций для удаленного открытия видеокадров на экранах «Local». Задать для каждого экрана группу пользователей, для которых разрешено управление отображением видеокадров на этом экране. Также можно изменить имена экранов, отредактировав их в таблице. Кнопка «Обновить» заново определяет количество подключенных мониторов и отображает эту информацию в таблице.

Чтобы настроить вывод видеокадров на экраны для удаленного рабочего места, необходимо выбрать название этого рабочего места из списка станций для удаленного открытия видеокадров. Затем задать количество экранов и их имена для выбранного рабочего места. Также можно запросить имена экранов с удаленной станции, нажав кнопку «Обновить».

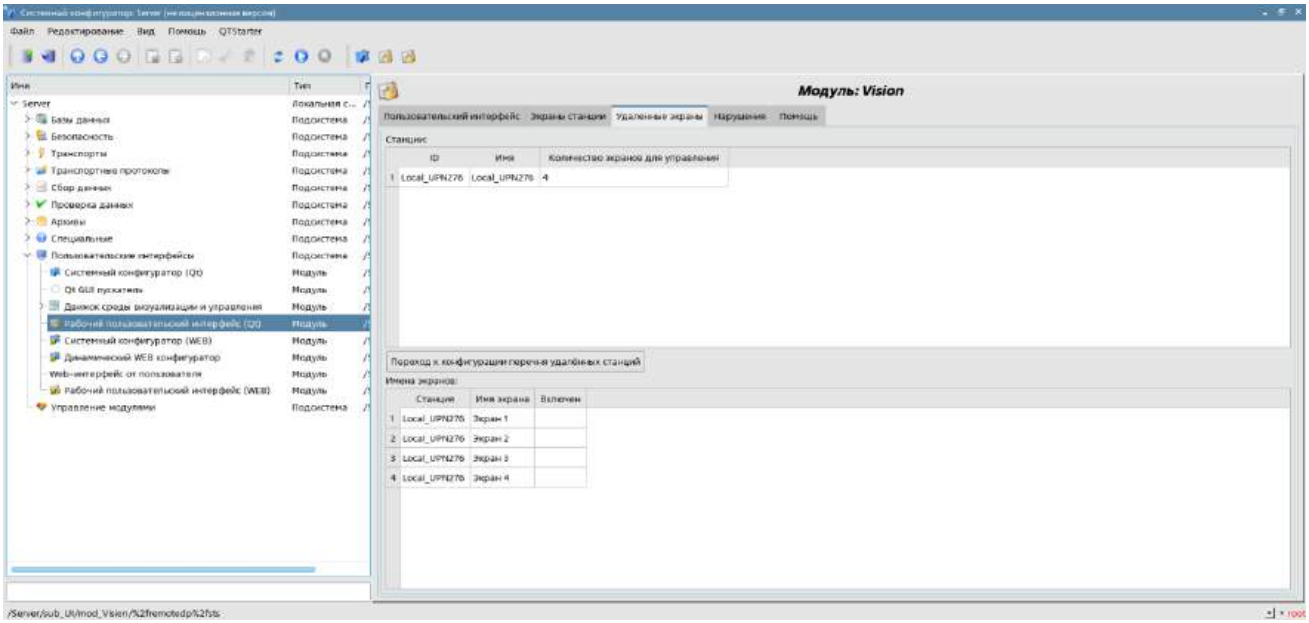


Рисунок 234

15.3 Настройка отображения проекта на нескольких мониторах

Для открытия окна проекта на нескольких мониторах одного АРМ необходимо в окне системного конфигуратора выбрать «Пользовательские интерфейсы»→«Рабочий пользовательский интерфейс (Qt)», на вкладке «Пользовательский интерфейс» ввести порядок отображения проектов на мониторах в строке «Перечень запускаемых проектов» в формате: ‘PrjName - 1’, где PrjName - название проекта; 1 - номер монитора (рисунок 235). Сохранить сделанные изменения.

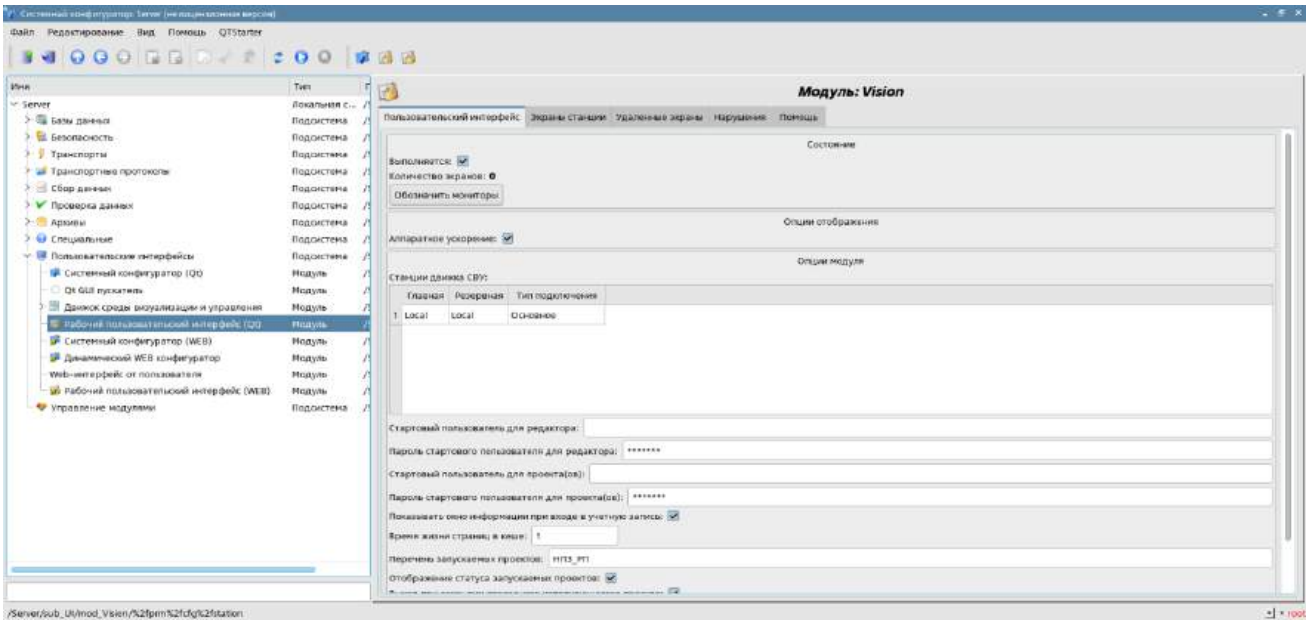


Рисунок 235

15.4 Сохранение проекта АСУ ТП и его частей

15.4.1 Рекомендации

Для сохранения проекта АСУ ТП и его частей рекомендуется использовать СУБД PostgreSQL, где сохраняется по умолчанию.

Для сохранения проекта и его частей необходимо создать новую БД. Количество БД должно быть достаточным для реализации поставленной задачи и не должны содержать идентичные элементы.

Сохранение архивов необходимо производить в отдельной БД.

15.4.2 Что не рекомендуется делать при сохранении проекта

Не рекомендуется один и тот же элемент сохранять в разных БД, т.к. это приводит к непредсказуемым результатам при одновременной загрузке БД. Под проектом АСУ ТП понимаем набор сконфигурированных БД, источников данных, включая их обработку и сохранение в архивы, а также сконфигурированный графический интерфейс оператора. Не путать с вкладкой «Проект» графического редактора.

Не рекомендуется непосредственно менять стандартные конфигурации и элементы стандартных библиотек, а также сохранять собственные, новые, библиотеки и элементы в базах данных стандартных библиотек.

