Быстрый старт OpenSCADA

Роман Савоченко (rom_as@oscada.org)

Максим Лысенко (mlisenko@oscada.org)

8. июн. 2012

Оглавление

Введение

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

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

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

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

1. Термины, определения и аббревиатуры

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

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

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

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

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

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

- *Световая сигнализация* — обычно осуществляется посредством смены цвета графического объекта (миганием) для возникающих событий и установкой статичных аварийных цветов (красный и жёлтый) для сквитированных событий.
- *Звуковая* — осуществляется выдачей звукового сигнала в момент возникновения события. Тип звукового сигнала может быть как монотонным, так и синтезированным речевым сообщением с информацией о нарушении.

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

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

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

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

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

2. Установка

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

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

Если у пользователя вызывает затруднение установка не только OpenSCADA, но и дистрибутива Linux, то на первое время он может воспользоваться "живым" дистрибутивом Linux, с установленной и готовой для работы и изучения демонстрацией OpenSCADA. На данный момент доступны "живые" сборки на основе дистрибутива ALTLinux в виде CD и Flash-образов, на странице: http://oscada.org/ru/glavnaja/zagruzit. Детальнее смотрите в разделе "Рецепты".

> ⚠ Динамическая модель компрессорной станции, на 6 газовых компрессоров, которая лежит в основе демонстрационной БД, требует значительных вычислительных ресурсов, а именно процессора с частотой более 1 ГГц. Данные ресурсы требуются именно для динамической модели и не являются общим показателем ресурсоёмкости программы в конечных задачах!

2.1. Установка OpenSCADA из готовых пакетов

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

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

Проверить на наличие пакетов OpenSCADA в репозиториях используемого дистрибутива Linux, а также загрузить пакеты OpenSCADA для ручной установки можно на странице загрузки официального сайта OpenSCADA (http://oscada.org/ru/zagruzka).

> ⚠ Загружать пакеты нужно непосредственно для версии используемого дистрибутива, иначе при установке могут возникнут неразрешимые проблемы с зависимостями.

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

Для ручной установки пакетов OpenSCADA загрузим их из официального сайта или другого источника. Загрузить можно два набора пакетов.

Первый набор представлен девятью пакетами:

- **openscada** — пакет со всеми файлами, нужными для запуска OpenSCADA в полном объёме, включая все модули;
- **openscada-LibDB.Main** — основные библиотеки OpenSCADA для сбора данных и другого в БД SQLite;
- **openscada-LibDB.VCA** — библиотеки визуальных компонетов в БД SQLite;
- **openscada-Model.AGLKS** — БД и конфигурация модели "АГЛКС" (Демо: EN,RU,UK);
- **openscada-Model.Boiler** — БД и конфигурация модели "Котёл" (EN,RU,UK);

- **openscada-docEN** — документация на систему OpenSCADA - Английский язык;
- **openscada-docRU** — документация на систему OpenSCADA - Русский язык;
- **openscada-docUK** — документация на систему OpenSCADA - Украинский язык;
- **openscada-devel** — пакеты разработки, для отдельного создания модулей к системе OpenSCADA.

Второй набор представлен порядка пятьюдесятью пакетами, с выделением модулей OpenSCADA в отдельные пакеты:

- **openscada-core** — содержит ядро OpenSCADA, базовую конфигурацию и запускающие файлы;
- **openscada-DB.*** — модули подсистемы "БД";
- **openscada-DAQ.*** — модули подсистемы "Сбор данных";
- **openscada-Archive.*** — модули подсистемы "Архивы";
- **openscada-Transport.*** — модули подсистемы "Транспорты";
- **openscada-Protocol.*** — модули подсистемы "Транспортные протоколы";
- **openscada-UI.*** — модули подсистемы "Пользовательские интерфейсы";
- **openscada-Special.*** — модули подсистемы "Специальные";
- **openscada-LibDB.Main** — основные библиотеки OpenSCADA для сбора данных и другого в БД SQLite;
- **openscada-LibDB.VCA** — библиотеки визуальных компонентов в БД SQLite;
- **openscada-Model.AGLKS** — БД и конфигурация модели "АГЛКС" (Демо: EN,RU,UK);
- **openscada-Model.Boiler** — БД и конфигурация модели "Котёл" (EN,RU,UK);
- **openscada-docEN** — документация на систему OpenSCADA - Английский язык;
- **openscada-docRU** — документация на систему OpenSCADA - Русский язык;
- **openscada-docUK** — документация на систему OpenSCADA - Украинский язык;
- **openscada-devel** — пакеты разработки, для отдельного создания модулей к системе OpenSCADA;
- **openscada** — виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA;
- **openscada-plc** — виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как ПЛК;
- **openscada-server** — виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как SCADA-сервер;
- **openscada-visStation** — виртуальный пакет, содержащий зависимости, для установки типовой конфигурации OpenSCADA как визуальная SCADA-станция.

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

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

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

```
# rpm -i openscada-LibDB.Main-0.8.0-alt1.noarch.rpm openscada-LibDB.VCA-0.8.0-
alt1.noarch.rpm openscada-Model.AGLKS-0.8.0-alt1.i586.rpm openscada-0.8.0-
alt1.i586.rpm
```

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

```
# dpkg -i openscada-libdb.main-0.8.0-1_all.deb openscada-libdb.vca-0.8.0-
1_all.deb openscada-model.aglks-0.8.0-1_all.deb openscada_0.8.0-1_i386.deb
```

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

2.2. Установка из исходных текстов

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

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

3. Первичная конфигурация и запуск

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

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

Запустить на исполнение OpenSCADA с БД модели "АГЛКС" можно из меню окружения рабочего стола в разделе "Графика", пункт "Модель 'АГЛКС' на открытой системе визуального контроля и сбора данных" с характерной иконкой (рис.3.1).

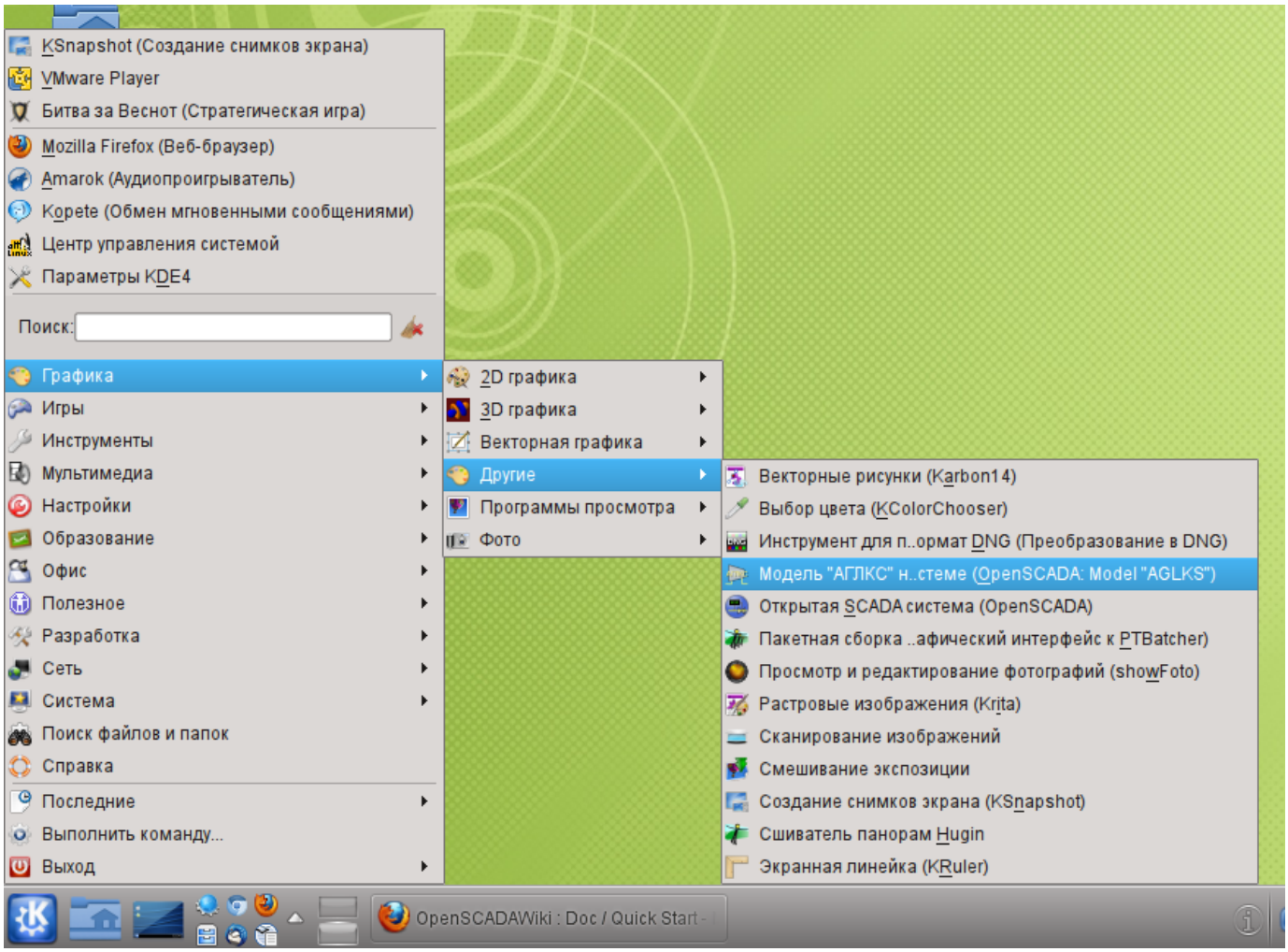

Рис. 3.1. Пункт меню окружения рабочего стола для запуска демонстрации OpenSCADA.

Запуск также можно осуществить из консоли командой:

```
$ openscada_demo
```

⚠ При запуске OpenSCADA из консоли, с помощью команды "**$ openscada**", осуществляется запуск без конфигурации, в результате которого запрашивается пользователь и пароль для входа. По умолчанию в системе OpenSCADA предусмотрен привилегированный пользователь "root" (пароль "openscada") и непривилегированный "user" (пароль "user"), которые не имеют никакого отношения к пользователям операционной системы. Запуск OpenSCADA таким образом имеет смысл только от суперпользователя ОС ("root") или в режиме демона.

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

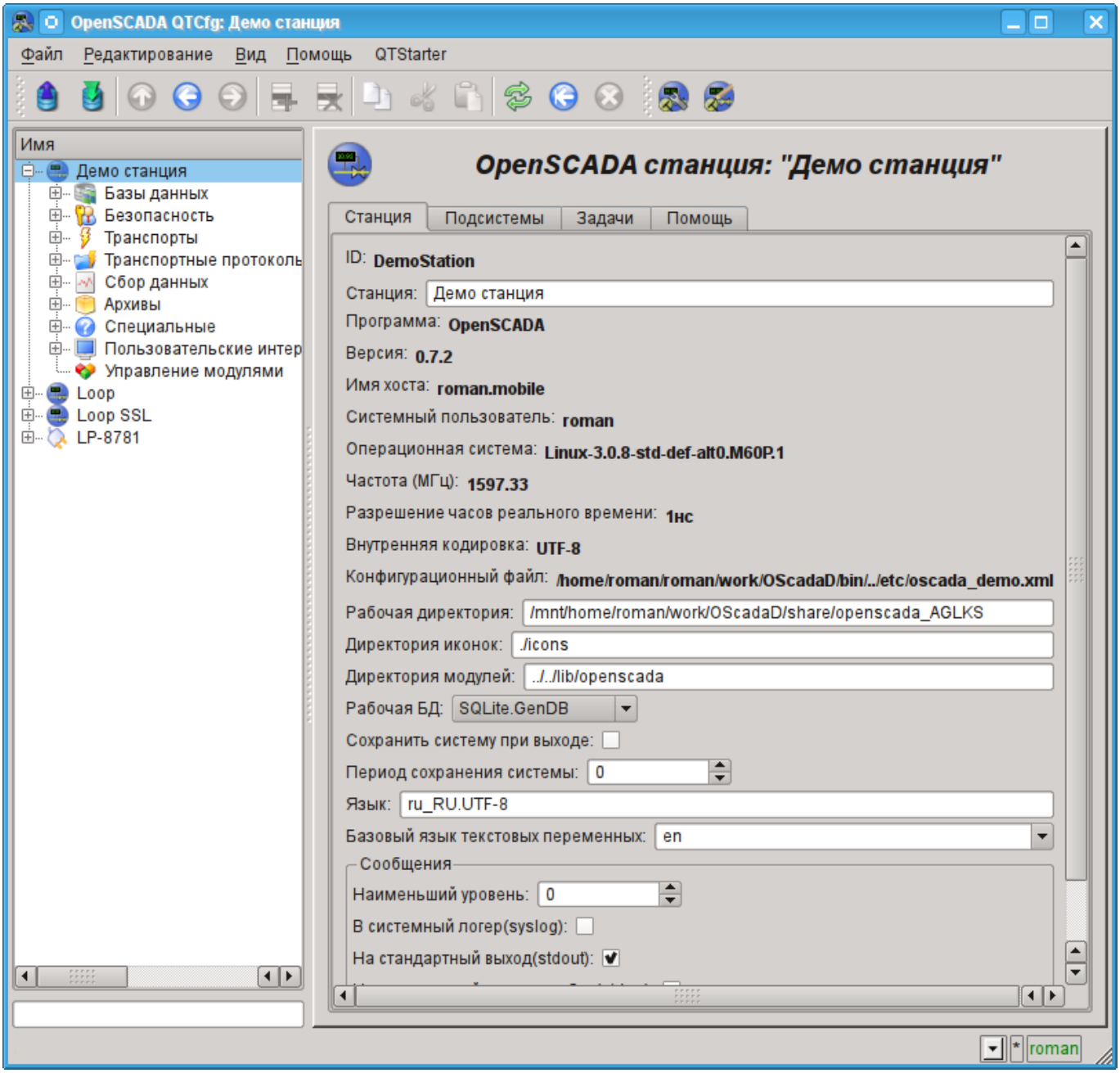

Рис. 3.2. OpenSCADA конфигуратор - QTCfg, корневая страница.

Конфигуратор OpenSCADA является основным и достаточным средством для конфигурации любого компонента системы. Как и многие компоненты OpenSCADA, конфигуратор реализован в виде модуля. Кроме конфигуратора QTCfg могут быть доступны другие конфигураторы,

выполняющие те же функции, но реализованные на основе других технологиях. Например, таковыми являются Web-конфигураторы: WebCfg и WebCfgD.

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

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

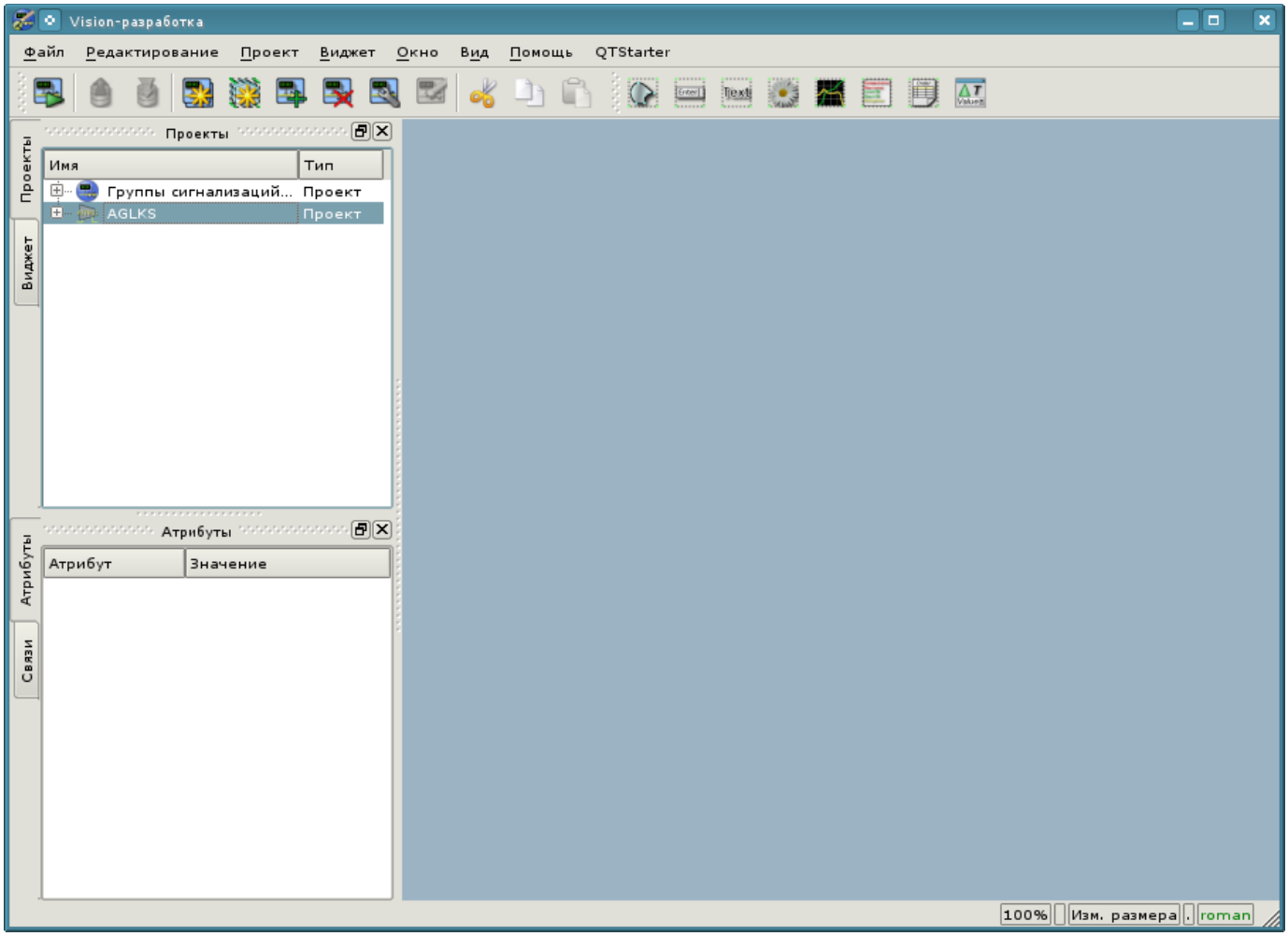

Рис. 3.3. Окно разработки пользовательского интерфейса.