15.5 Перенос конфигурации СКАДА из одного проекта в другой

При переносе отдельной БД необходимо произвести ее копирование с источника и загрузку в новом проекте приемника.

Перенос конфигурации производится в зависимости от типа БД:

- *для БД SQLite* – необходимо скопировать нужный файл *.db из директории БД старого проекта в директорию баз данных нового (например, /var/spool/scada/DATA/). После этого необходимо запустить СКАДА и провести подключение БД в СКАДа в модуле БД SQLite, т.е. создать объект БД, ввести для нее корректные параметры в поле «Адрес» и сохранить сделанные изменения. В строке состояние поставить галочку «Включен» и нажать на кнопку «Загрузить систему из этой БД» (производится аналогично описанному в пункте 12.2.2 настоящей части руководства оператора).

- *для БД PostgreSQL* – необходимо создать дамп копируемой БД. Для этого в окне терминала Fly ввести команду:

pg_dump -U postgres -Ft -c -d <имя_БД> | gzip > /путь приемника/<ИмяБД>.tar.gz

После чего поместить дамп в директорию /var/opt АРМ приемника и восстановить БД во временную новую БД. Для этого необходимо из системного меню вызвать окно терминала Fly и запустить оболочку psql от имени непривилегированного пользователя ОС - postgres (администратор БД), выполнив команду:

psql –U postgres

При этом система попросит ввести пароль для пользователя *postgres*, заданный при установке базового ПО. В результате, в окне терминала высветится приглашение postgres=#.

Далее следует создать новую базу данных командой:

create database <“имя временной БД”>;

Выйти из программной оболочки psql командой \q.

Затем выполнить команду:

gunzip -c /var/opt/<имяБД>.tar.gz | pg_restore -U postgres -Ft -d <“имя временной БД”> -v

После этого необходимо запустить СКАДА и в модуле БД PostgreSQL создать БД с именем временной БД (в соответствии с 12.2), ввести для нее корректные параметры в поле «Адрес», сохранить сделанные изменения. В строке состояние поставить галочку «Включен» и нажать на кнопку «Загрузить систему из этой БД». Далее для каждого загруженного контроллера необходимо изменить «БД контроллера» на «PostgreSQL.GenDB» и сохранить данные в БД.

После пересохранения всех контроллеров в основной БД проекта временную БД можно удалить. Для этого необходимо в окне терминала Fly ввести следующие команды:

psql –U postgres

drop database <“имя временной БД”>;

\q

и провести перезагрузку СКАДА.

16. ОБЩЕСИСТЕМНОЕ API ПОЛЬЗОВАТЕЛЬСКОГО ПРОГРАММИРОВАНИЯ

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

16.1 Общесистемные пользовательские объекты

Объект представляет собой контейнер свойств и функций. Свойства могут содержать данные базовых типов и другие объекты. Существует 4 базовых типа: нулевой, логический, числовой и строковый. Доступ к свойствам может осуществляться посредством записи имен через точку <obj.prop> или через квадратные скобки <obj["prop"]>. Объект создается посредством оператора new: <var0=newObject()>. Различные компоненты могут дополнять объект свойствами и функциями. Базовые типы также обладают свойствами и функциями.

Свойства и функции базовых типов:

- нулевой тип, функции:
  - *bool isEVal()* - возвращает true;
- логический тип, функции:
  - *bool isEVal()* - проверка на EVAL;
  - *string toString()* - представление значения в виде строки "false"или "true";
- целое и вещественное число, свойства:
  - *MAX_VALUE* - максимальное значение;
  - *MIN_VALUE* - минимальное значение;
  - *NaN*- недостоверное значение;
- целое и вещественное число, функции:
  - *bool isEVal()* - проверка на EVAL;
  - *string toExponential(int numbs=-1)* - возврат числа с плавающей точкой в виде строки с количеством значащих цифр <numbs>;
  - *string toFixed(int numbs=0, int len=0, bool sign=false)* – возврат числа с фиксированной точкой в виде строки с количеством цифр после точки <numbs>, минимальной длиной <len> и знаком <sign>;

- *string toPrecision(in tprec=-1)* – возврат числа в виде строки с количеством значащих цифр <prec>;

- *string toString(int base=10, int len=-1, bool sign=false)* – возврат целого числа в виде строки с базой <base=2-36>, минимальной длиной <len> и знаком <sign>.

Строка, свойства:

- *int length* - длина строки.

Строка, функции:

- *bool is EVal()* – проверка на EVAL;
- *string charAt(int symb)* – извлекает из строки символ <symb>;
- *int charCodeAt(int symb)* – извлекает из строки код символа <symb>;
- *string concat (sting val1, string val2,...sring valN)* - возвращает строку, сформированную путем присоединения val1,..valN к исходной;
- *int indexOf(string substr, int start)* – возвращает позицию <substr> в строке начиная со <start>. Если позиция не указана, поиск идет с начала;
- *int lastIndexOf(string substr, int start)* – возвращает позицию <substr> в строке начиная со <start> с конца. Если позиция не указана, поиск идет с конца;
- *int search(string pat, string flg="")* – поиск в строке по шаблону <pat> с флагами <flg>. Возвращает позицию найденной подстроки;
- *int search (RegExp pat)* - поиск в строке по шаблону RegExp <pat> (см. описание объекта RegExp);
- *Array match(string pat, string flg="")* – поиск в строке по шаблону <pat> с флагами <flg>. Возвращает массив с найденной подстрокой и подвыражениями;
- *Array match (RegExp pat)* - поиск в строке по шаблону RegExp <pat>. Возвращает массив с найденной подстрокой и подвыражениями;
- *string slice(int beg, int end), string substr(int beg, int end)* - возврат строки, извлеченной из исходной, начиная с позиции <beg> и заканчивая <end>. Если <beg> или <end> отрицательна, поиск ведется с конца;
- *Array split(string sep, int limit)* - возврат массива элементов строки, разделенных <sep>. Количество элементов ограничено <limit>;
- *Array split(RegExp pat, int limit)* - возврат массива элементов строки, разделенных шаблоном RegExp <pat>. Количество элементов ограничено <limit>;
- *string insert(int pos, string substr)* – вставка в позицию <pos> подстроки <substr>;
- *string replace(int pos, int n, string str)* - заменяет на строку <str> <n> символов, начиная с <pos>;
- *string replace(string substr, string str)* – замена всех строк <substr> на <str>;

- *string replace(RegExp pat, string str)* – замена всех строк по шаблону <pat> на <substr>;

- *real toReal()* - преобразование строки в число типа real;

- *int toInt(int base=0)* - преобразование строки в целое число с основанием <base>. Если <base>=0, то преобразование происходит с учетом префикса строки (123 - десятичное, 0123 - восьмеричное, 0x123 - шестнадцатеричное);

- *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 в <off>;

- *string parsePath(int pos, int off=0)* - выделение из исходной строки элемента <pos> от смещения <off>. Результирующее смещение помещается в <off>;

- *string path2sep(string sep=".")* - преобразование пути в текущей строке в строку с разделителем <sep>.

16.1.1 *Объект "Array"*

Объект "Array" представляет собой массив, который создается командой *<var0=newArray(prm1, prm2,... prmN)>*. Массив работает со свойствами как с индексами, поэтому обращение к свойствам доступно только через квадратные скобки (var0[1]). Массив имеет набор стандартных свойств и функций.

Свойства массива:

- *length* - возвращает размер массива.