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

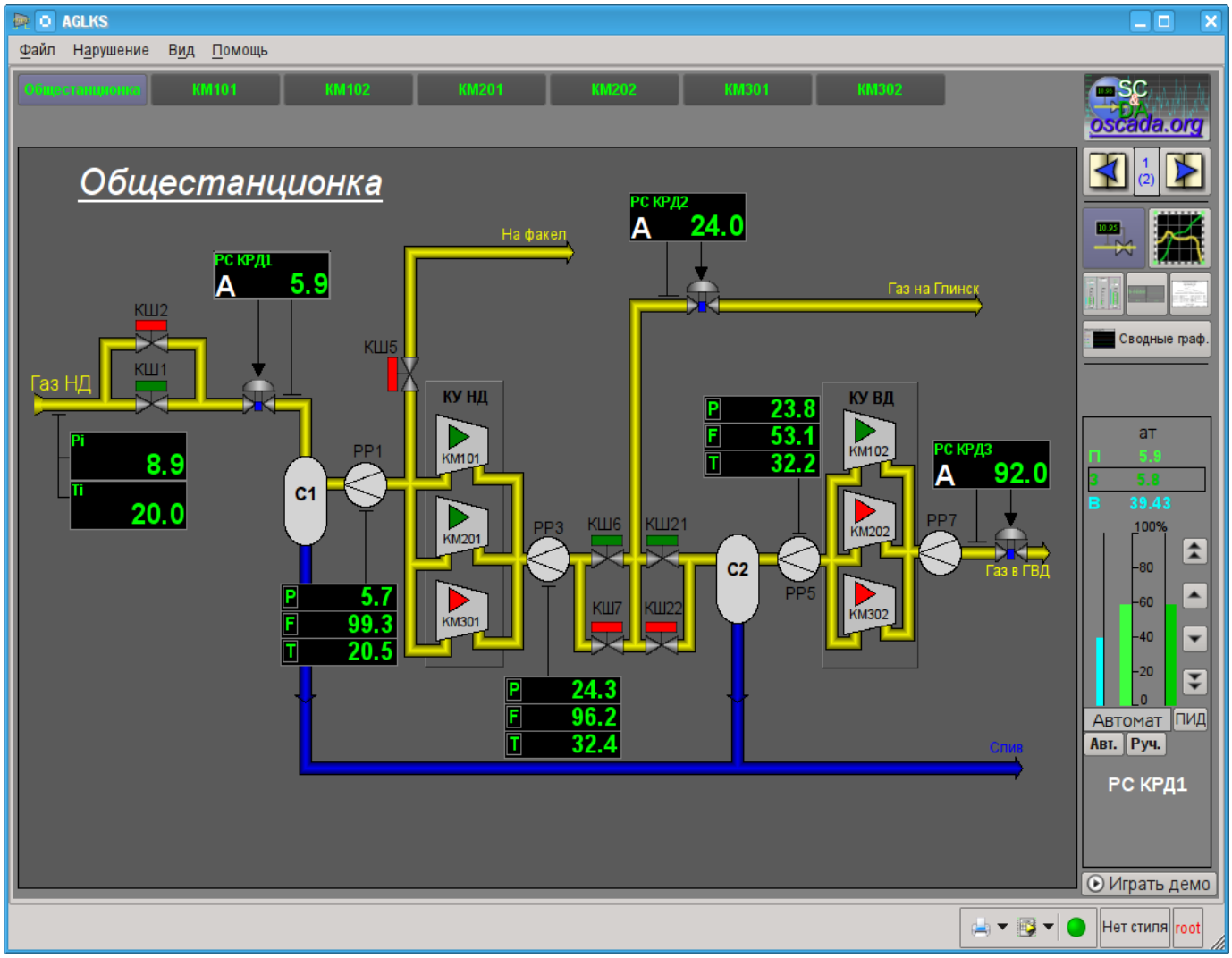

Рис. 3.4. Окно пользовательского интерфейса проекта "AGLKS".

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

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

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

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

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

3.1. Создание пользовательского проекта с чистого листа

Все действия в последующих разделах описаны в окружении БД модели "АГЛКС" (демонстрация) с целью как можно более полного и наглядного представления процесса конфигурации, с возможностью подключения к реальному-живому источнику данных, реализованному на основе модели технологического процесса (ТП) газо-компрессорной станции. Однако нужно несколько отвлечься и описать процесс создания пользовательского проекта с чистого листа, что очевидно является Вашей конечной целью. На основе нового пользовательского проекта можно выполнять все описанные ниже действия с БД модели "АГЛКС", но с оглядкой на источники данных собственного, нового, проекта.

Для запуска чистого пользовательского проекта в меню окружения рабочего стола, разделе "Графика", предусмотрен пункт "Открытая SCADA система" с характерной иконкой (рис.3.1.1).

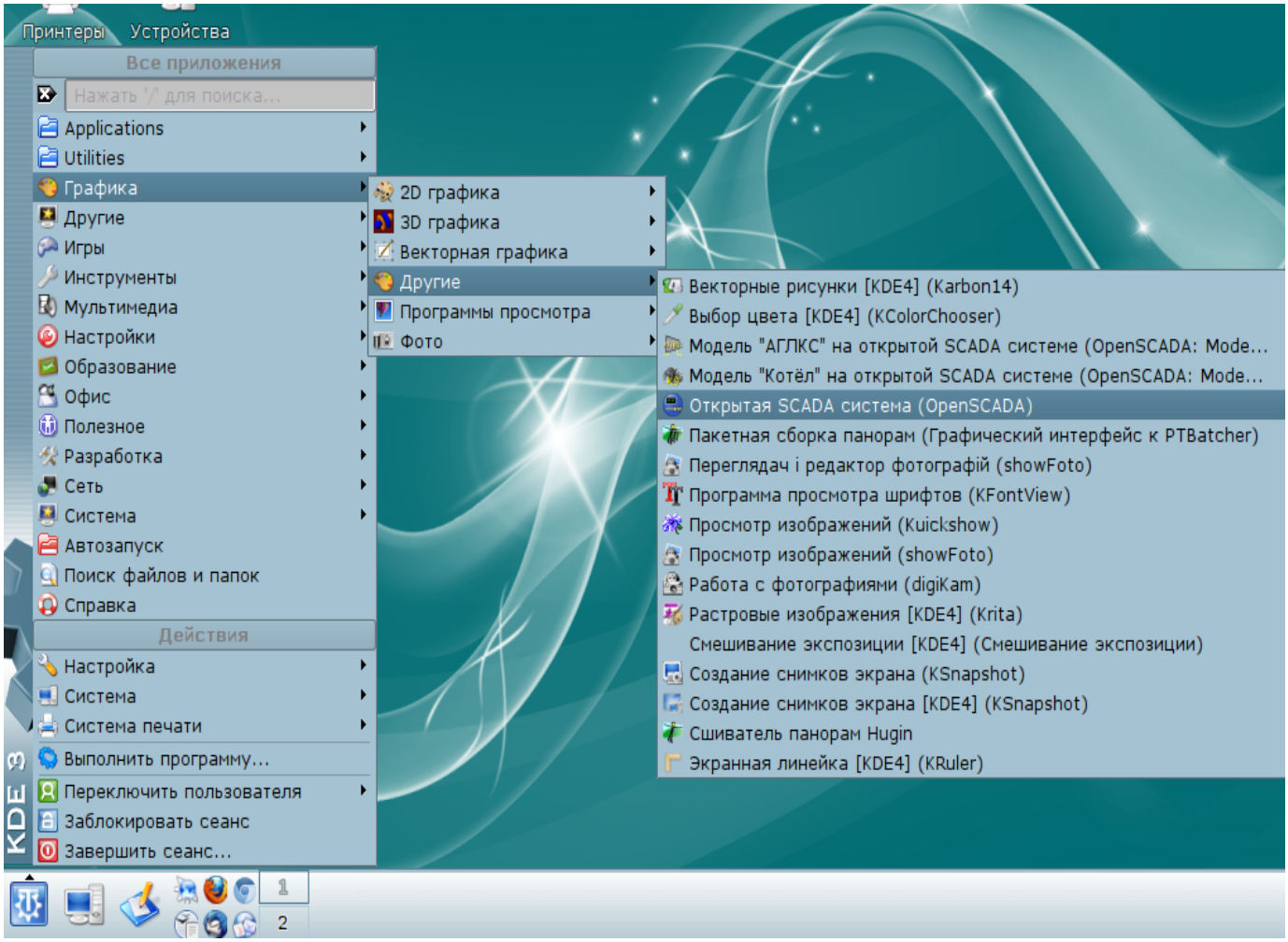

Рис. 3.1.1. Пункт меню окружения рабочего стола для запуска чистого пользовательского проекта.

Запуск также можно осуществить из консоли командой:

```
$ openscada_start
```

Чистый пользовательский проект не содержит никакой специфической для конкретного проекта конфигурации и настроен на работу в директории пользователя "**/.openscada**", с основной SQLite БД в файле "**DATA/MainSt.db**". Создавать сложный проект SCADA-системы проще с использованием библиотек функций API объектной модели системы OpenSCADA, библиотек графических элементов, а так-же других библиотек OpenSCADA. Для использования библиотек OpenSCADA, хранящихся в файле БД, их нужно подключить, добавить в объекте модуля БД "SQLite" (рис.3.1.2), установить адрес и кодировку БД в "UTF-8" (рис.3.1.3).

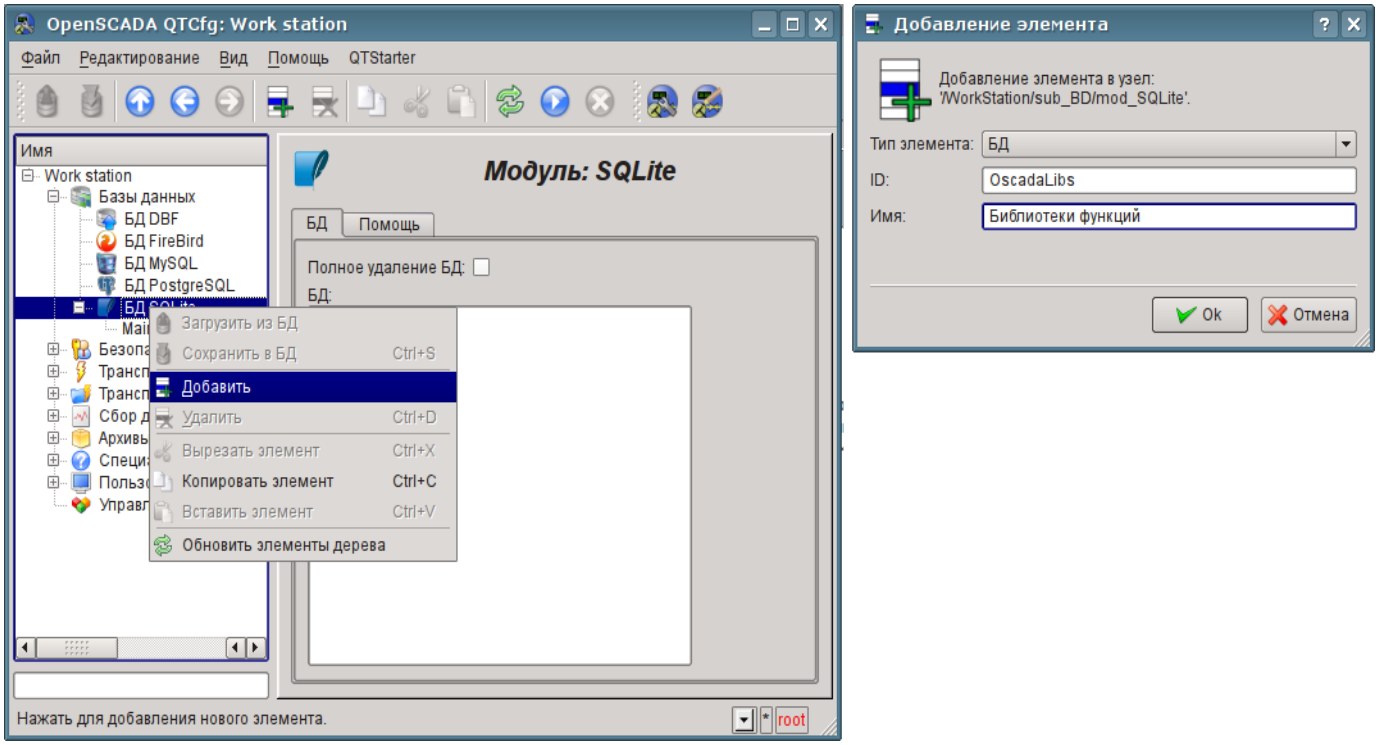

Рис. 3.1.2. Добавление объекта БД "SQLite".

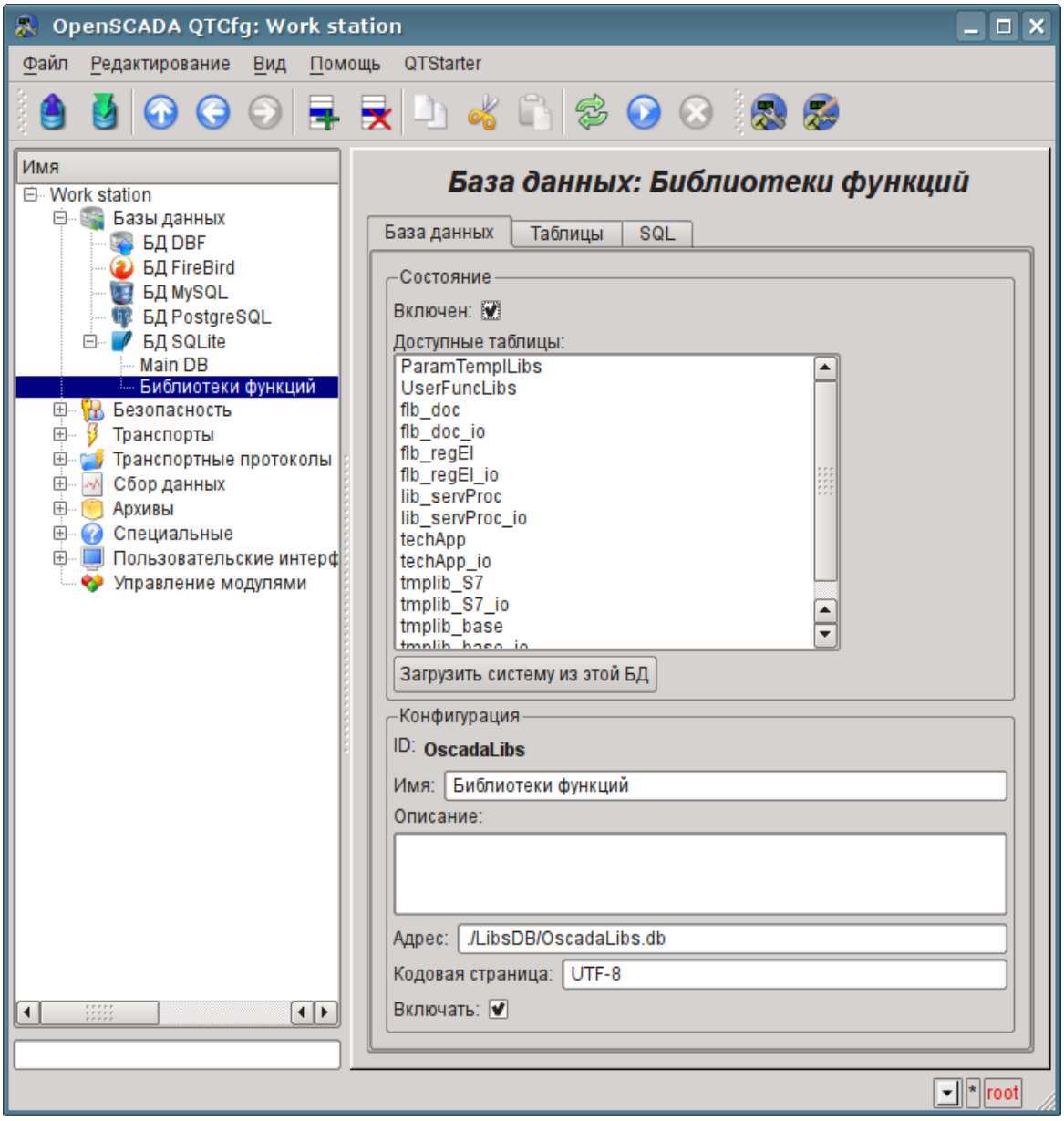

Рис. 3.1.3. Объект БД "SQLite" библиотеки OpenSCADA.

Дистрибутивы OpenSCADA поставляются с рядом библиотек в виде файлов БД "SQLite" (таблица 3.1), которые, при запуске чистого пользовательского проекта, помещаются в директорию "**LibsDB/**". Согласно этому списку добавляем их в объекте модуля БД "SQLite", устанавливаем им флаг "Включать" и сохраняем. Далее, для загрузки содержимого библиотек можно включить БД и нажать кнопку "Загрузить систему из этой БД", однако при загрузке ряд новых объектов не включатся поэтому проще завершить чистый пользовательский проект и запустить по новой.

Таблица 3.1. Библиотеки OpenSCADA в составе дистрибутива.

ID	Имя	Адрес	Языки/кодировка
OscadaLibs	Библиотеки функций	./LibsDB/OscadaLibs.db	EN,RU,UK/UTF-8
vcaBase	СВУ: Главные библиотеки	./LibsDB/vcaBase.db	EN,RU,UK/UTF-8
vcaTest	СВУ: Тесты	./LibsDB/vcaTest.db	EN,RU,UK/UTF-8
vcaElectroEls	СВУ: Библиотека электроэлементов мнемосхем пользовательского интерфейса	./LibsDB/vcaElectroEls.db	EN,RU,UK/UTF-8

В результате добавления библиотек OpenSCADA Вы получите окружение, готовое для добавления источников данных и формирования интерфейса своего, нового проекта SCADA-системы.

4. Работа с источниками данных

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

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

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

4.1. Опрос данных аппарата ТП

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

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

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

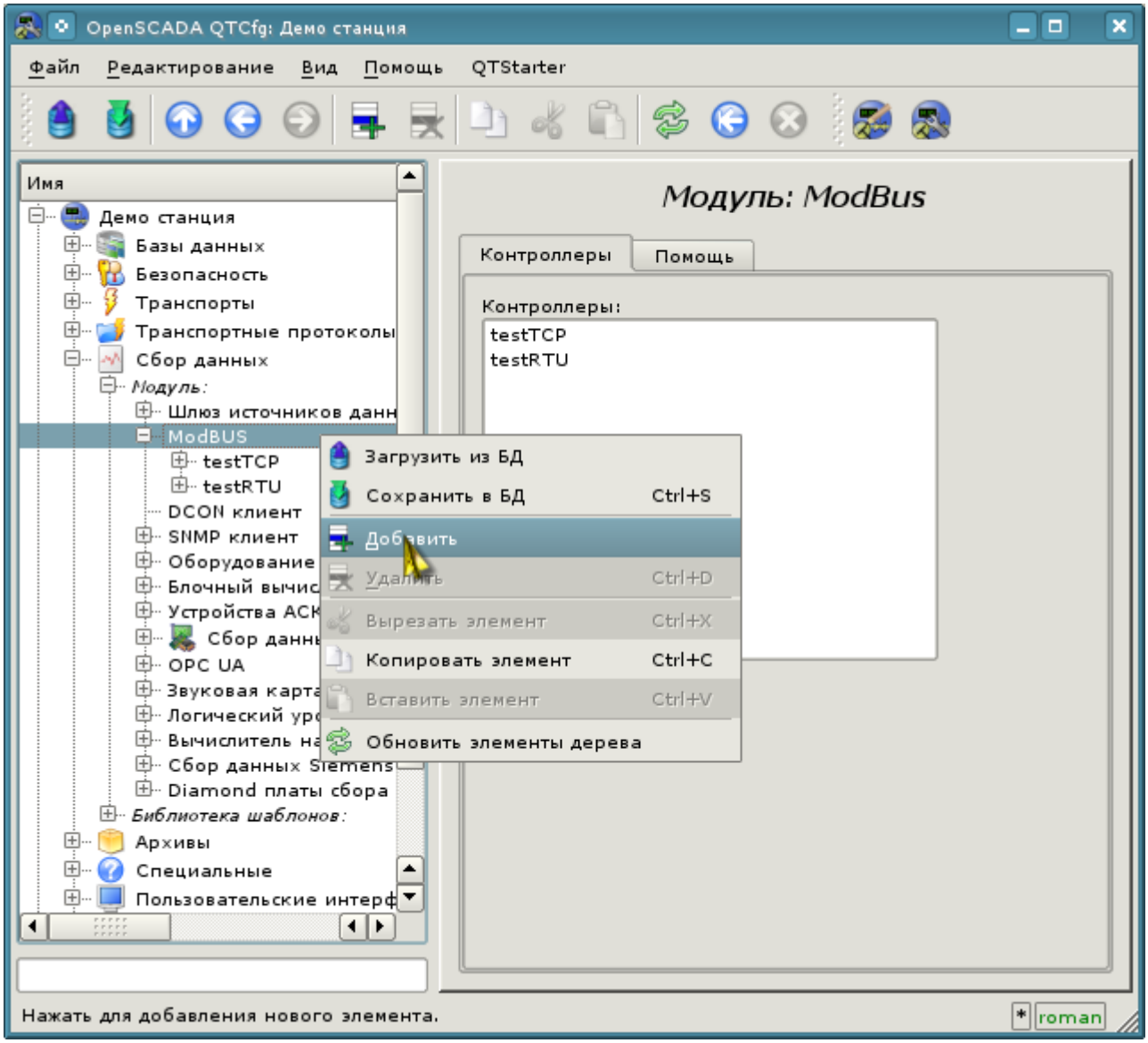

Рис. 4.1.1. Добавление контроллера в модуле "ModBUS" подсистемы "Сбор данных".

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

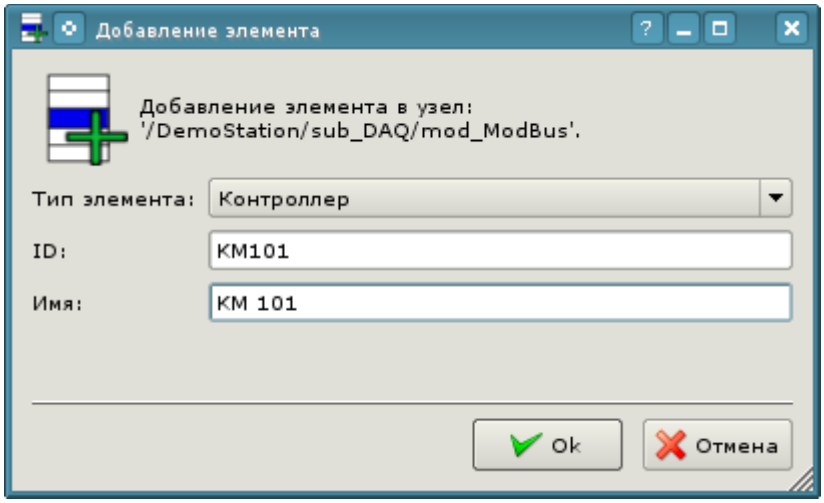

Рис. 4.1.2. Диалог для указания идентификатора и имени нового объекта.

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

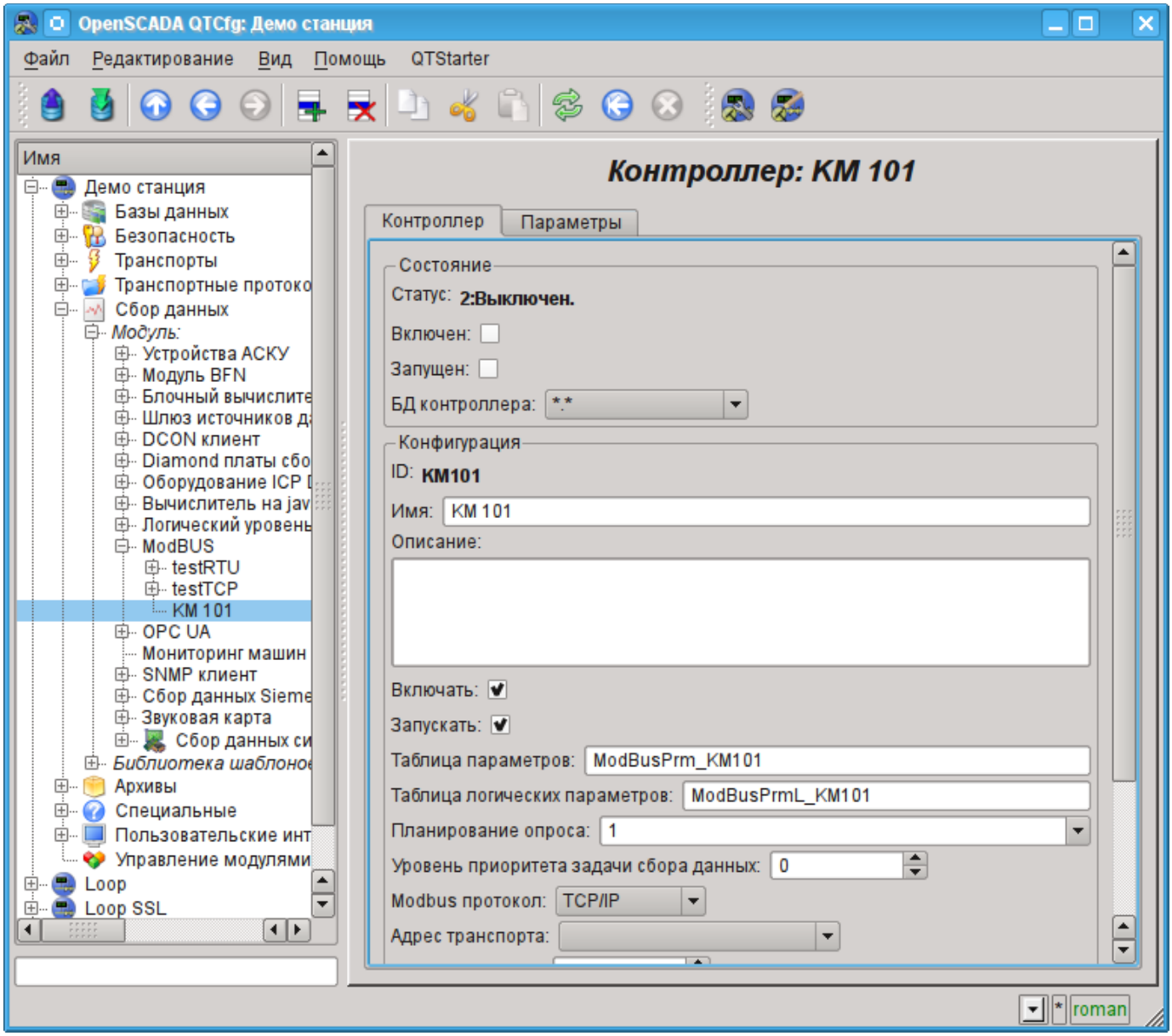

Рис. 4.1.3. Главная вкладка настройки объекта контроллера модуля ModBUS.

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

Перед конфигурацией связи со своим контроллером предварительно необходимо, из документации на контроллер, выяснить настройки его сетевых интерфейсов и протоколов, а также, в случае использования "ModBus", 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 мы можем поменять прямо здесь, а вот идентификатор невозможно изменять прямо, однако объект можно вырезать (Ctrl+X) и затем вставить (Ctrl+V), тем самым переименовав его.
- "Описание" может содержать развёрнутую характеристику и назначение объекта контроллера. В нашем случае значение этого поля непринципиально.
- "Включать" и "Запускать" указывают на то, в какое состояние переводить объект контроллера при запуске OpenSCADA. Установим оба поля.
- "Таблица параметров" — содержит имя таблицы БД, в которой будет храниться конфигурация параметров данного контроллера. Оставим по умолчанию.
- "Планирование опроса" — содержит конфигурацию планировщика для запуска задачи опроса. Получить описание формата конфигурации данного поля можно из всплывающей подсказки. Одиночная цифра указывает на периодичность запуска в секундах. Укажем одну секунду.
- "Уровень приоритета задачи сбора данных" — указывает насколько приоритетная данная задача (от -1 до 99). Приоритеты выше нуля имеют смысл только при запуске OpenSCADA от привилегированного пользователя. Оставим это поле без изменений.
- "ModBUS протокол" — указывает на вариант протокола ModBUS. Возможны варианты протокола "TCP/IP", "RTU" и "ASCII". На данный момент нас интересует вариант "TCP/IP", поэтому оставим как есть. Варианты протоколов "RTU" и "ASCII" нужно устанавливать в случае связи с контроллером посредством последовательных интерфейсов, часто "RS-485".
- "Адрес транспорта" — указывает на исходящий транспорт подсистемы "Транспорты", который используется для соединения с контроллером. В случае с вариантом "TCP/IP" нам нужен транспорт в модуле Sockets, а в случае вариантов "RTU", "ASCII" и последовательного интерфейса нам нужен транспорт в модуле Serial. На создании исходящего транспорта в "Sockets" и "Serial" детальнее остановимся ниже.
- "Узел назначения" — указывает узел источника данных или контроллера в сети ModBUS. В нашем случае это должно быть "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 (0x0F,0x10)" — вместо функций одно-элементной записи будут использованы много-элементные. 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.
- "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, нажав вторую слева иконку на панели инструментов.