Функции массива:

- *string join(string sep=","), string toString(string sep=","), string valueOf(string sep=",")* – возвращает строку с элементами массива, разделенными <sep> или ",";

- *Array concat(Array arr)* - добавляет к исходному массиву элементы массива <arr>;

- *int push(EITp var,...)* - помещает элементы <var> в конец массива, как в стек, возвращает новый размер массива;

- *EITp pop()* - удаление последнего элемента массива и возврат его значения, как из стека;

- *Array reverse()* - изменение порядка расположения элементов массива;

- *EITp shift()* - сдвиг массива вверх, при этом первый элемент удаляется, а его значение возвращается;

- *int unshift(EITp var,...)* - задвигает элементы <var> в массив;

- *Array slice(int beg, int end)* - возвращает фрагмент массива от <beg> до <end>;

- *Array splice(int beg, int remN, EITp val1, EITp val2,...)* - вставляет, удаляет или заменяет элементы массива. Возвращает массив удаленных элементов. В первую

очередь удаляются элементы с позиции <beg> и количеством <remN>, затем вставляются значения <val1> и т.д., начиная с позиции <beg>;

- *Array sort()* - сортировка элементов в лексикографическом порядке.

16.1.2 Объект "RegExp"

Объект работы с регулярными выражениями. При создании объекта в качестве аргументов передается строка с текстом регулярного выражения и флаги в виде строки символов:

- "g" - режим глобального поиска;
- "i" - режим регистронезависимого поиска;
- "m" - режим многострочного поиска;
- "u" - принудительное разрешение символов UTF-8;
- "p" - тестирование выражения по обычному шаблону с ключевыми символами "?", "*", "\";

Свойства объекта:

- source - исходный шаблон регулярного выражения;
- global - признак глобального поиска;
- ignoreCase - игнорировать регистр при поиске;
- multiline - признак многострочного поиска;
- UTF8 - UTF-8 - символы разрешены;
- lastindex - индекс символа за подстрокой последнего поиска, используется для продолжения сканирования при следующем вызове.

Функциииобъекта:

- Array exec(string val) - вызов поиска по строке <val>. Возвращает найденную подстроку и подвыражения в массиве. Устанавливает атрибут массива <index> в позицию найденной подстроки, атрибут <input> в значение исходной строки;
- bool test(string val) - возвращает <true>, если подстрока найдена в <val>.

Пример работы с объектом:

```
varre = newRegExp("(\\d\\d)[-/](\\d\\d)[-\](\\d\\d(:?\\d\\d)?)","");
var rez = re.exec("12/30/1969");
var month = rez[1];
var day = rez[2];
var year = rez[3];
var OK=rez.test("12/30/1969");
```

16.1.3 Модуль FLibSys

Специальный модуль FLibSYS предоставляет в систему СКАДА статическую библиотеку функций для работы с системой СКАДА, на уровне её системного API. Эти функции могут использоваться в среде пользовательского программирования системы СКАДА для организации неординарных алгоритмов взаимодействия.

Для адресации к функциям этой библиотеки можно использовать статический адрес вызова *"Special.FLibSYS.{Func}()"* или динамический *"SYS.Special.FLibSYS["{Func}"].call()", "SYS.Special.FLibSYS.{Func}()"*. Где *{Func}* — идентификатор функции в библиотеке.

16.1.4 Объект XMLNodeObj

Функции:

- *string name( )* – имя узла, XML-тега;
- *string text(bool full=false)* – текст узла, содержимое XML-тега.Установить full для получения комбинированного текста со всех включенных узлов;
- *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 clear (bool full = false)* – очищает узел, удалением дочерних узлов, очищает текст и атрибуты, для <full>;
- *XMLNodeObj childAdd(ElTp no = XMLNodeObj); XMLNodeObjchildAdd(string no)* – добавление объекта <no> как вложенного. <no> может быть как непосредственнообъектом-результатом функции SYS.XMLNode(), так и строкой с именем нового тега. Возвращаетсявложенный узел;
- *XMLNodeObj childIns(int id, ElTp no = XMLNodeObj); XMLNodeObj childIns(int id, string no)* – вставка объекта <no> как вложенного в позицию <id>. <no> может быть как непосредственно объектом-результатом функции SYS.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>;

- *XMLNodeObj childGet(string name, int num = 0)* – получает вложенный узел с именем тега <name> и порядковым номером <num>;
- *XMLNodeObj parent()* – получить родительский узел;
- *string load(string str, bool file = false, int flg = 0, string cp = "UTF-8")* – загружает XML из строки <str> или из файла с путём в <str> если <file> равно "true", с кодировкой <cp>. Где <flg> – флаги загрузки:
  - 0x01 – полная загрузка, с блоками текста и комментариями в специальных узлах;
  - 0x02 – не удалять пробелы в начале и конце текста тега;
- *string save(int flg = 0, string path = "", string cp = "UTF-8")* – сохраняет дерево XML в строку или в файл <path> с параметрами форматирования <flg> и кодировкой <cp>. Возвращает текст XML или код ошибки. Предусмотрены следующие флаги форматирования <flg>:
  - 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;
  - 0x20 – вставлять стандартный XML-заголовок;
  - 0x40 – вставлять стандартный XHTML-заголовок;
  - 0x80 – очищать сервисные теги: <??>, <!-- -->;
  - 0x100 – не кодировать наименований тегов;
  - 0x200 – не кодировать наименований атрибутов.
- *XMLNodeObj getElementBy(string val, string attr = "id")* – получить элемент из дерева по атрибуту <attr> со значением <val>;
- *TArrayObj <XMLNodeObj> getElementsBy (string tag, string attrVal = "", string attr = "id")* – получает массив элементов из дерева по тегу <tag> (пустой для всех) и атрибуту <attr> со значением <attrVal> (пустые для пропуска).

16.2 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 СКАДА 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;

- *string fileRead(string file);* – возвращает содержимое файла <*file*> в строке;
- *int fileWrite(string file, string str, bool append = false);* – записывает строку <*str*>в файл <*file*>, удаляя присутствующий файл или добавляя в него <*append*>. Возвращает количество записанных байт;
- *int fileRemove(string file);* – удаляетфайл<*file*>. Возвращает результат удаления;
- *int message(string cat, int level, string mess);* – формирование системного сообщения <*mess*>с категорией <*cat*>, уровнем <*level*>(-7...7). 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;

  Пример:

  ```
  //получение идентификатора станции
  req = SYS.XMLNode("get").setAttr("path","/%2fgen%2fid");
  SYS.cntrReq(req);
  idSt = req.text();
  ```

- *string sleep(int tm, int ntm = 0);* — приостановить поток исполнения на <*tm*> секунд и <*ntm*> наносекунд. Функция не рекомендована к использованию, особенно в процедурах пользовательского интерфейса, поскольку это приведет к блокированию интерфейса;
- *int time(int usec);* — возвращает абсолютное время в секундах от эпохи 1.1.1970 и микросекундах, если <*usec*> указан;

- *int {local time|gm time}(int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int yday, int isdst);* — возвращает полную дату и время 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. *gmtime* возвращает время в GMT(UTC);
- *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(string 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 "2017-08-08 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;
- *string strCodeConv(string src, string fromCP, string toCP);* — кодирование текста <*src*> из кодировки <*fromCP*> в <*toCP*>. Если кодировка опущена, то используется внутренняя.

16.3 Любой объект (TCntrNode) дерева СКАДА (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 СКАДА. Если указывается разделитель в <sep>, то путь обрабатывается как строка с разделителем;
- *TCntrNodeObj nodePrev();* — получить предыдущий, родительский, узел;
- *string nodePath(string sep = "", bool from_root = true);* — получение пути к текущему узлу, в дереве объектов СКАДА. Один символ разделителя указывается в <sep> для получения пути через разделитель, например, "DAQ.ModBus.PLC1.P1.var", иначе "/DAQ/ModBus/PLC1/P1/var". <from_root>

указывает на необходимость формировать путь от корня и без указания идентификатора станции;

- *int messSys(int level, string mess)* — формирует системное сообщение <mess> с уровнем <level>, с путём узла в качестве категории и с читабельным путём перед сообщением.

16.4 Подсистема "Безопасность" (SYS.Security)

Функции объекта подсистемы (SYS.Security):

- *int access(string user, int mode, string owner, string group, int access)* — проверка для пользователя <user> доступа к ресурсу, который принадлежит <owner> и <group> с доступом <access> для режима <mode>:

  user — пользователь для проверки доступа;

  mode — режим доступа (4-R, 2-W, 1-X);

  owner — владелец ресурса;

  group — группа ресурса;

  access — режим доступа к ресурсу (RWXRWXRWX — 0777).

Функции объекта пользователя (SYS.Security["usr_User"]) и группы (SYS.Security["grp_Group"]):

- *ElTp cfg(string nm)* — получение значения конфигурационного поля <*nm*> объекта;
- *bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *Array groups( )* — возвращает перечень групп пользователя;
- *bool user(string nm)* — проверяет присутствие пользователя <*nm*> в данной группе.

16.5 Подсистема "БД" (SYS.BD)

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

- *ElTp cfg(string nm)* — получение значения конфигурационного поля <*nm*> объекта;
- *Bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *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);
//> получить значение в столбце по имени
if(i_rw) SYS.messDebug("TEST DB: ","Row "+i_rw+": 'NAME'"+DBTbl[i_rw]
["NAME"]);
}
```

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

- *XMLNodeObj fieldStruct();* – получение структуры таблицы в виде XML узла "field" с дочерними узлами-колонками <RowIdtype="real" len="10.2" key="1" def="Defaultvalue">{Value}</RowId>, где:
  - {RowId} – идентификатор колонки;
  - {Value} – значение колонки;
  - type – тип значения колонки: *str* – строка, *int* – целое, *real* – вещественное и *bool* – логическое;
  - len – размер значения колонки, в знаках;
  - key – признак того, что колонка является ключом, и поиск осуществляется по его значению;
  - def – значение колонки по умолчанию.
- *string fieldSeek(int row, XMLNodeObj fld);* – Запрос поля *<row>* таблицы. Если поле получено, то возвращается "1" иначе "0". В случае ошибки возвращается "0:Error";
- *string fieldGet(XMLNodeObj fld);* – Запрос значений поля. В случае ошибки возвращается "0:Error".

Пример:

```
req = SYS.XMLNode("field");
req.childAdd("user").setAttr("type","str").setAttr("key","1"). setText("root");
req.childAdd("id").setAttr("type","str").setAttr("key","1"). setText("/Lang2CodeBase");
req.childAdd("val").setAttr("type","str");
SYS.BD.MySQL.GenDB.SYS.fieldGet(req);
SYS.messDebug("TESTDB","Value: "+req.childGet(2).text());
```

- *string fieldSet(XMLNodeObj fld);* – Установка поля. В случае ошибки возвращается "0:Error";
- *string fieldDel(XMLNodeObj fld);* – Удаление поля. В случае ошибки возвращается "0:Error".

16.6 Подсистема "Сбор данных" (SYS.DAQ)

Функции объекта подсистемы (SYS.DAQ):

- *bool funcCall(string progLang, TVarObj args, string prog);* – вызов текста функции <*prog*> с аргументами в объекте <*args*> для языка программирования <*progLang*>. Возвращает "true" при корректном вызове.

  Пример:

```
varargs = newObject();
args.y = 0;
args.x = 123;
SYS.DAQ.funcCall("JavaLikeCalc.JavaScript",args,"y=2*x;");
SYS.messDebug("TESTCalc","TESTCalcrezult: "+args.y);
```

Функции объекта контроллера (SYS.DAQ["Modul"]["Controller"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *string name()* – имя контроллера;
- *string descr()* – описание контроллера;
- *string status()* – статус контроллера;
- *bool alarmSet(string mess, int lev = -5, string prm = "")* – установка/снятие нарушения <mess> с уровнем <lev> (отрицательный для установки, иначе снятие), для параметра <prm>. Функция формирует нарушение с категорией: al{ModId}:{CntrId}[.{PrmId}], где:
  - *ModId* — идентификатор модуля;
  - *CntrId* — идентификатор контроллера;
  - *PrmId* — идентификатор параметра, из аргумента <*prm*>;
- *bool enable(bool newSt = EVAL)* – получение состояния "Включен" или изменение его назначением атрибута <*newSt*>;
- *bool start(bool newSt = EVAL)* – получение состояния "Запущен" или изменение его назначением атрибута <*newSt*>.

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

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <*nm*> объекта;
- *ElTp cfgSet(string nm, ElTp val)* – установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *TCntrNodeObj cntr()* – возвращает объект контроллера этого параметра, независимо от вложенности.

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>;
- *TCntrNodeObj arch()* – получение объекта архива, связанного с этим атрибутом. В случае отсутствия связанного архива возвращается "false";
- *string descr()* – описание атрибута;
- *int time(int utm)* — время последнего значения в секундах и микросекундах в <utm>;
- *int len()* – длина поля;
- *int dec()* – разрешение для вещественного;
- *int flg()* – флаги поля
- *string def()* – значение по умолчанию;
- *string values()* – список допустимых значений или диапазон;
- *string selNames()* – список имён допустимых значений;
- *string reserve()* – резервное свойство значения.

Функции объекта библиотеки шаблона (SYS.DAQ[tmplb_Lib"]) и шаблона (SYS.DAQ[tmplb_Lib"] ["Tmpl"]) параметра контроллера:

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>.

16.6.1 *Модуль DAQ.JavaLikeCalc*

Объект "Библиотека функций" (SYS.DAQ.JavaLikeCalc["lib_Lfunc"])

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.DAQ.JavaLikeCalc["lib_Lfunc"]["func"])

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами *<prm{N}>*. Возвращает результат вызываемой функции.

16.6.2 Модуль DAQ.ModBus

Объект "Контроллер" (this.cntr())

- *string messIO(string pdu)* – отправка PDU <pdu> через транспорт объекта контроллера посредством ModBus протокола. PDU результата запроса помещается вместо запроса в <pdu>, а ошибка возвращается в результате функции.

Объект «Параметр» (this)

- *bool attrAdd(string id, string name, string tp = "real", string selValsNms = "")* [для включенного параметра логического типа] – добавление атрибута <id> с именем <name> и типом <tp>. Если атрибут уже присутствует, то будут применены свойства, которые возможно изменить "на ходу": имя, режим выбора и параметры выбора.
  - id, name – идентификатор и имя нового атрибута;
  - tp – тип атрибута [boolean | integer | real | string | text | object] + режим выбора [sel | seled] + только для чтения [ro];
  - selValsNms – две строки со значениями в первой и их именами во второй, разделённые ";";
- *bool attrDel(string id)* [для включенного параметра логического типа] – удаление атрибута <id>.

16.7 Подсистема "Архивы" (SYS.Archive)

Функции объекта подсистемы:

- *Array 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>. Возвращается массив объектов сообщений со свойствами:
  - *tm* – время сообщения, секунды;
  - *utm* – время сообщения, микросекунды;

- *categ* – категория сообщения;
- *level* – уровень сообщения;
- *mess* – текст сообщения.

- *bool messPut(int tm, int utm, string cat, int lev, string mess);* – запись сообщения <mess> с категорией <cat>, уровнем <lev> (-7...7) и временем <tm>.<utm> в архив и/или список нарушений.

Функции объекта архиватора сообщений (SYS.Archive["mod_Modul"]["mess_Archivator"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *bool status()* – получение статуса архиватора;
- *int end()* – поучение времени окончания данных архиватора;
- *int begin()* – поучение времени начала данных архиватора.

Функции объекта архиватора значений (SYS.Archive["val_Modul"]["val_Archivator"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *bool status()* – получение статуса архиватора «Исполнение».

Функции объекта архива (SYS.Archive["va_Archive"]):

- *ElTp cfg (string nm)* – получает значение конфигурационного поля <nm> объекта;
- *bool cfgSet (string nm, ElTp val)* – устанавливает конфигурационное поля <nm> объекта в значение <val>;
- *bool status ()* – статус архиватора "Исполнение";
- *int end (string arch = "")* – время конца данных архива для архиватора <arch>, в микросекундах;
- *int begin (string arch = "")* – время начала данных архива для архиватора <arch>, в микросекундах;

- *int period (string arch = "")* – период данных архива для архиватора <arch>, в микросекундах;
- *TArrayObj archivatorList ()* – список архиваторов, использующих данный архив как источник;
- *VarType getVal (int tm, bool up_ord = false, string arch = "")* – получает значение из архива на время <tm>, подтянув к верху <up_ord> и архиватор <arch>:
  - tm – время запрашиваемого значения, в микросекундах, установить в 0 для "end()"; этот атрибут также является выходом, соответственно реальное время полученного значения помещается сюда, если это переменная;
  - up_ord – подтягивать время запрашиваемого значения к верху сетки;
  - arch – архиватор запроса, установить в пустую строку для проверки всех архиваторов, установить в "<buffer>" для обработки только буфера.
- *bool setVal (int tm, VarType vl, string arch = "")* – устанавливает значение <vl> в архив на время <tm> и архиватор <arch>:
  - tm – время устанавливаемого значения, в микросекундах;
  - vl – значение;
  - arch – архиватор установки, установить в пустую строку для всех архиваторов, установить в "<buffer>" для обработки только буфера.

16.8 Подсистема "Транспорты" (SYS.Transport)

Функции объекта входящего транспорта (SYS.Transport["Modul"]["in_Transp"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *string status()* – строка статуса транспорта;
- *string addr(string vl = "")* – адрес транспорта, устанавливает в непустое значение <vl>;
- *string writeTo(string sender, string mess)* – отправляет сообщение <mess> отправителю <sender>, как ответ;
- *TArrayObj assTrsList( )* – список связанных выходных транспортов с данным входящим.

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

- *ElTp cfg(string nm)* — получение значения конфигурационного поля *<nm>* объекта;

- *bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *string status()* — строка статуса транспорта;
- *bool start(boolvl = EVAL, inttm = 0)* — установка статуса транспорта «Запущен», включить/остановить его для значения <vl> (если оно не EVAL). Возможно установить таймаут <tm>;
- *string addr(string vl = "")* — адрес транспорта, устанавливает в непустое значение <*vl*>;
- *string timings(string vl = "")* —синхронизация транспорта, устанавливает в непустое значение <*vl*>;
- *int attempts(int vl = EVAL)* — попытка соединения транспорта, устанавливает в непустое значение <*vl*>;
- *string messIO(string mess, real timeOut = 0);* — отправка сообщения <*mess*> через транспорт с таймаутом ожидания <*timeOut*> (в секундах). В случае нулевого таймаута это время берётся из настроек исходящего транспорта.

  Пример:

```
rez=SYS.Transport.Serial.out_ttyUSB0.messIO(SYS.strFromCharCode(0x4B,
0x00,0x37,0x40),0.2);
while(true)
{
   trez = SYS.Transport.Serial.out_ttyUSB0.messIO("");
   if( !trez.length ) break;
   rez+=trez;
   }
```

- *int messIO(XMLNodeObj req, string prt);* — отправка запроса <*req*> к протоколу <*prt*> для осуществления сеанса связи через транспорт посредством протокола. Пример:

```
req = SYS.XMLNode("TCP");
req.setAttr("id","test").setAttr("reqTm",500).setAttr("node",1).
setAttr("reqTry",2).setText(SYS.strFromCharCode(0x03,0x00,0x00, 0x00,0x05));
SYS.Transport.Sockets.out_testModBus.messIO(req,"ModBus");
test = Special.FLibSYS.strDec4Bin(req.text());
```

16.9 Подсистема "Пользовательские интерфейсы" (SYS.UI)

16.9.1 Модуль UI.VCAEngine

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*>.

Пример:

```
//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)* – удалениевиджета<wid>;
- *TCntrNodeObj wdgAt(string wid, bool byPath = false)* – подключение к дочернему виджету или глобальному посредством пути <byPath>. В случае глобального подключения можно использовать абсолютный или относительный путь к виджету. Точкой отсчёта абсолютного адреса выступает объект корня модуля "VCAEngine", а значит первым элементом абсолютного адреса является идентификатор сеанса, который опускается. Относительный адрес берёт отсчёт от текущего виджета. Специальным элементом относительного адреса является элемент вышестоящего узла "...";
- *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.

  Пример:

```
//Установить ссылку восьмого тренда параметром
this.linkSet("el8.name", "prm:/LogicLev/experiment/Pi", true);
```

Объект "Виджет", примитива "Документ" (this)

string getArhDoc(integer nDoc) — получить текст документа архива на глубине *<nDoc>* (0-{aSize-1}).

16.10 Подсистема "Специальные" (SYS.Special)

16.10.1 Модуль Special.FLibSYS

Объект "Библиотека функций" (SYS.Special.FLibSYS):

ElTp {funcID}(ElTp prm1, ...) — вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibSYS["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами *<prm{N}>*. Возвращает результат вызываемой функции.

16.10.2 Модуль Special.FLibMath

Объект "Библиотека функций" (SYS.Special.FLibMath):

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibMath["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами <*prm{N}*>. Возвращает результат вызываемой функции.

16.10.3 Модуль Special.FLibComplex1

Объект "Библиотека функций" (SYS.Special.FLibComplex1):

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibComplex1["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами <*prm{N}*>. Возвращает результат вызываемой функции.

17. ОПИСАНИЕ JAVA-ПОДОБНОГО ЯЗЫКА

17.1 Элементы языка

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

Константы:

- 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", без переноса на другую строку, однако с поддержкой прямой конкатенации строковых констант.

Типы переменных:

- целое: -231...231, EVAL_INT(-2147483647);
- вещественное: 3.4 * 10308, EVAL_REAL(-3.3E308);
- логическое: false, true, EVAL_BOOL(2);
- строка: последовательность символов-байтов (0...255) любой длины, ограниченной объёмом памяти и хранилищем в БД;
- EVAL_STR("<EVAL>").