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

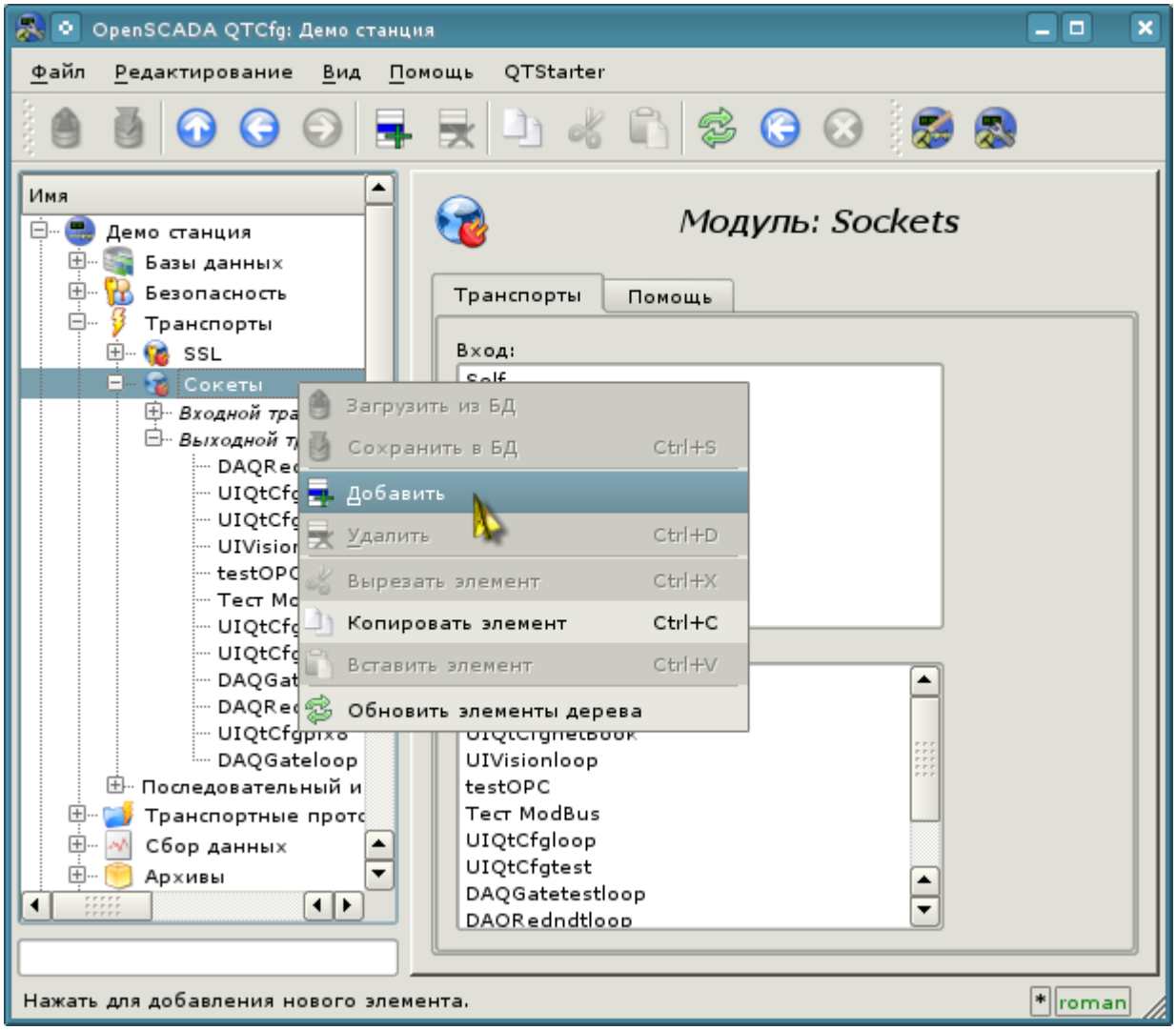

Рис. 4.1.4. Добавление исходящего транспорта в модуле "Sockets" подсистемы "Транспорты".

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

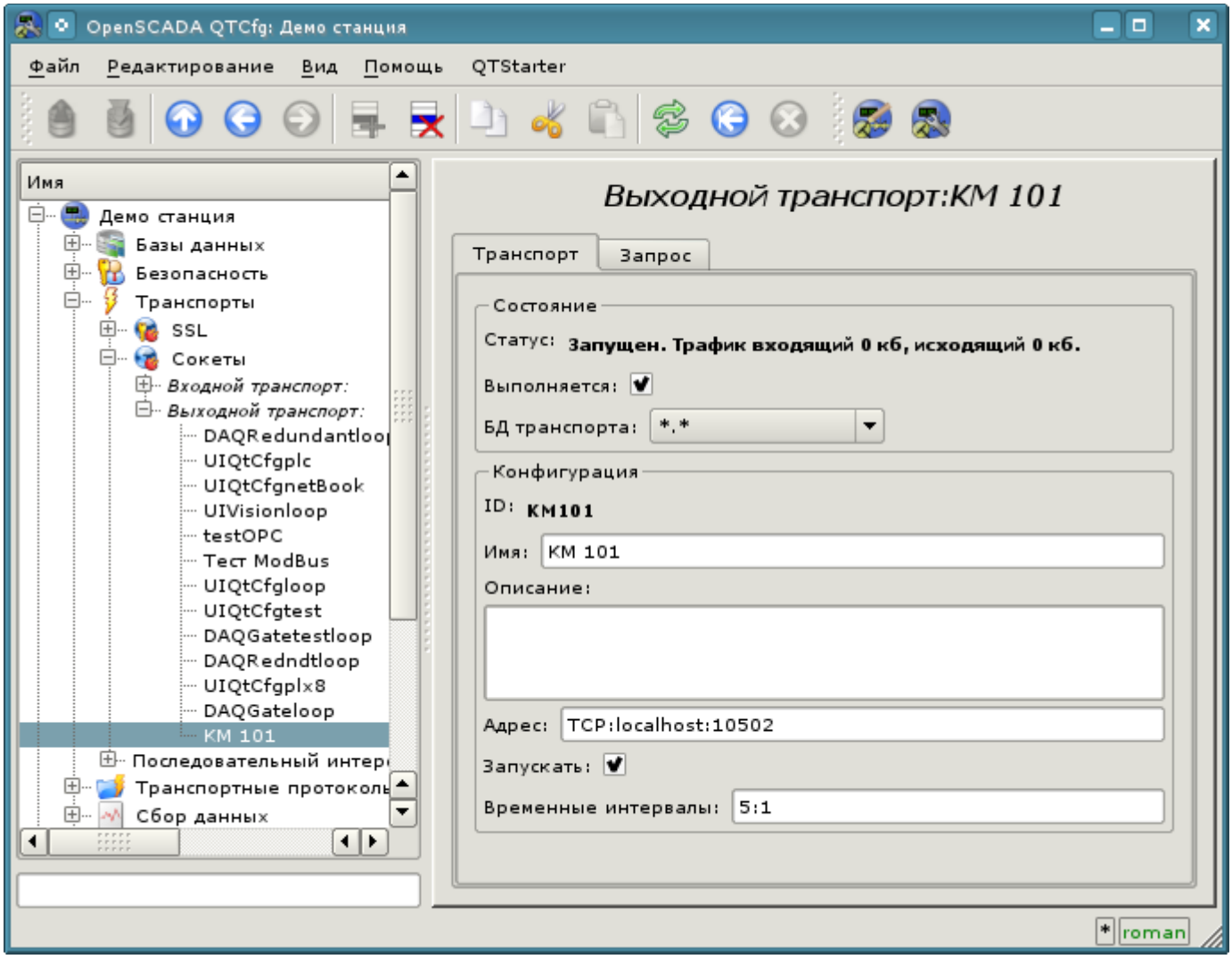

Рис. 4.1.5. Страница конфигурации исходящего транспорта модуля "Sockets" подсистемы "Транспорты".

В разделе "Конфигурация" непосредственно содержится конфигурация объекта транспорта:

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

Транспорты других типов создаются подобным к рассмотренному для "Sockets" способу, а конфигурация их отличается обычно только форматом записи адреса и таймаутов. В случае с транспорта модуля "Serial" в адресе указывается путь к последовательному устройству, скорость, и формат. Для переходников *USB->Serial* этот 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.KM101". На этом настройка объекта контроллера закончена, включим его установив флаг "Включен". Следующим этапом является конфигурация и выбор тех данных, которые нужно опрашивать из контроллера. Эта настройка производится путём создания объекта "Параметр" контроллера. Объект "Параметр" позволяет описать перечень данных, получаемых у контролера и передать их в окружение OpenSCADA.

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

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

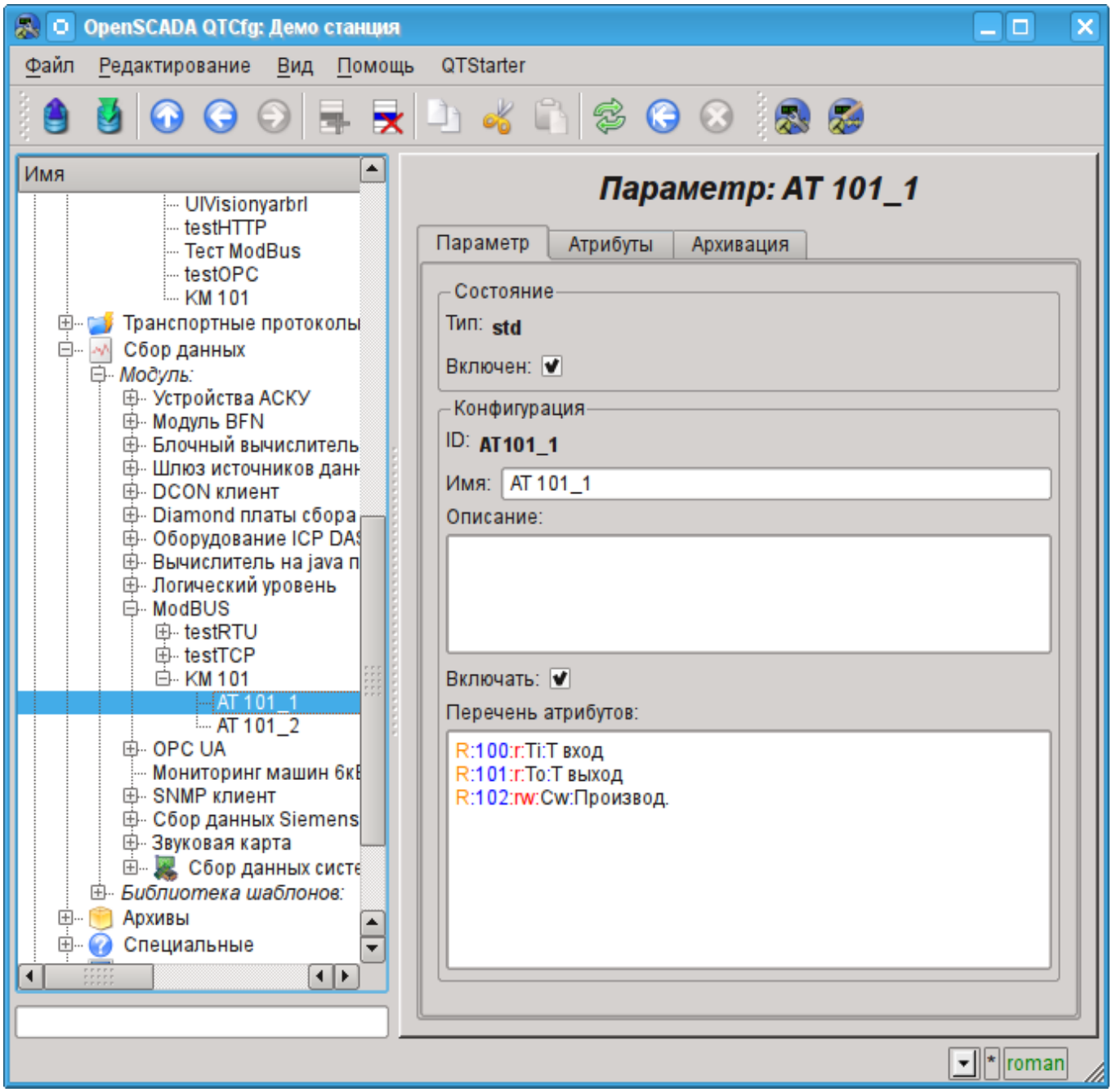

Рис. 4.1.6. Страница конфигурации параметра контроллера "ModBUS".

В разделе "Конфигурация" непосредственно содержится конфигурация объекта параметра:

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

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

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

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

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

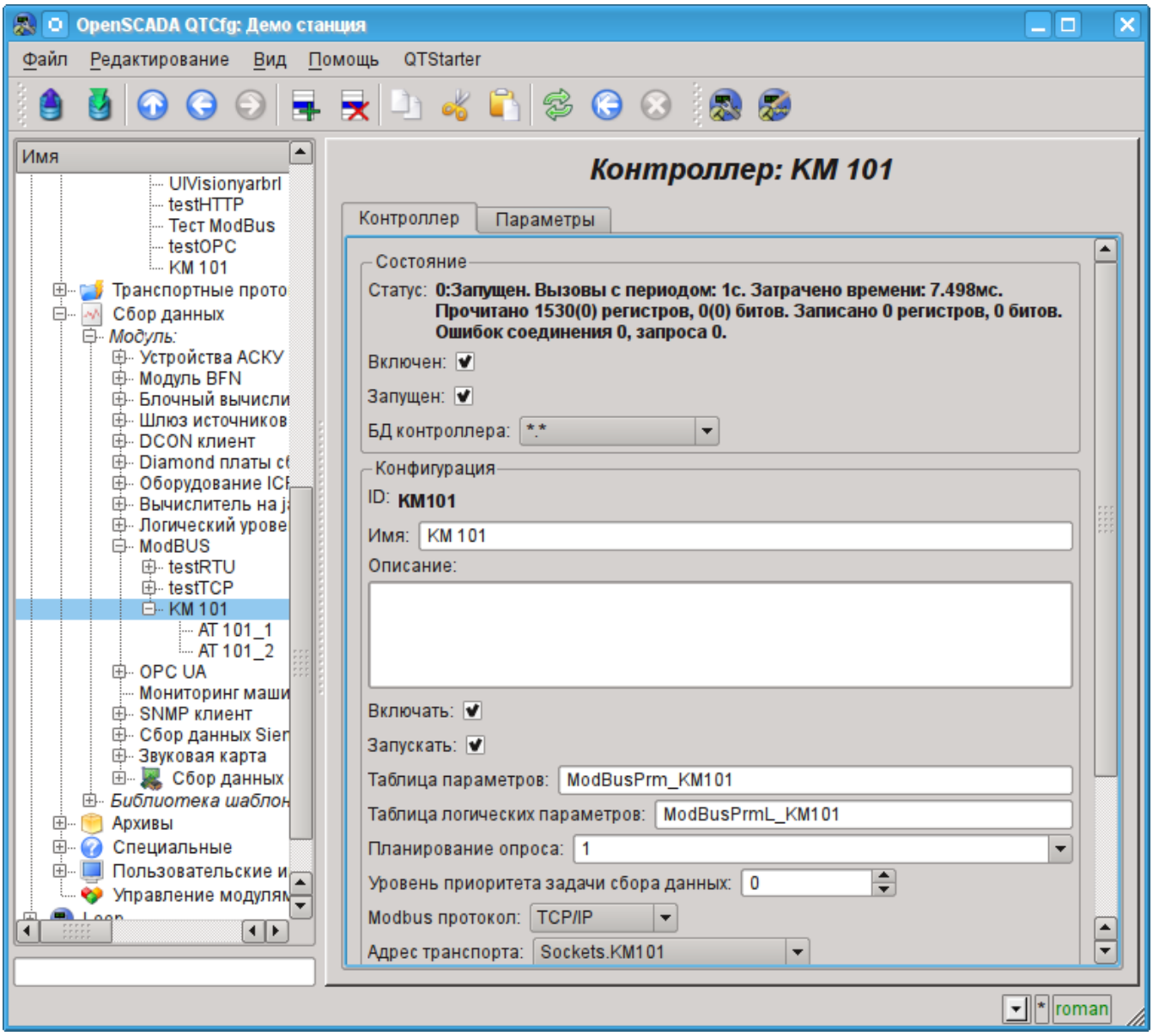

Рис. 4.1.7. Страница объекта контроллера при успешном обмене с физическим контроллером.

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

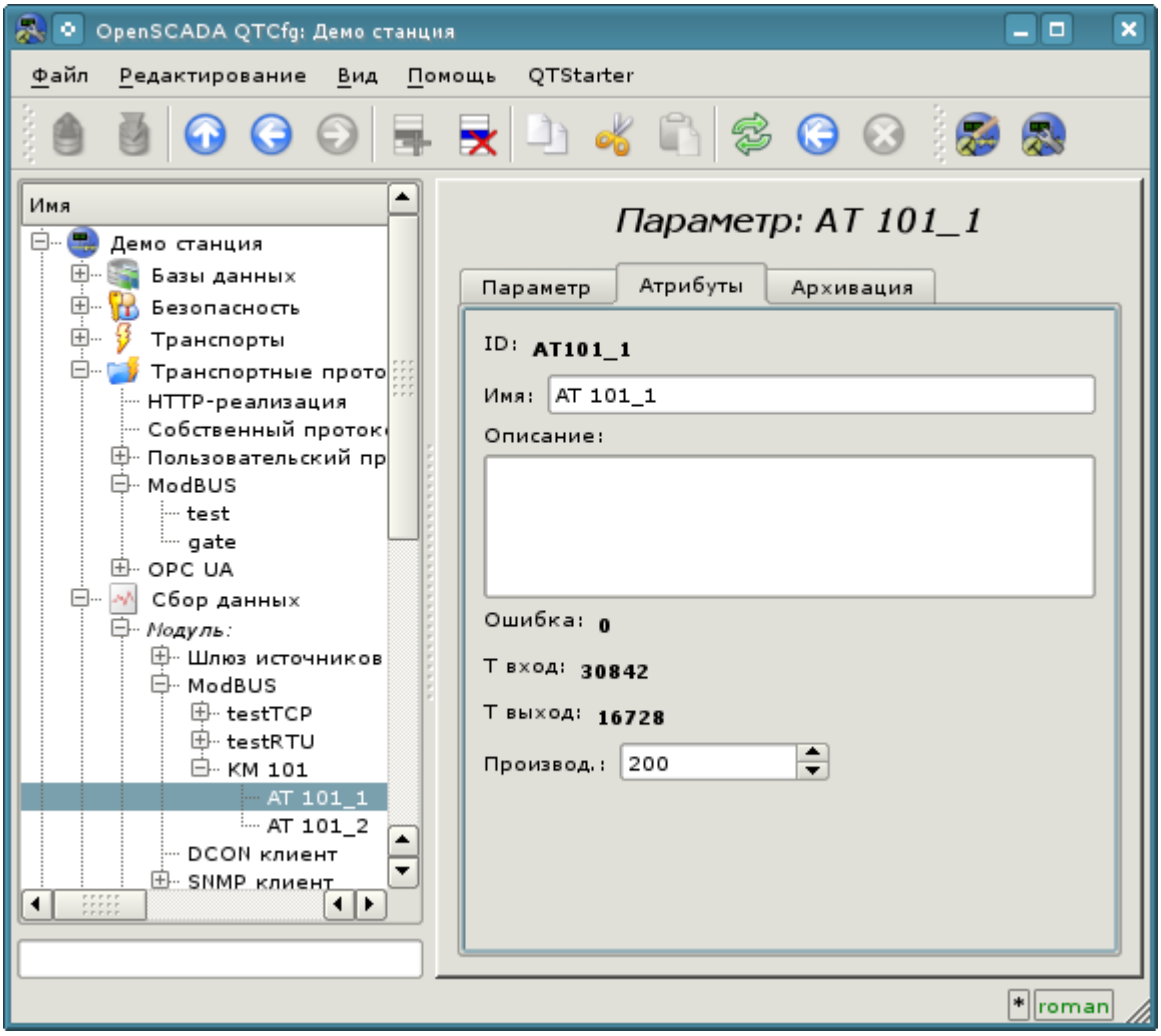

Рис. 4.1.8. Страница описанных атрибутов параметра AT101_1.

На этом конфигурация сбора данных считается законченной.

4.2. Обработка полученных данных ТП

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

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

Для выполнения вычислений в модуле логического уровня нужно предварительно создать шаблон параметра подсистемы "Сбор данных". Для этого откроем страницу библиотеки шаблонов

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

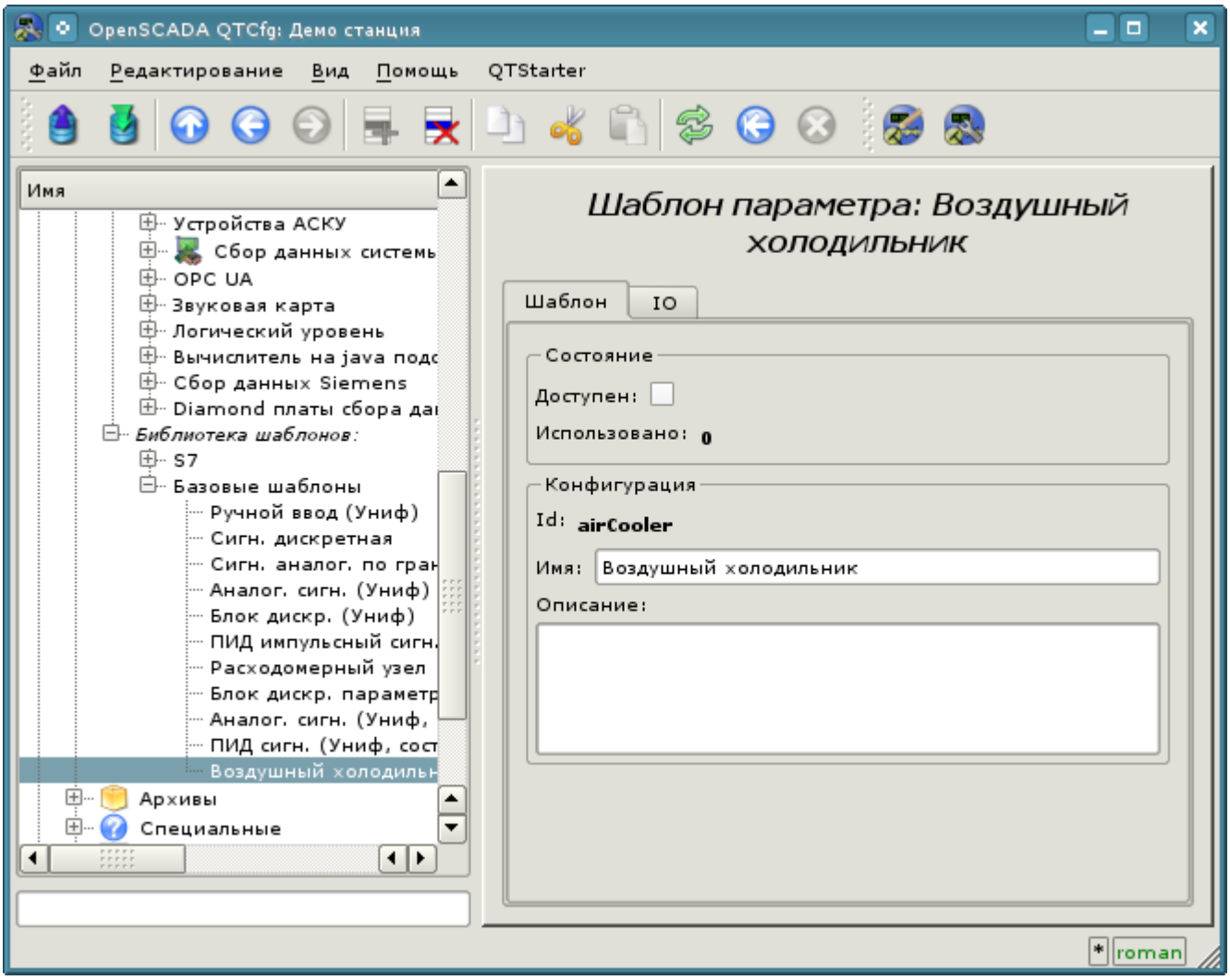

Рис. 4.2.1. Страница конфигурации объекта шаблона.

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

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

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

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

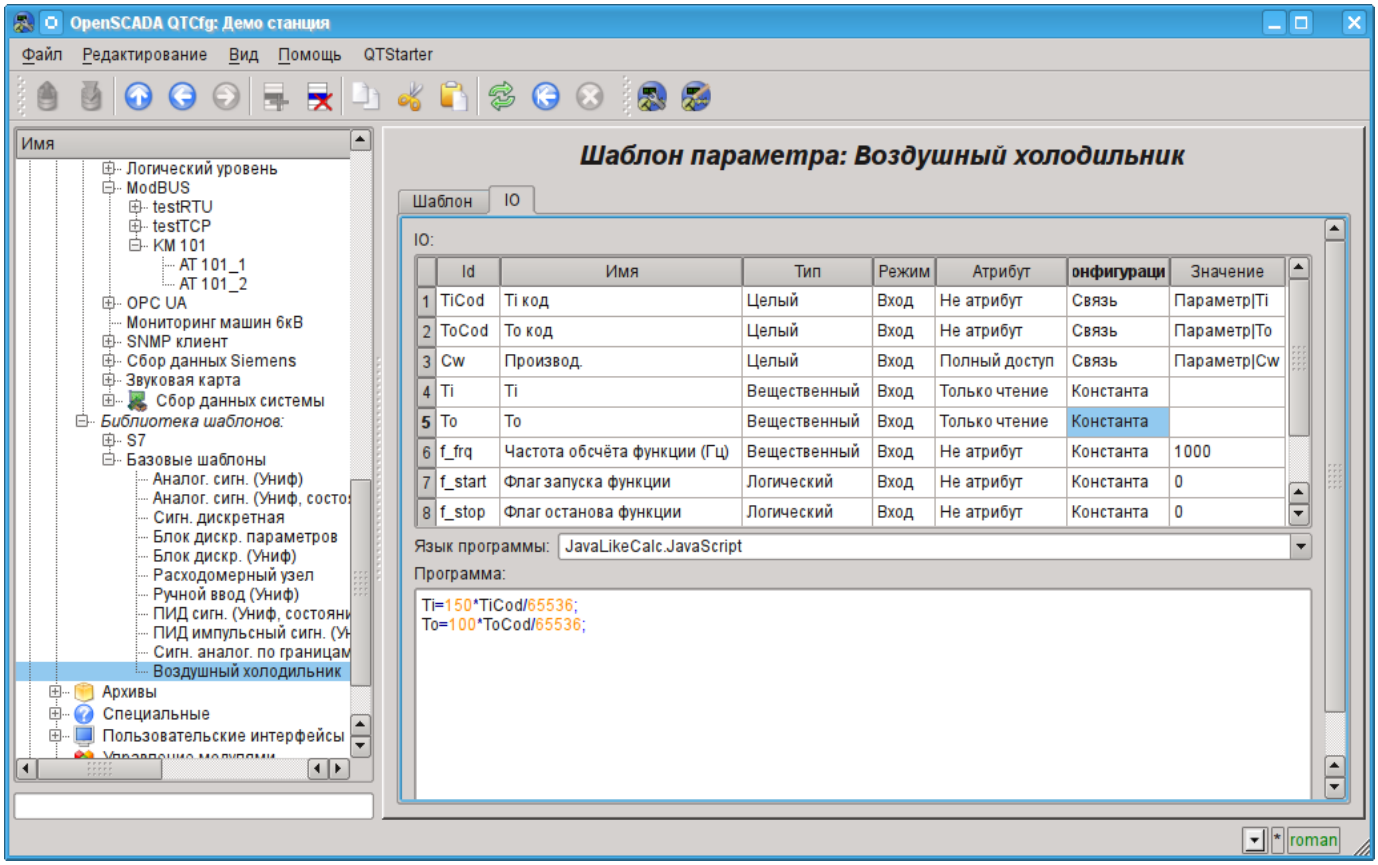

Рис. 4.2.2. Вкладка "IO" страницы конфигурации объекта шаблона.

Полученный шаблон сохраним и установим флаг доступности.

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

Объект контроллера модуля "LogicLev" (рис.4.2.3) не имеет специфических настроек и установленные по умолчанию можно не трогать, кроме поля "Планирование вычислений", которое установим в одну секунду. Перед добавлением параметров нужно включить контроллер, установив флаг "Включен".

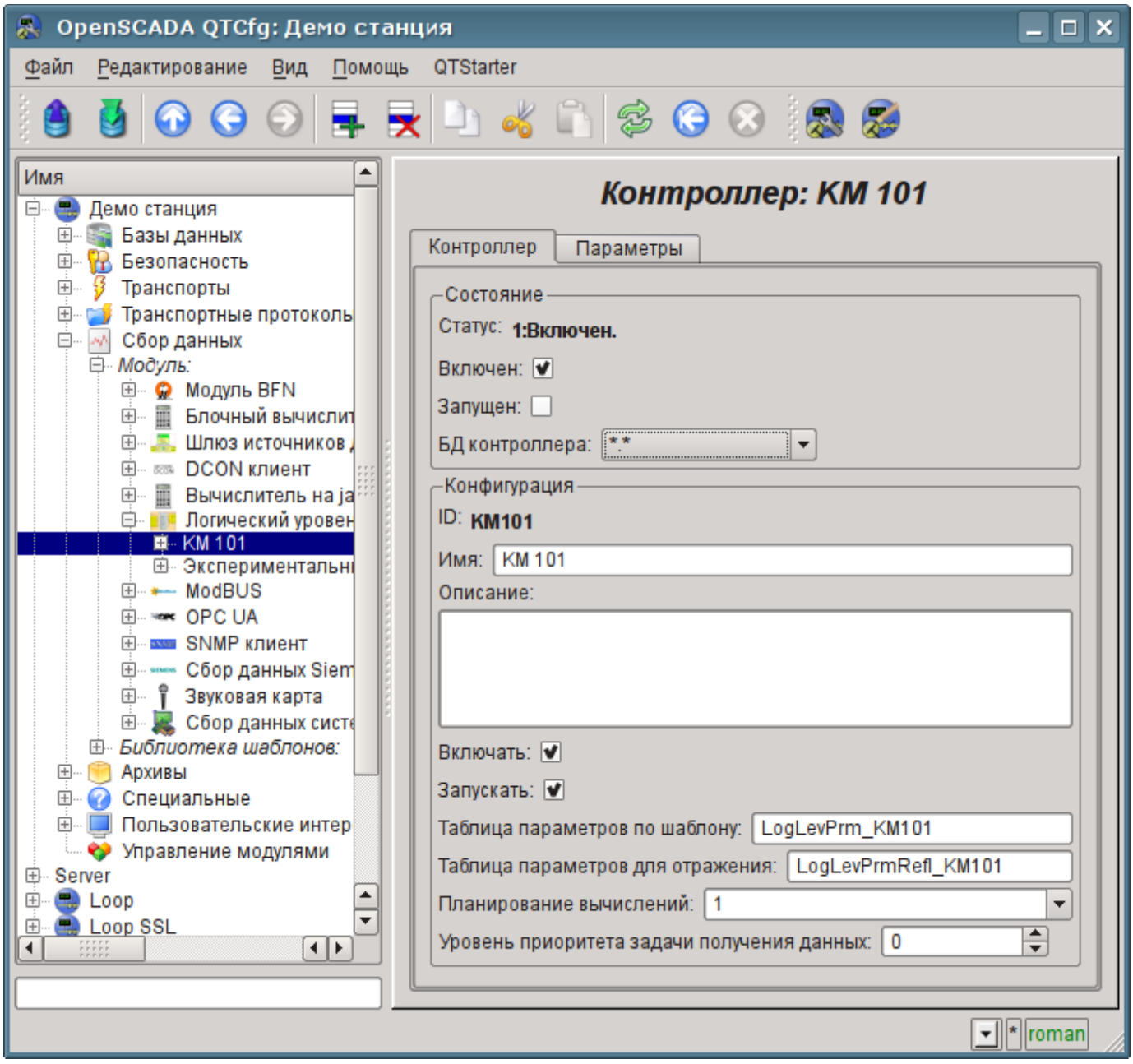

Рис. 4.2.3. Главная вкладка настройки объекта контроллера модуля LogicLev.

Объект параметра контроллера модуля "LogicLev" (рис.4.2.4) из специфических настроек имеет "Тип", где нужно установить "Логический" (std), а в поле "Шаблон параметра" выбрать адрес созданного нами шаблона.