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

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

17.2 Операции языка

Операции, поддерживаемые языком:

Символ	Описание
()	Вызов функции
{}	Программные блоки
++	Инкремент
--	Декремент
-	Вычитание, унарный минус
+	Сложение
!	Логическое отрицание
~	Побитовое отрицание
*	Умножение
/	Деление
%	Остаток от целочисленного деления
<<	Поразрядный сдвиг влево
>>	Поразрядный сдвиг вправо
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно
|	Поразрядное "или"
&	Поразрядное "и"
^	Поразрядное "исключающее или"
&&	Логическое "и"
||	Логическое "или"
?:	Условная операция (i=(i<0)?0:i;)
=	Присваивание
+=	Присваивание со сложением
-=	Присваивание с вычитанием

*=	Присваивание с умножением
/=	Присваивание с делением

17.3 Встроенные функции языка

Синтаксис функции	**Описание функции**
double max(double x, double x1)	Максимум из x и x1
double min(double x, double x1)	Минимум из x и x1
string typeof(E1tp v1)	Тип значения v1
double sin(double x)	Синус
double cos(double x)	Косинус
double tan(double x)	Тангенс
double sinh(double x)	Синус гиперболический
double cosh(double x)	Косинус гиперболический
double tanh(double x)	Тангенс гиперболический
double asin(double x)	Арксинус
double acos(double x)	Арккосинус
double atan(double x)	Арктангенс
double rand(double x)	Случайное число от 0 до x
double lg(double x)	Десятичный логарифм
double ln(double x)	Натуральный логарифм
double exp(double x)	Экспонента
double pow(double x, double x1)	x в степени x1
double sqrt(double x)	Квадратный корень
double abs(double x)	Модуль x
double sing(double x)	Знак x
double ceil(double x)	Округление до большего целого
double floor(double x)	Округление до меньшего целого

17.4 Операторы языка

- условие внутри выражения: <st_open=(pos>100)?true:false;>;
- глобальное условие if: <if (st_open>100) pos=true; else pos=false>;
- цикл while: while (<условие>) <тело_цикла>;
- цикл for: for (<пре_условие>;<анализ>;<пост_вычисление>)<тело_цикла>;

- цикл for-in: for(<переменная> in<объект>)<тело_цикла>;
- прерывание выполнения цикла: break;
- продолжение выполнения цикла с начала: continue;
- установка области видимости часто используемой библиотеки: using;
- прерывание функции и возврат результата: return;
- создание объекта: new.

17.4.1 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.

17.4.2 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.

17.4.3 Специальные символы строковых переменных

Символ	Описание
\n	Перевод строки
\t	Символ табуляции
\b	Забой
\f	Перевод страницы
\r	Возврат каретки
\\	Символ "\"
\041	"!" восьмеричный
\x21	"!" шестнадцатиричный

17.5 Общесистемные функции

sysCall - Вызов консольных команд и утилит операционной системы.

Описание: Осуществляет вызовы консольных команд ОС. Функция открывает широкие возможности пользователю СКАДА 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);
```

UUID – код UUID.

Описание: Формирование уникального постоянного идентификатора раздела жесткого диска.

Параметры:

	Имя	Тип	Реж	По умолчанию
	UUID	строка	Возв	

dbReqSQL - SQL запрос.

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	
trans	Транзакция	Логический	Вход	2

Для закрытия транзакции после выполнения запроса необходимо установить значение параметра trans =1. По умолчанию транзакция остается открытой.

dbImportLO – импорт большого объекта в БД.

Описание: используется для импорта большого объекта в БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
addr	Адрес БД	Строка	Вход	
data	Дата	Строка	полная	

dbExportLO – экспорт большого объекта в БД.

Описание: используется для экспорта большого объекта в БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
addr	Адрес БД	Строка	Вход	
id	ID	Строка	Вход	

dbRemoveLO –удаление большого объекта из БД.

Описание: используется для удаления большого объекта из БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Логический	Возврат	
addr	Адрес БД	Строка	Вход	
id	ID	Строка	Вход	

xmlNode- узел XML.

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");
```

xmlCntrReq - запрос интерфейса управления.

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:

```
usingSpecial.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);
```

vArh – архив значений.

Описание: Получение объекта архива значений (VArchObj) путём подключения к архиву по его адресу.

Параметры:

ID	Имя	Тип	Режим
rez	Результат	Объект(VArchObj)	Возврат
name	Имя, адрес к атрибуту параметра с архивом (DAQ.{Module}.{Cntr}.{Prm}.{Attr}) или непосредственно к архиву значений (Archive.{ValArchive}).	Строка	Вход

17.5.1 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;
s = 0.0;
us = 0.0;
pathAttr = "DAQ.LogicLev.cntr.prm.value";
//получение времени начала архива
s = vArch(pathAttr).begin(us,"DBArch.ArchiveChange");
time_begin = s*1.0e6 + us;
//получение времени окончания архива
s = vArch(pathAttr).end(us,"DBArch.ArchiveChange");
time_end = s*1.0e6 + us;
//пример получения значения
val = vArch(pathAttr).get(s,us,1,"DBArch.ArchiveChange");
//пример получения периода архива
s = vArch(pathAttr).period(us,"DBArch.ArchiveChange");
T = s*1.0e6 + us;
//пример установки значения
s = SYS.tmTime(0);
us = 0.0;
val = 3.14;
vArch(pathAttr).set(val, s, us);
```

vArcBuf – 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-	Целый	Вход	1
sz	Максимальный размер буфера	Целый	Вход	100
per	Периодичность буфера (в микросекундах)	Целый	Вход	1000000

hgrd	Режим «Жесткая сетка	Логический	Вход	0
hres	Режим “Высокого разрешения времени (микросекунды)”	Логический	Вход	0

17.5.2 Функции работы с астрономическим временем

tmFStr - Строка времени.

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

Пример:

```
using Special.FLibSYS;
test=tmFStr(SYS.time(),"%d %m %Y");
messPut("Example",0,"tmFStr(): "+test);
```

tmDate – полная дата.

Описание: Возвращает полную дату в секундах, минутах, часах и т.д, исходя из абсолютного времени в секундах от 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);
```

tmTime - абсолютное время.

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

tmStr2Tm – конвертация строки во время

Описание: Возвращает время в секундах от 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».

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

tmCron - планирование времени в формате Cron.

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

17.5.3 Функции работы с сообщениями

messGet - запрос системных сообщений.

Параметры:

ID	Параметр	Тип	Режим	По
rez	Результат	Объект(Массив	Возврат	
btm	Время начала	Целое	Вход	
etm	Время конца	Целое	Вход	
cat	Категория	Строка	Вход	
lev	Уровень	Целый	Вход	
arch	Архиватор	Строка	Вход	

messPut - генерация сообщения.

Описание: Формирование системного сообщения.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
time	время	Строка	вход	
cat	Категория	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);
```

messPut2 - генерация сообщения с отметкой времени.

Описание: поместить собщение с отметкой времени в систему.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
time	время	Строка	вход	
cat	Категория	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	
mess	Текст сообщения	Строка	Вход	

17.5.4 Функции работы со строками

strSize - получение размера строки.

Описание: Используется для получения размера строки.

Параметры*:*

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целый	Возврат	
str	Строка	Строка	Вход	

strSize8 - получение строки (UTF8).

Описание: Используется для получения размера строки UTF8.

Параметры*:*

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целый	Возврат	
str	Строка	Строка	Вход	

strSubstr – получение части строки.

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);
```

strInsert – вставка одной строки в другую.

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	Вход	

-

strReplace - замена части строки другой.

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	Вход	

strParse - разбор строки по разделителю.

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	

strParsePath – разбор пути на части.

Описание: Используется для разбора 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	

strPath2Sep- путь к разделенной строке.

Описание: Используется для преобразования пути в разделенную строку.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Ресурс	Строка	Вход	
sep	Сепаратор	Строка	Вход	"."

strEnc2HTML – преобразование строки в кодировку HTML.

Описание: Используется для кодирования строки в HTML.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

strEnc2Bin - кодирование текста в бинарный вид.

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	

strDec4Bin – декодирование текста из бинарного вида.

Описание: Используется для декодирования текста из бинарного вида в формат <00 A0 FA DE>.

Параметры:

ID	Парамет	Тип	Режим	По умолчанию
rez	Результа	Строка	Возврат	
src	Источник	Строка	Вход	

-

real2str - преобразование вещественного в строку.