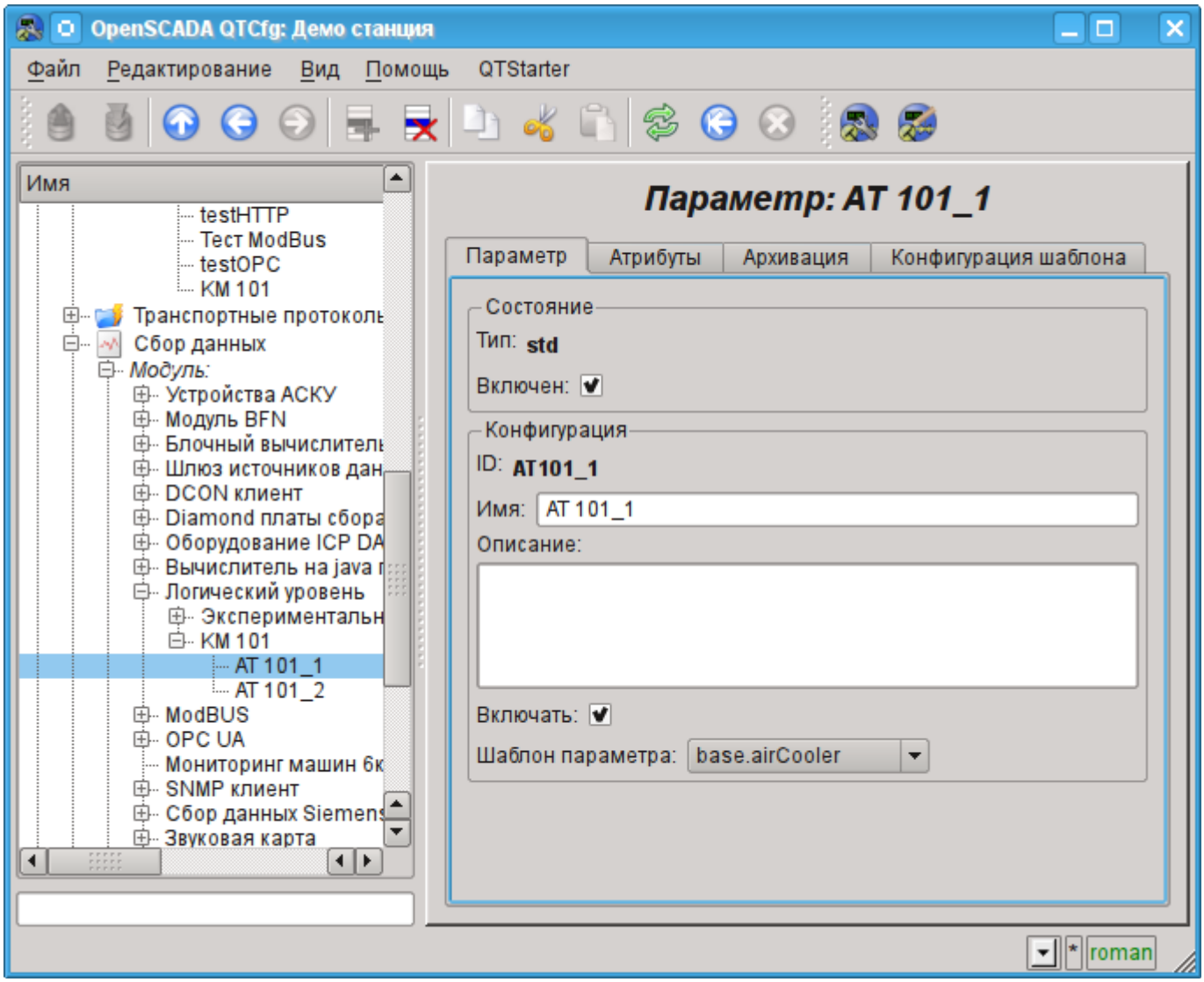

Рис. 4.2.4. Страница конфигурации параметра контроллера "LogicLev".

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

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

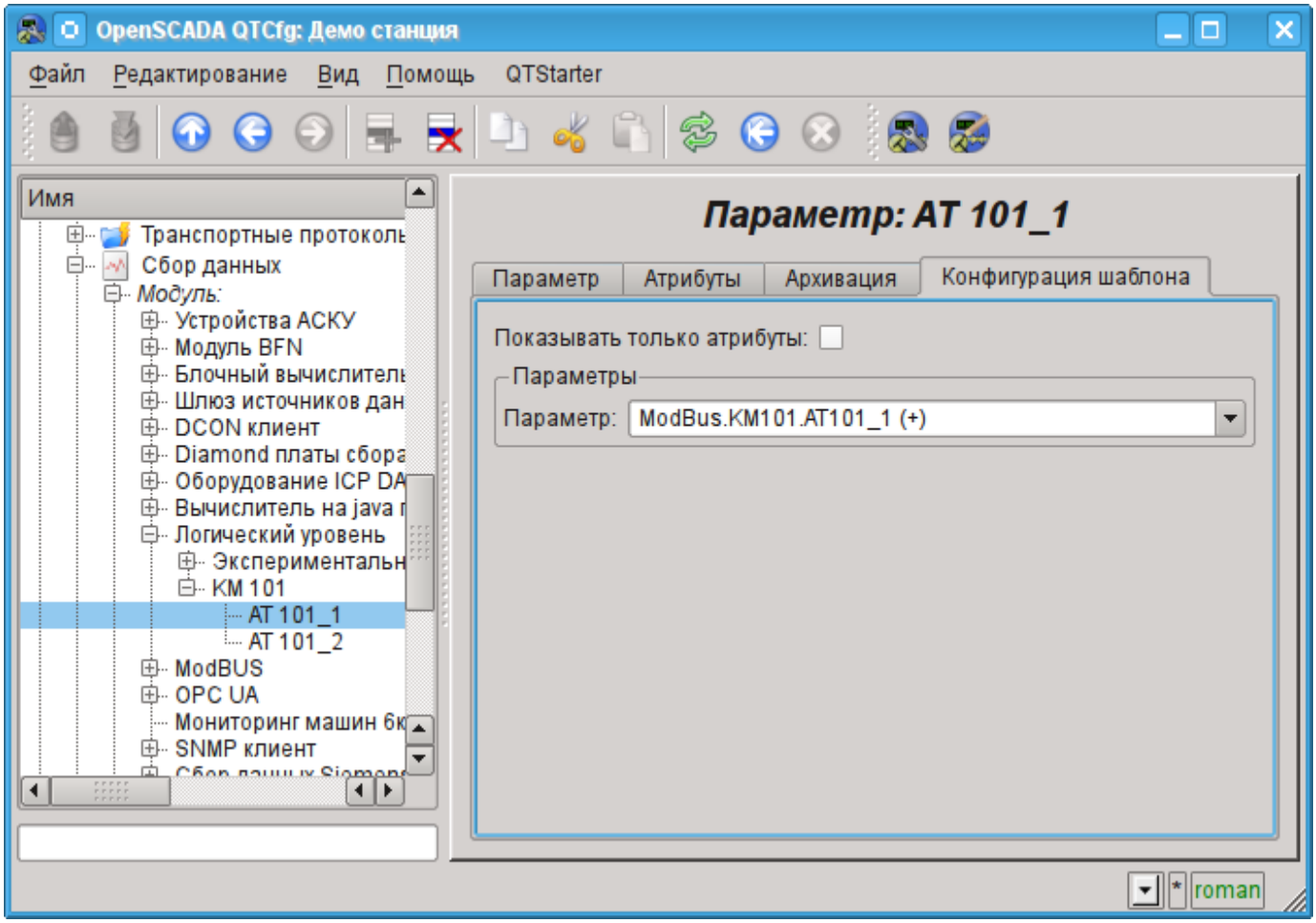

Рис. 4.2.5. Вкладка "Конфигурация шаблона" страницы параметра контроллера "LogicLev".

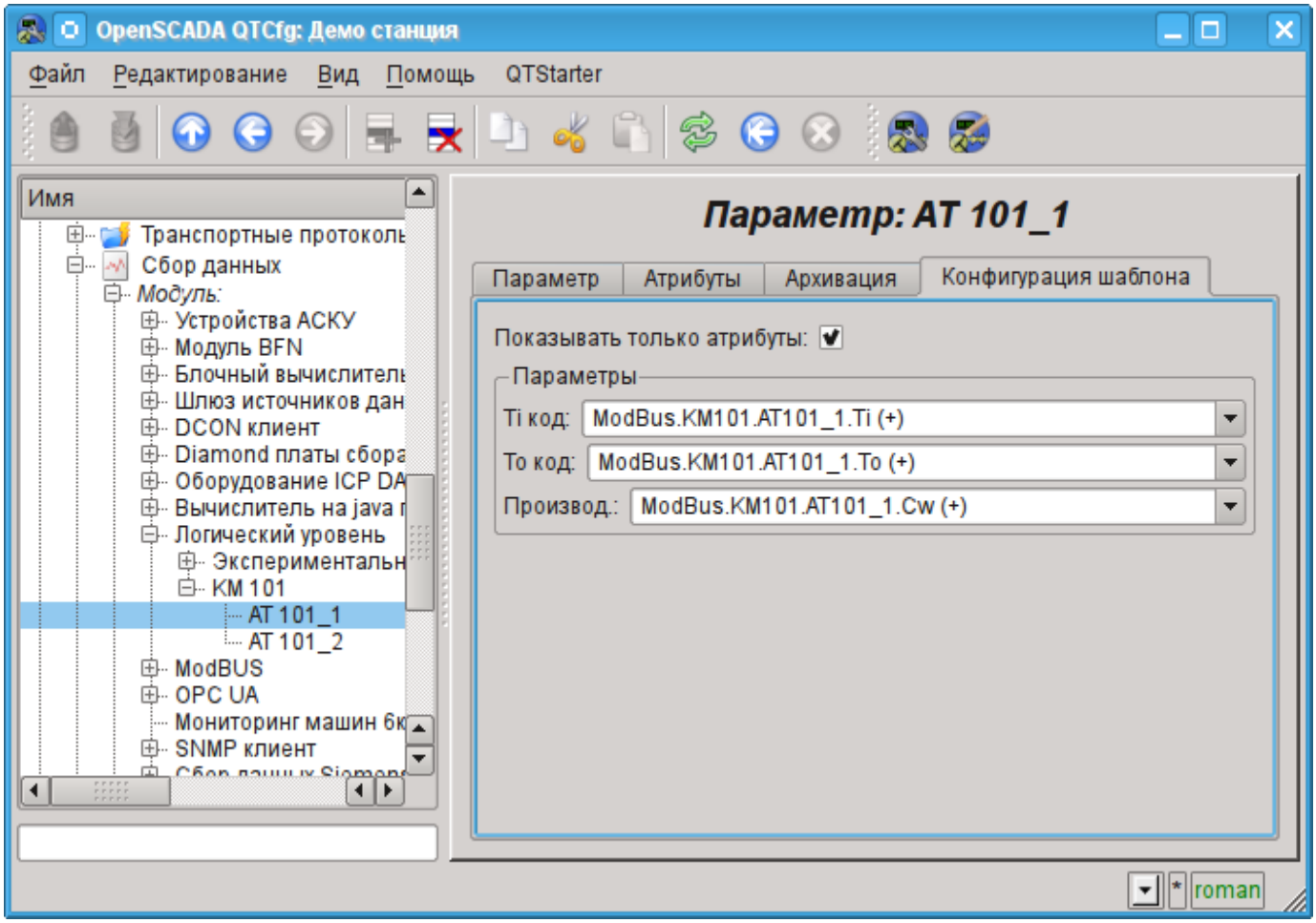

Рис. 4.2.6. Вкладка "Конфигурация шаблона" страницы параметра контроллера "LogicLev" с разворотом связей.

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

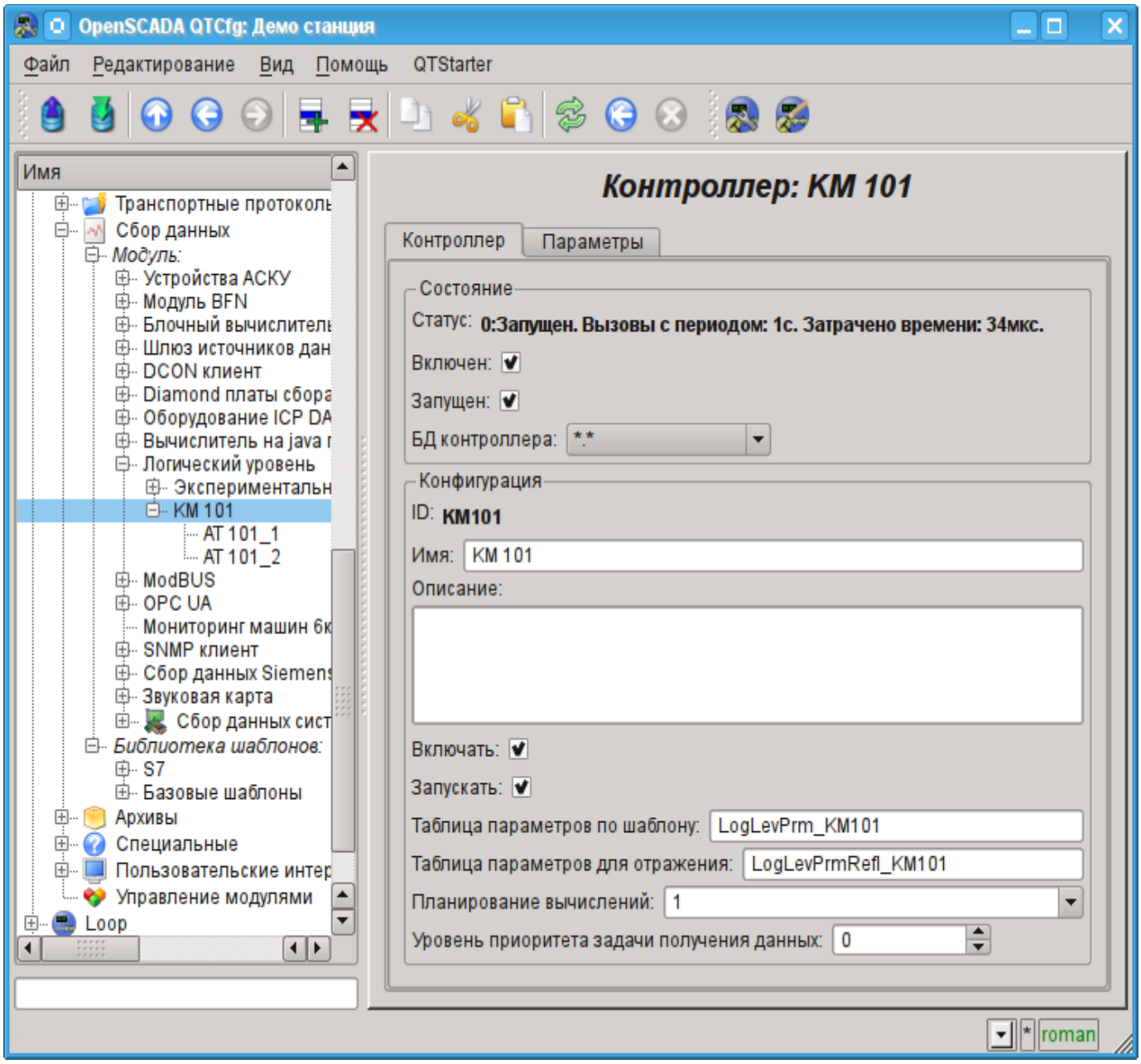

Рис. 4.2.7. Страница объекта контроллера при успешном вычислении контроллера в модуле "LogicLev".

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

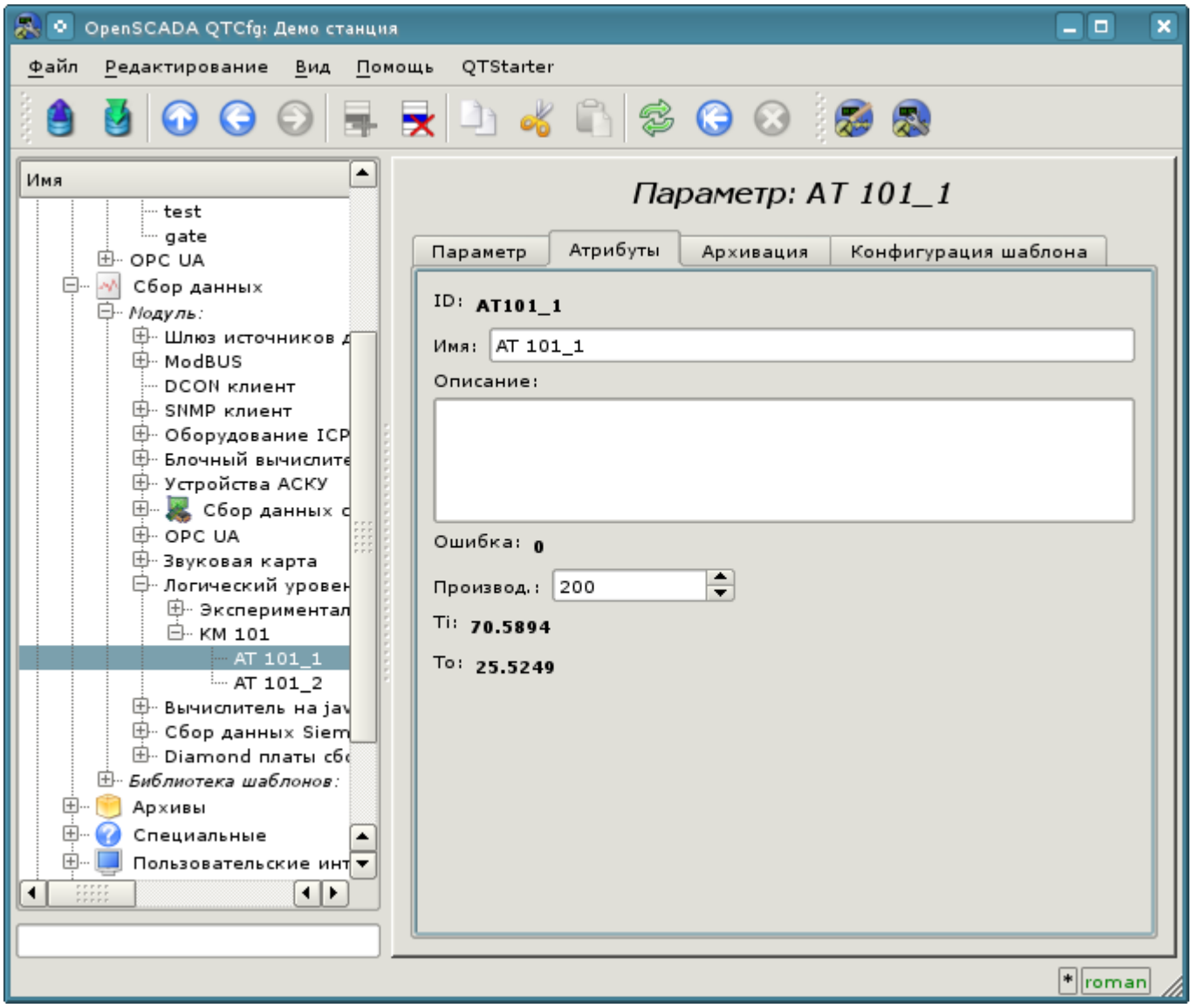

Рис. 4.2.8. Страница атрибутов параметра AT101_1 модуля "LogicLev".

На этом конфигурация обработки данных считается законченной.

4.3. Типизированные параметры источников данных

В предыдущих разделах был описан механизм подключения источника данных по объекту аппарата ("Воздушный холодильник"), который предусматривает объединение всех сигналов в одном объекте параметра источника данных. Однако более распространённым подходом является формирование объекта параметра вокруг сигнала, например, "Температура на выходе холодильника AT101_1 (TE1314_1)".

Создание объекта параметра вокруг сигнала позволяет формализовать его описание до шаблонов аналоговых и дискретных сигналов, включив в них всю необходимую обработку, сигнализацию и другую характерную информацию. Для простой конфигурации типизированных аналоговых и дискретных сигналов в библиотеках OpenSCADA предусмотрены шаблоны параметров, а многие образы визуального представление адаптированны к работе и связыванию с такими параметрами прямо, без детализации по атрибутам.

Обычно для формирования объекта параметра на основе шаблона используется модуля логического уровня LogicLev, как это было описано в предыдущем разделе. Однако ряд модулей, в том числе и ModBus предоставляют возможность сразу создавать логические параметры, на основе шаблона. Добавим новые объекты параметра, открыв в конфигураторе страницу нашего объекта контроллера "ModBus", ранее созданного, и в контекстном меню пункта "KM101" нажмём "Добавить".

Объект аналогового параметра назовём "TE1314_1" и имя "TE1314_1" (рис.4.3.1). Тип параметра установим в "Логический (logic)", шаблон параметра выберем "base.anUnif", описание установим в "Температура на выходе AT101_1", установим флажки "Включать" и "Включен". Далее нам нужно настроить шаблон параметра, в появившейся вкладке "Конфигурация шаблона" (рис.4.3.2): поле "Вход" устанавливаем в значение адреса ModBus-регистра этого параметра "R:101"; поле "Максимум шкалы модуля" устанавливаем в значение 65535, что соответствует 100 °C. Далее переходим во вкладку "Атрибуты" (рис.4.3.3) и устанавливаем некоторые поля: "Ед. изм" устанавливаем в "град. С"; "Шкала минимум" в "0"; "Шкала максимум" в "100"; "Граница верхняя ав." в "40"; "Граница верхняя пред." в "30". Сохраним объект параметра.

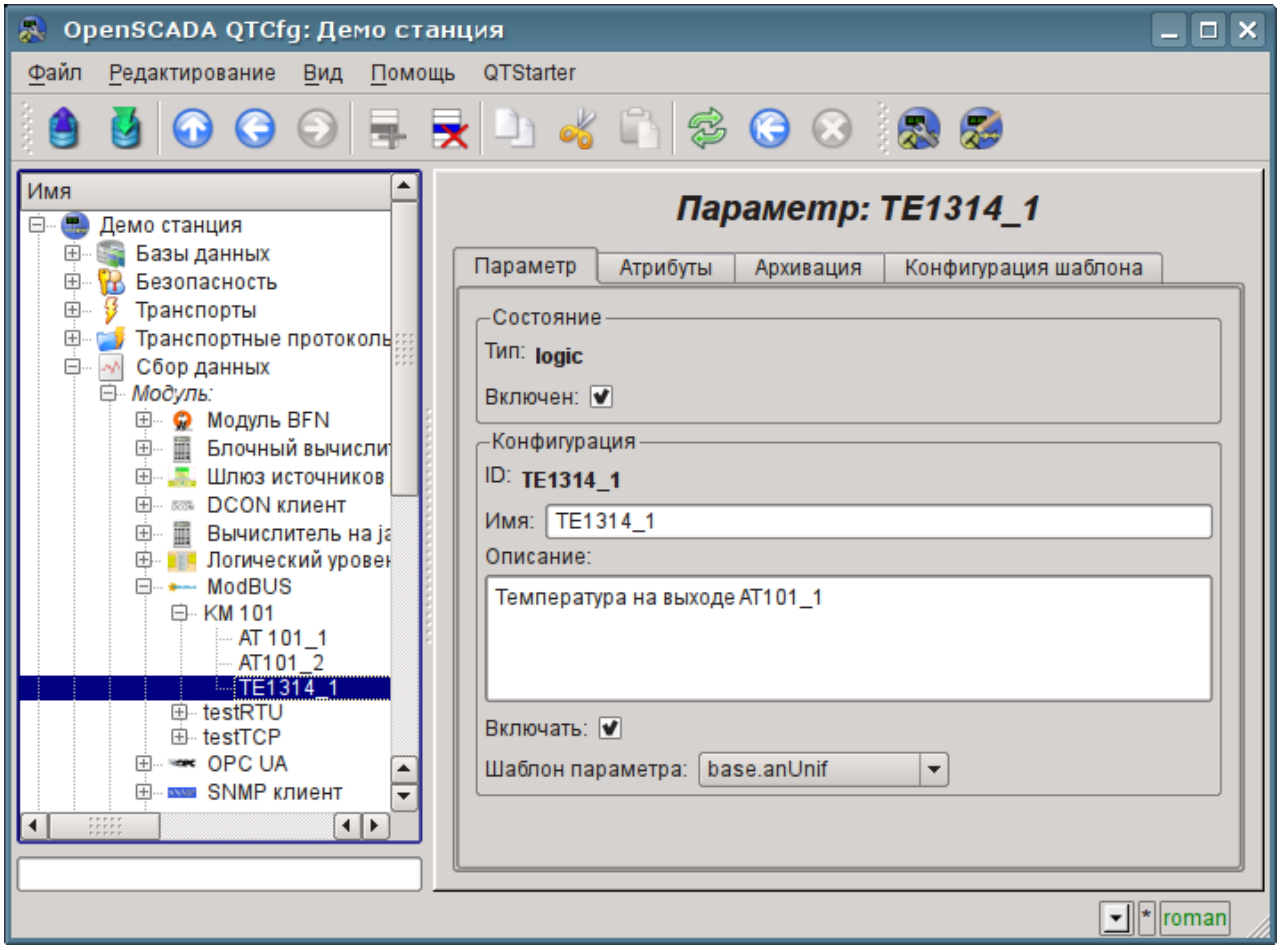

Рис. 4.3.1. Страница логического параметра "TE1314_1", модуля "ModBus".

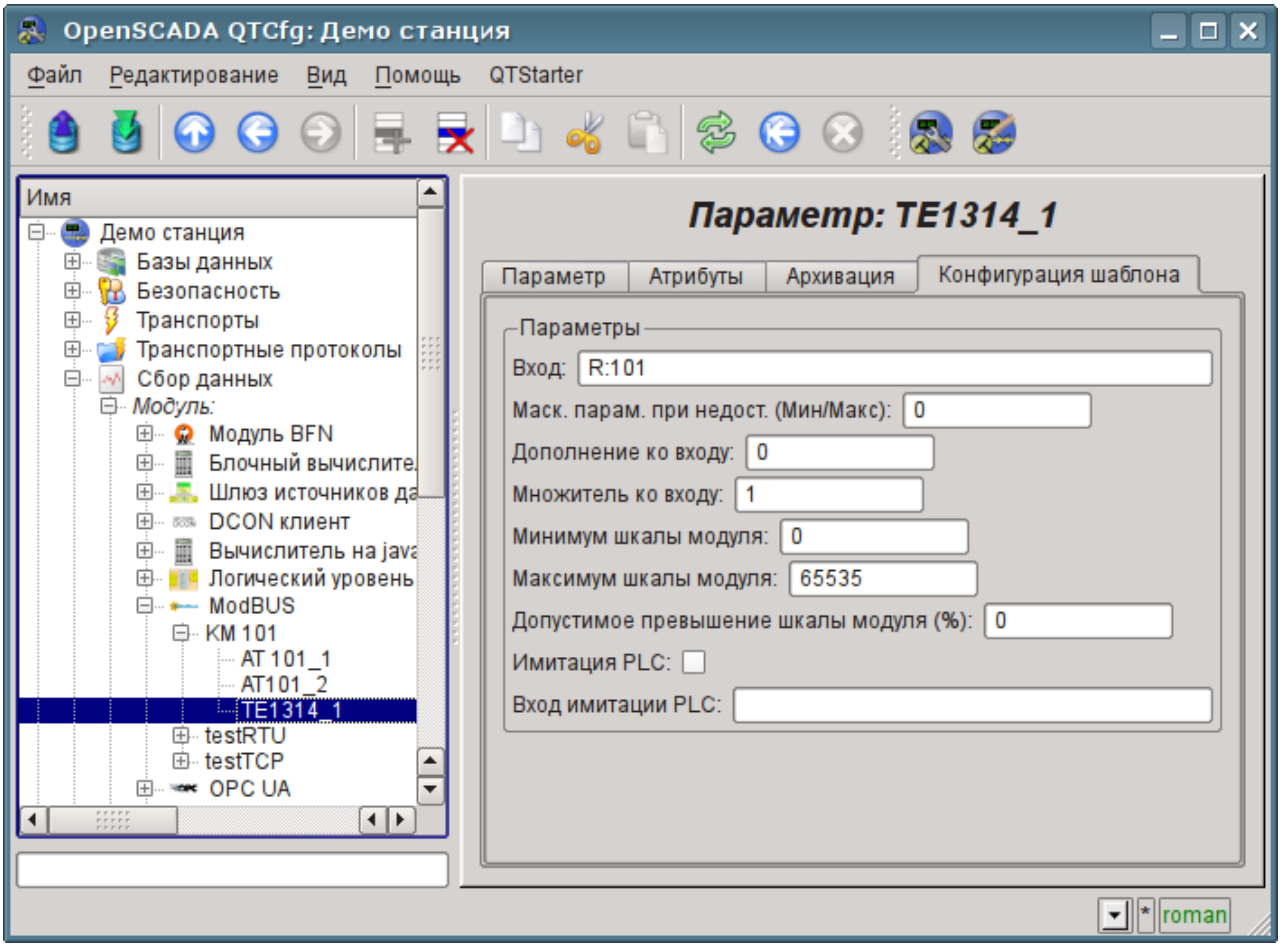

Рис. 4.3.2. Страница конфигурации шаблона параметра "TE1314_1", модуля "ModBus".

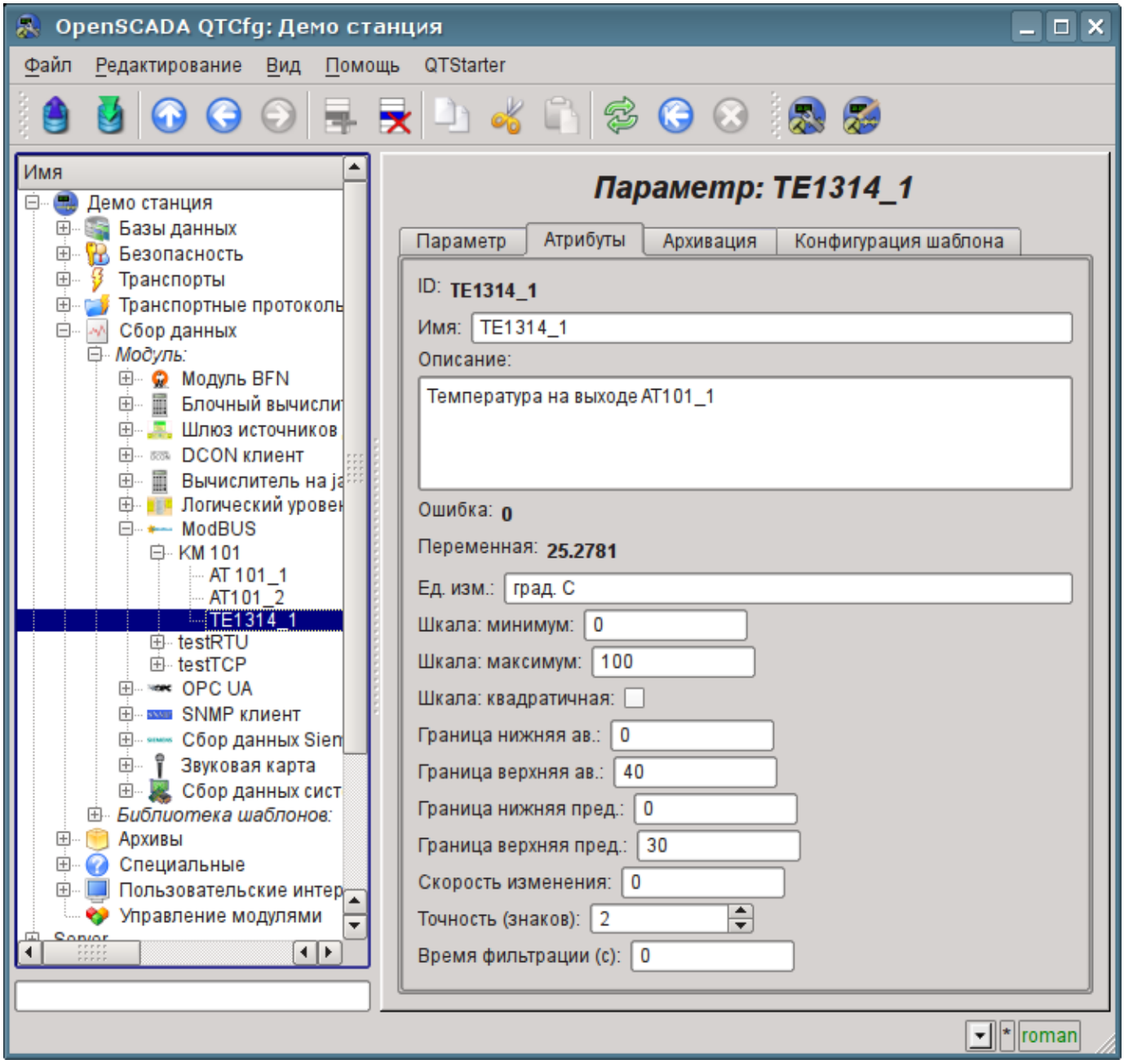

Рис. 4.3.3. Страница атрибутов параметра "TE1314_1", модуля "ModBus".

Объект дискретного параметра назовём: "KSH102" и имя "КШ102". Тип параметра установим в "Логический (logic)", шаблон параметра выберем "base.digitBlockUnif", установим флажки "Включать" и "Включен". Далее нам нужно настроить шаблон параметра, в появившейся вкладке "Конфигурация шаблона" (рис.4.3.4): поле "Команда 'Открыть'" устанавливаем в значение адреса ModBus-бита этого параметра "C:100:rw"; поле "Состояние 'Открыт'" устанавливаем в значение адреса ModBus-бита "C:101"; поле "Состояние 'Закрыть'" устанавливаем в значение адреса ModBus-бита "C:102"; поле "Время удерж. команды (с)" устанавливаем в 0, поскольку команда не импульсная. Далее переходим во вкладку "Атрибуты" (рис.4.3.5) и удостоверяемся в доступности команды и состояний. Сохраним объект параметра.