Описание: Используется для преобразования вещественного в строку.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результа	Строка	Возврат	
val	Значение	Вещественное	Вход	
prc	Точность	Целое	Вход	4
tp	Тип	Строка	Вход	“f”

int2str - преобразование целого в строку.

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

str2real–преобразование строки в вещественное.

Описание: Используется для преобразования строки в вещественное.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результа	Вещественно	Возврат	
val	Значение	Строка	Вход	

str2int - преобразование строки в целое.

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

17.5.5 Функции работы с вещественным

floatSplitWord - разделение float на слова.

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	

floatMergeWord - объединение float из слов.

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	

doubleSplitWord – разделение на слова.

Описание: Разделение double (8 байт) на слова (2байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	
w3	Слово 3	Целый	Вход	
w4	Слово 4	Целый	Вход	

doubleMergeWord – объединение double из слов.

Описание: Объединение double (8 байт) из слов (2байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	
w3	Слово 3	Целый	Вход	
w4	Слово 4	Целый	Вход	

CRC – объединение float из слов.

Описание: Используется для создания унифицированного кода (CRC) шириной 8-64 бит.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целое	Возврат	
data	Дата	Строка	Вход	
poly	Polynomial (reversion)	Целый	Вход	40961
wight	Ширина	Целый	Вход	16
init	Идентификатор	Целый	Вход	-1

ПРИЛОЖЕНИЕ 1

Пример листинга файла dac.xml

```
<?xml version="1.0" encoding="utf8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation='dac.xsd' codepage="utf8">

    <dirs>
        <dir name="etc" path="etc" />
        <dir name="log" path="log" />
        <dir name="tmp" path="tmp" />
    </dirs>

    <view x="5" y="25" width="1024" height="600" />

    <debug level="8" wnd="no" log="no" />

    <snmphosts>
        <host id="100001" addr="127.0.0.1" /> <!-- ServerCpuMain -->

        <host id="100003" addr="199.168.190.22" />
  <!--MOXA Modbus1-->

    </snmphosts>

    <threads>
        <thread type="snmp" tupdt="1000" mupdt="60000"
caption="datathread_snmp" logid="001"/>

    </threads>

    <network id="001" port_r="3725" port_t="3726" logid="002" />  <!--080-->

    <clients>
        <client id="055" addr_b="127.0.0.1" addr_r="127.0.0.1" caption="dsc"/>
      <client id="056" addr_b="199.168.190.16" addr_r="199.168.190.16"
caption="dsc"/>
```

```
</clients>

<logs>
        <log id="001" caption="snmpthread"/>
        <log id="002" caption="udpserver"/>
</logs>

<base>
<!-- ServerCpuMain 100001 -->
        <!--  snmpd connect -->
        <snmp                   id        = "100001000"
                        ptype  = "value"
                        vtype   = "short"
                        addr    = ".1.3.6.1.2.1.1.3.0"
                        fval     = "1"
                        gval    = "0"
                        rexp    = ".*\s+"
                        update  = "1000" >
        </snmp>

        <!--  ipHCInOctets(loadnetwork) -->
        <snmp                   id        = "100001004"
                        ptype  = "value"
                        vtype   = "string"
                        addr    = ".1.3.6.1.2.1.31.1.1.1.2.4"
                        rexp    = ".*\s+"
                        sdep    = "100001000"
                        update  = "10000" >
        </snmp>
        <!-- ipHCOutOctets(loadnetwork) -->
        <snmp                   id        = "100001005"
                        ptype  = "value"
                        vtype   = "string"
                        addr    = ".1.3.6.1.2.1.31.1.1.1.15.4"
                        rexp    = ".*\s+"
                        sdep    = "100001000"
```

```
                update  = "10000" >
</snmp>

<!-- physicalIoial1Mb -->
<snmp                   id      = "100001008"
                ptype  = "value"
                vtype   = "double"
                rexp    = ".*\s+"
                addr    = ".1.3.6.1.2.1.25.2.3.1.5.1"
                sdep    = "100001000"
                math    = "* ;1024 / ;1024 / ;1024"
                send    = "no"
                update  = "10000" >
</snmp>

<!-- physicalUsed1Mb -->
<snmp                   id      = "100001010"
                ptype  = "value"
                vtype   = "double"
                rexp    = ".*\s+"
                addr    = ".1.3.6.1.2.1.25.2.3.1.6.1"
                sdep    = "100001000"
                math    = "* ;1024 / ;1024 / ;1024"
                send    = "no"
                update  = "10000" >
</snmp>

<!-- physicalFree1Mb -->
<calc           id      = "100001011"
                ptype  = "value"
                vtype   = "double"
                math    = ",100001008 -,100001010" >
</calc>
<!-- discIoial1Mb -->
<snmp                   id      = "100001012"
                ptype  = "value"
```

```
                    vtype   = "double"
                    rexp    = ".*\s+"
                    addr    = ".1.3.6.1.2.1.25.2.3.1.5.32"
                    sdep    = "100001000"
                    math    = "* ;4096 / ;1024 / ;1024"
                    send    = "no"
                    update  = "10000" >
            </snmp>

            <!-- discUsed1Mb -->
            <snmp               id      = "100001013"
                    ptype   = "value"
                    vtype   = "double"
                    rexp    = ".*\s+"
                    addr    = ".1.3.6.1.2.1.25.2.3.1.6.32"
                    sdep    = "100001000"
                    math    = "* ;4096 / ;1024 / ;1024"
                    send    = "no"
                    update  = "10000" >
            </snmp>

            <!-- discFree 1Mb -->
            <calc       id      = "100001014"
                    ptype   = "value"
                    vtype   = "double"
                    math    = ",100001012 -,100001013" >
            </calc>

        <!-- MOXA Modbus1 -->
<!--  snmpd connect -->
            <snmp               id      = "100003000"
                    ptype   = "value"
                    vtype   = "short"
                    addr    = ".1.3.6.1.2.1.1.3.0"
                    fval    = "1"
                    gval    = "0"
```

```
                        rexp    = ".*\s+"
                        update  = "1000" >
                </snmp>
<!-- ifOperStatus1 -->
                <snmp                   id      = "100003001"
                        ptype  = "value"
                        vtype   = "short"
                        rexp    = ".*\s+"
                        addr    = ".1.3.6.1.2.1.2.2.1.8.1"
                        sdep    = "100003000"
                        update  = "5000" >

                        <rule key="1" value="0" />
                        <rule key="2" value="1" />
                        <rule key="3" value="1" />

                </snmp>
<!-- ifOperStatus2 -->
                <snmp                   id      = "100003002"
                        ptype  = "value"
                        vtype   = "short"
                        rexp    = ".*\s+"
                        addr    = ".1.3.6.1.2.1.2.2.1.8.2"
                        sdep    = "100003000"
                        update  = "5000" >

                        <rule key="1" value="0" />
                        <rule key="2" value="1" />
                        <rule key="3" value="1" />

                </snmp>
<!-- ifOperStatus3 -->
                <snmp                   id      = "100003003"
                        ptype  = "value"
                        vtype   = "short"
                        rexp    = ".*\s+"
```

```
                        addr    = ".1.3.6.1.2.1.2.2.1.8.3"
                        sdep    = "100003000"
                        update  = "5000" >

                <rule key="1" value="0" />
                <rule key="2" value="1" />
                <rule key="3" value="1" />

        </snmp>
<!-- ifOperStatus4 -->
        <snmp                   id      = "100003004"
                        ptype  = "value"
                        vtype   = "short"
                        rexp    = ".*\s+"
                        addr    = ".1.3.6.1.2.1.2.2.1.8.4"
                        sdep    = "100003000"
                        update  = "5000" >

                <rule key="1" value="0" />
                <rule key="2" value="1" />
                <rule key="3" value="1" />

        </snmp>

    </base>
</config>
```

ПРИЛОЖЕНИЕ 2

Пример листинга файла dsc.xml

```
<?xml version="1.0" encoding="utf8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation='dsc.xsd' codepage="utf8">

    <dirs>
        <dir name="etc" path="etc" />
        <dir name="log" path="log" />
        <dir name="tmp" path="tmp" />
    </dirs>

    <debug level="8" log="no" />

    <network_c id="055" port_r="3726" port_t="3725" updt="500" lupdt="1000" vupdt="1000" logid="001" />
    <network_s id="000" port_r="3728" port_t="3727" logid="002" />

    <servers>

        <server id="001" addr_b="127.0.0.1" addr_r="127.0.0.1" caption="ServerCpuMain" />
        <server id="081" addr_b="199.168.190.16" addr_r="199.168.190.16" caption="ServerCpuRes" />
        <server id="005" addr_b="199.168.190.28" addr_r="199.168.190.28" caption="ArmIngener" />
        <server id="006" addr_b="199.168.190.34" addr_r="199.168.190.34" caption="ArmAsn" />

    </servers>

    <clients>
        <client id="901" addr_b="127.0.0.1" addr_r="127.0.0.1" port="3724" caption="scada" />
    </clients>
```

```
        <logs>
                <log id="001" caption="udpclient"/>
                <log id="002" caption="udpserver"/>
        </logs>

        <base>

<!-- ServerCpuMain -->
        <!-- PU 100001 -->
                <through id="100001000" serv="001" vtype="short"    />
                <through id="100001004" serv="001" vtype="string"   />
                <through id="100001005" serv="001" vtype="string"   />
                <through id="100001008" serv="001" vtype="double"   />
                <through id="100001010" serv="001" vtype="double"   />
                <through id="100001011" serv="001" vtype="double"   />
                <through id="100001012" serv="001" vtype="double"   />
                <through id="100001013" serv="001" vtype="double"   />
                <through id="100001014" serv="001" vtype="double"   />

<!--  MOXA Modbus1  -->

                <through id="100003000" serv="001" vtype="short"    />
                <through id="100003001" serv="001" vtype="short"    />
                <through id="100003002" serv="001" vtype="short"    />
                <through id="100003003" serv="001" vtype="short"    />
                <through id="100003004" serv="001" vtype="short"    />

<!-- ServerCPUres -->

                <through id="101181000" serv="081" vtype="short"    />
                <through id="101181004" serv="081" vtype="string"   />
                <through id="101181005" serv="081" vtype="string"   />
                <through id="101181008" serv="081" vtype="double"   />
                <through id="101181010" serv="081" vtype="double"   />
                <through id="101181011" serv="081" vtype="double"   />
                <through id="101181012" serv="081" vtype="double"   />
```

```
            <through id="101181013" serv="081" vtype="double"  />
            <through id="101181014" serv="081" vtype="double"  />

<!-- ARmIngener -->

            <through id="100005000" serv="005" vtype="short"    />
            <through id="100005004" serv="005" vtype="string"   />
            <through id="100005005" serv="005" vtype="string"   />
            <through id="100005008" serv="005" vtype="double"  />
            <through id="100005010" serv="005" vtype="double"  />
            <through id="100005011" serv="005" vtype="double"  />
            <through id="100005012" serv="005" vtype="double"  />
            <through id="100005013" serv="005" vtype="double"  />
            <through id="100005014" serv="005" vtype="double"  />
            <through id="100005015" serv="005" vtype="short"    />
            <through id="100005016" serv="005" vtype="short"    />
            <through id="100005017" serv="005" vtype="string"   />
            <through id="100005018" serv="005" vtype="string"   />
            <through id="100005019" serv="005" vtype="string"   />
            <through id="100005020" serv="005" vtype="string"   />

<!-- ArmAsn -->
            <through id="100006000" serv="006" vtype="short"    />
            <through id="100006004" serv="006" vtype="string"   />
            <through id="100006005" serv="006" vtype="string"   />
            <through id="100006008" serv="006" vtype="double"  />
            <through id="100006010" serv="006" vtype="double"  />
            <through id="100006011" serv="006" vtype="double"  />
            <through id="100006012" serv="006" vtype="double"  />
            <through id="100006013" serv="006" vtype="double"  />
            <through id="100006014" serv="006" vtype="double"  />
            <through id="100006015" serv="006" vtype="short"    />
            <through id="100006016" serv="006" vtype="short"    />
            <through id="100006017" serv="006" vtype="string"   />
            <through id="100006018" serv="006" vtype="string"   />
            <through id="100006019" serv="006" vtype="string"   />
```

```
                <through id="100006020" serv="006" vtype="string"    />

        </base>

</config>
```

Коды единиц измерения

Код	Единицы измерения
6	PSI
7	Бар
11	Паскаль
12	Килопаскаль
16	Галлон/Минуту
17	Литр / Минуту
19	Метр кубический/Час
20	Feet / Секунду
21	Метр / Секунду
22	Галлон / Секунду
24	Литр / Секунду
26	Feet кубический / Секунду
27	Feet кубический / День
32	Градус Цельсия
33	Градус Фаренгейта
35	Кельвин
36	Милливольт
38	Вольт
40	Галлон
41	Литр
43	Метр Кубический
50	в минуту
51	в секунду
52	в час
53	в день
112	Feet кубический
130	Feet кубический / Час
131	Feet кубический / Мин
132	Баррель/Секунду
133	Баррель/Минуту
134	Баррель/Час

135	Баррель/День
136	Галлон/Час
138	Литр/Час
235	Галлон/День
237	Мегапаскаль
246	Литр/День
247	Децибел

Расшифровка статусов команд модуля HART

Статус	Тип статуса	Расшифровка
passed	1	Выполнено успешно
Passed parameter too small	1	Введенный параметр слишком мал
Write-protect mode	1	Режим защиты от записи
Set to nearest possible value	1	Установлено в ближайшее возможное значение
Update in progress	1	Обновление в процессе
Not in proper current mode (fixed at 4 mA or 20 mA)	1	Не верное значение тока (фиксировано от 4 до 20 мА)
In multidrop mode	1	В multidrop режиме
Applied process too high	1	Значение слишком велико
Applied process too low	1	Значение слишком мало
Lower range value too low	1	Нижний предел слишком мал
Lower range value too high	1	Нижний предел слишком высок
Upper range value too high	1	Верхний предел слишком высок
Upper range value too low	1	Верхний предел слишком мал
Both range values out of limits	1	Оба предела выходят за диапазон
Span too small	1	Значение слишком мало
Pushed upper range value over limit	1	Верхнее значение вышло за диапазон
Command not implemented	1	Команда не разработана
Field device malfunction	2	Неисправность HART-устройства
Configuration changed	2	Конфигурация изменена
Cold start	2	Ожидается запуск
More status available	2	Больше статуса доступно (в 48 команде)
Analogue output current fixed	2	Значение тока установлено

Статус	Тип статуса	Расшифровка
Analogue output saturated	2	Значение тока вернулось в нормальный режим
Non primary variable out of limits	2	Не основная переменная вышла за границу диапазона
Primary variable out of limits	2	Основная переменная вышла за границу диапазона

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД	база данных
ОЗУ	оперативное запоминающее устройство
ОС	операционная система
ПО	программное обеспечение
API	application programming interface (программный интерфейс приложения)
HTML	язык гипертекстовой разметки (Hypertext Mark-up Language)
SCADA	диспетчерское управление и сбор данных (Supervisory Control And Data Acquisition)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					