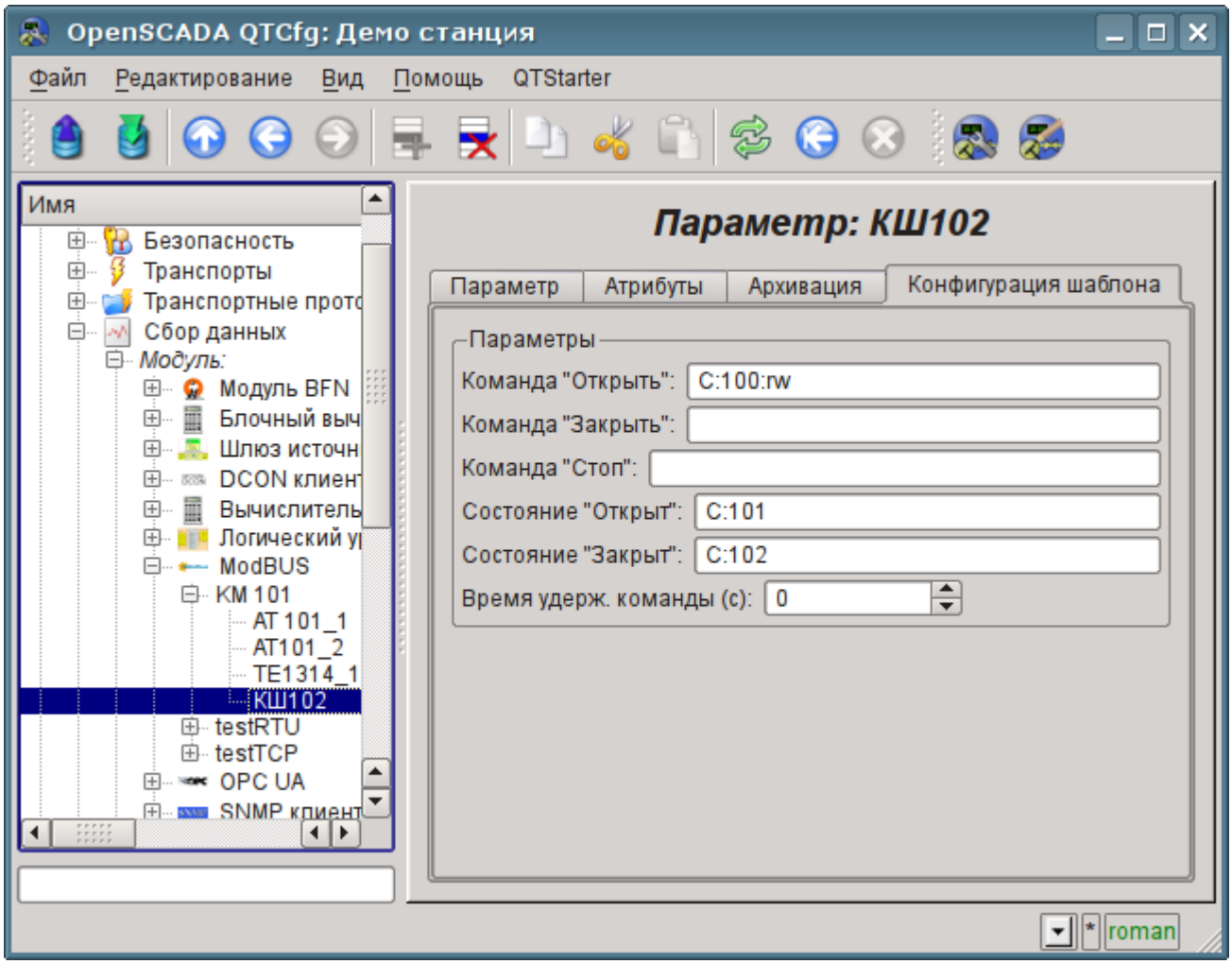

Рис. 4.3.4. Страница конфигурации шаблона параметра "KSH102", модуля "ModBus".

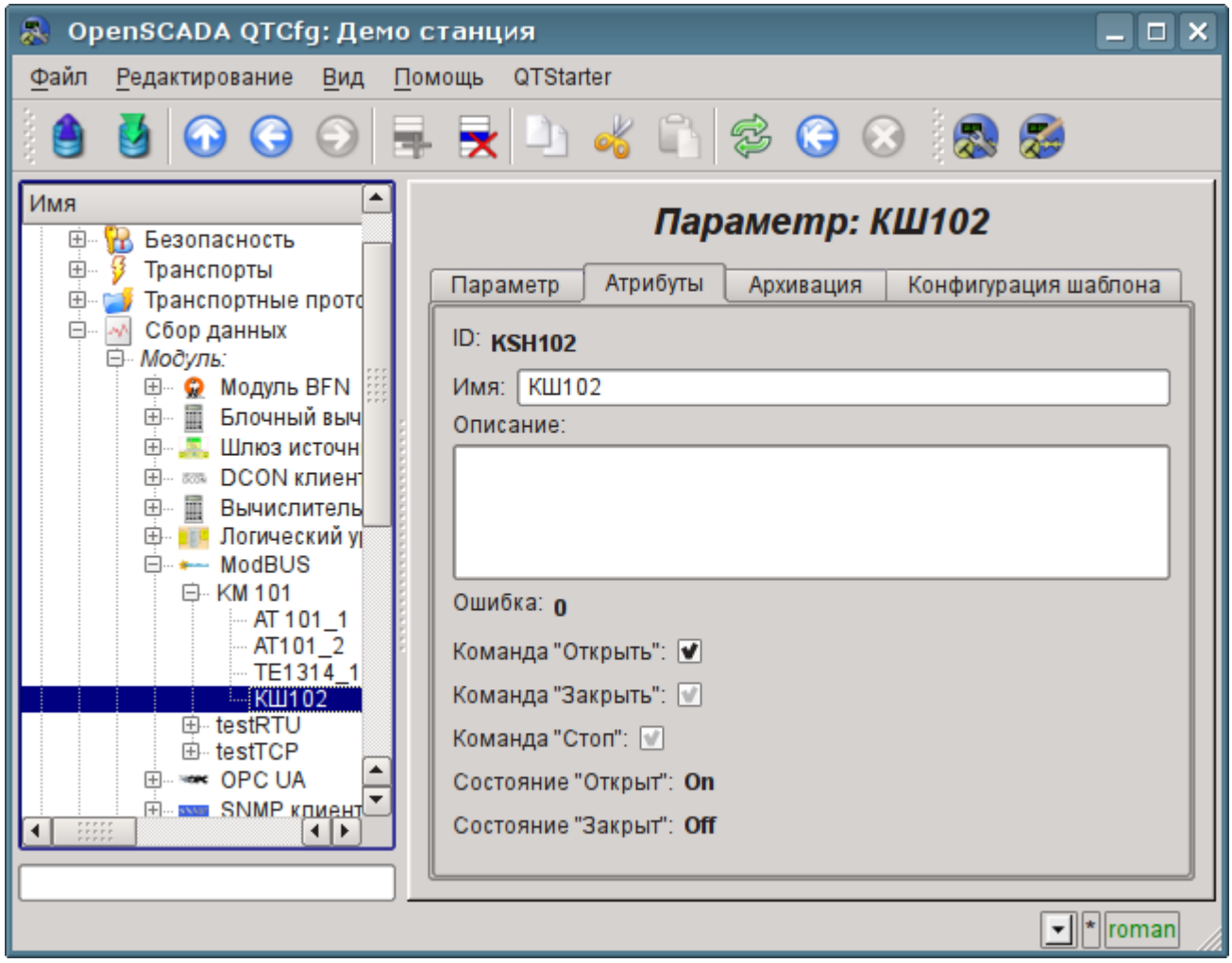

Рис. 4.3.5. Страница атрибутов параметра "KSH102", модуля "ModBus".

4.4. Включение архивирования данных ТП

Во многих задачах требуется ведение истории параметров ТП. Для включения архивирования атрибутов "Ti" и "To" параметров AT101_1 и AT101_2 в ранее созданном контроллере модуля "LogicLev" достаточно на вкладке "Архивация" страницы параметров выбрать, какие атрибуты архивировать и на каких архиваторах (рис.4.4.1). Выберем архивацию атрибутов "Ti" и "To" в архиваторе "FSArch.1s". Тоже самое можно сделать для атрибута "var" аналогового параметра "ModBus.KM101.TE1314_1" и "com" дискретного параметра "ModBus.KM101.KSH102".

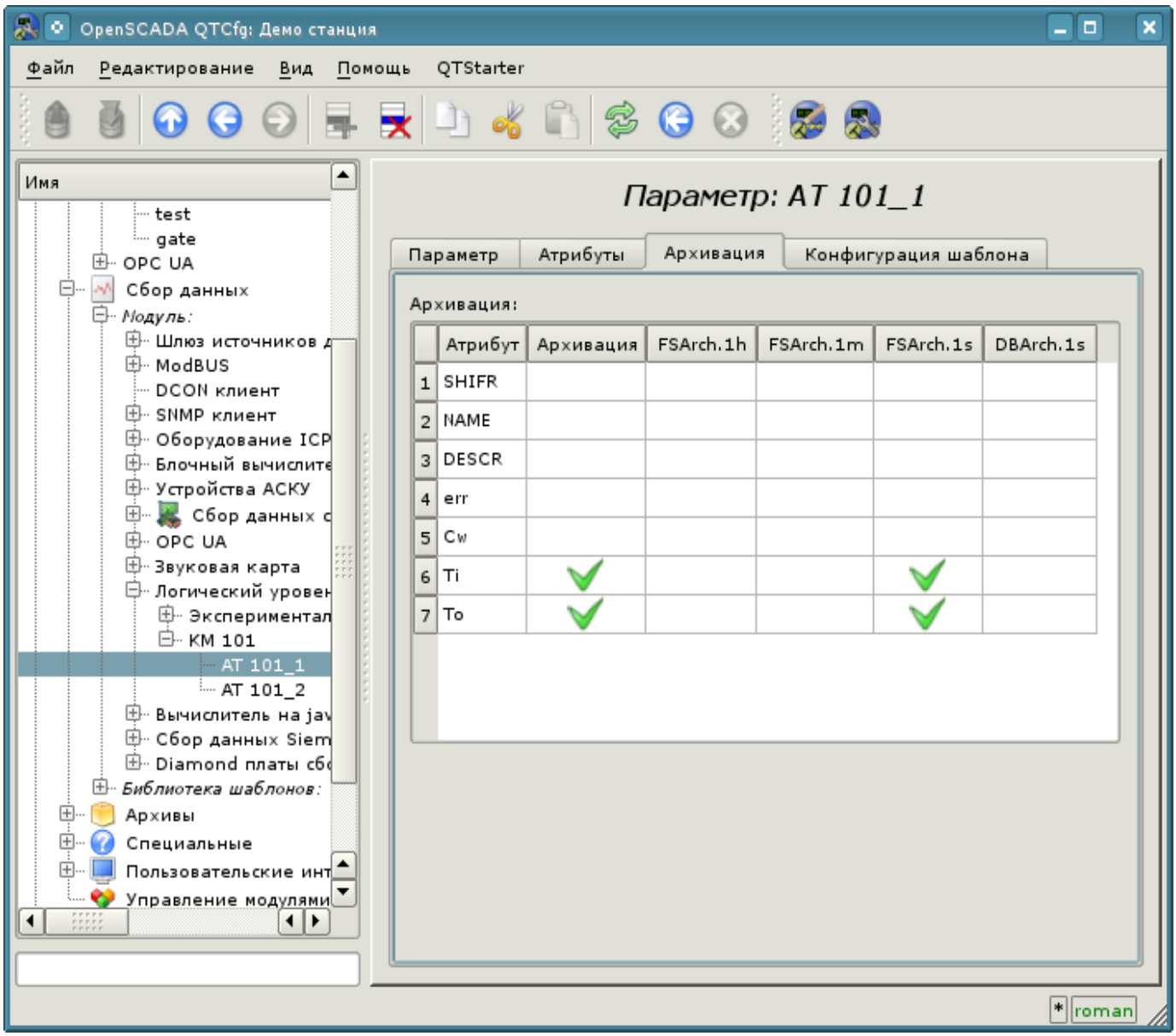

Рис. 4.4.1. Вкладка "Архивация" параметра AT101_1 модуля "LogicLev".

В результате этой операции будут автоматически созданы объекты архивов для выбранных атрибутов. Например, объект архива для атрибута "Ti" параметра AT101_1 представлен на рис.4.4.2.

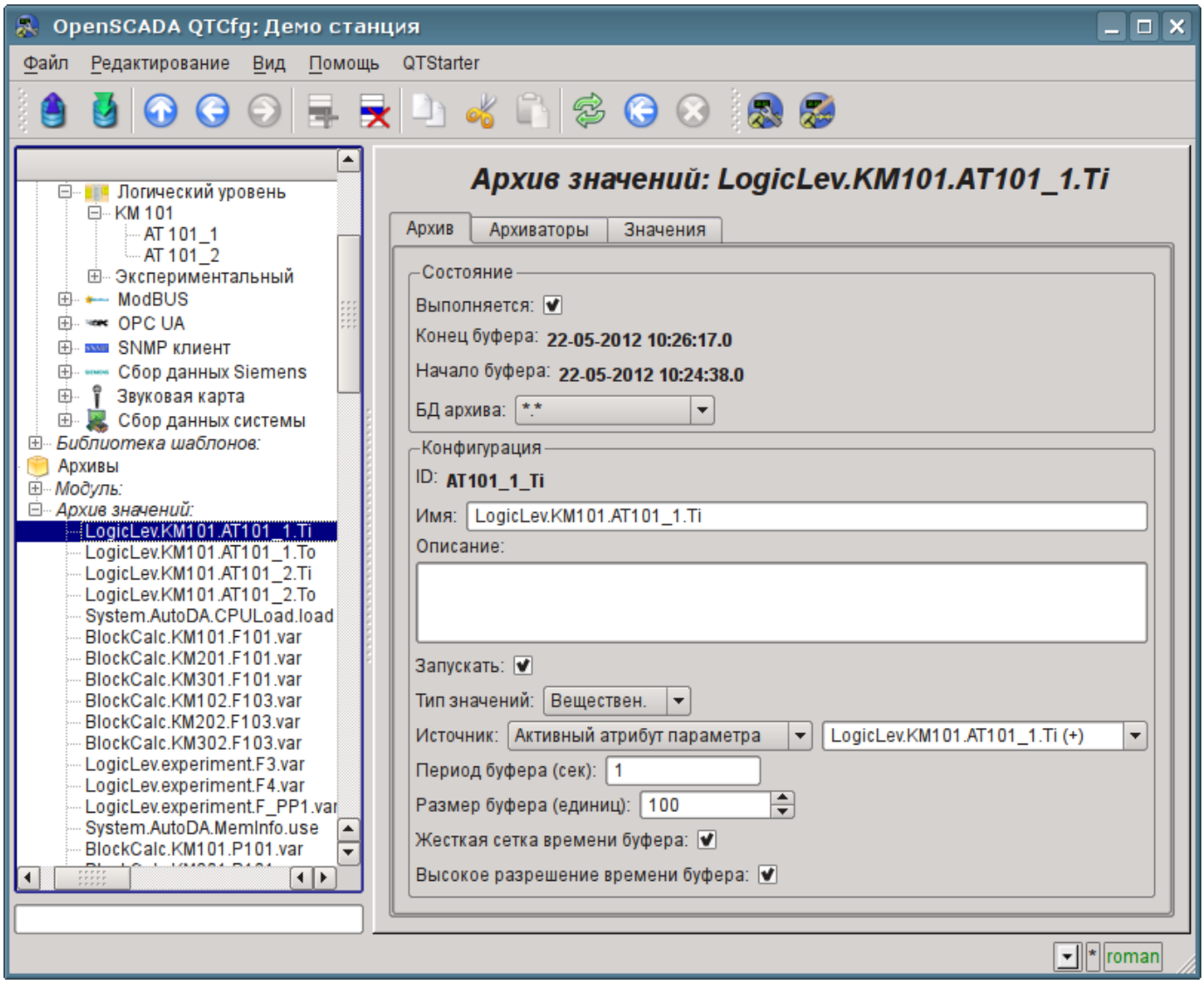

Рис. 4.4.2. Страница объекта архива атрибута "Ti" параметра AT101_1.

Обычно настройки архива менять не нужно, однако, если потребуется особая конфигурация, то её можно сделать на указанной выше странице. Чаще может понадобиться получение информации об архиве. Например, узнать размер архива как по времени, так и на носителе, а также взглянуть на график параметра (рис.4.4.3).

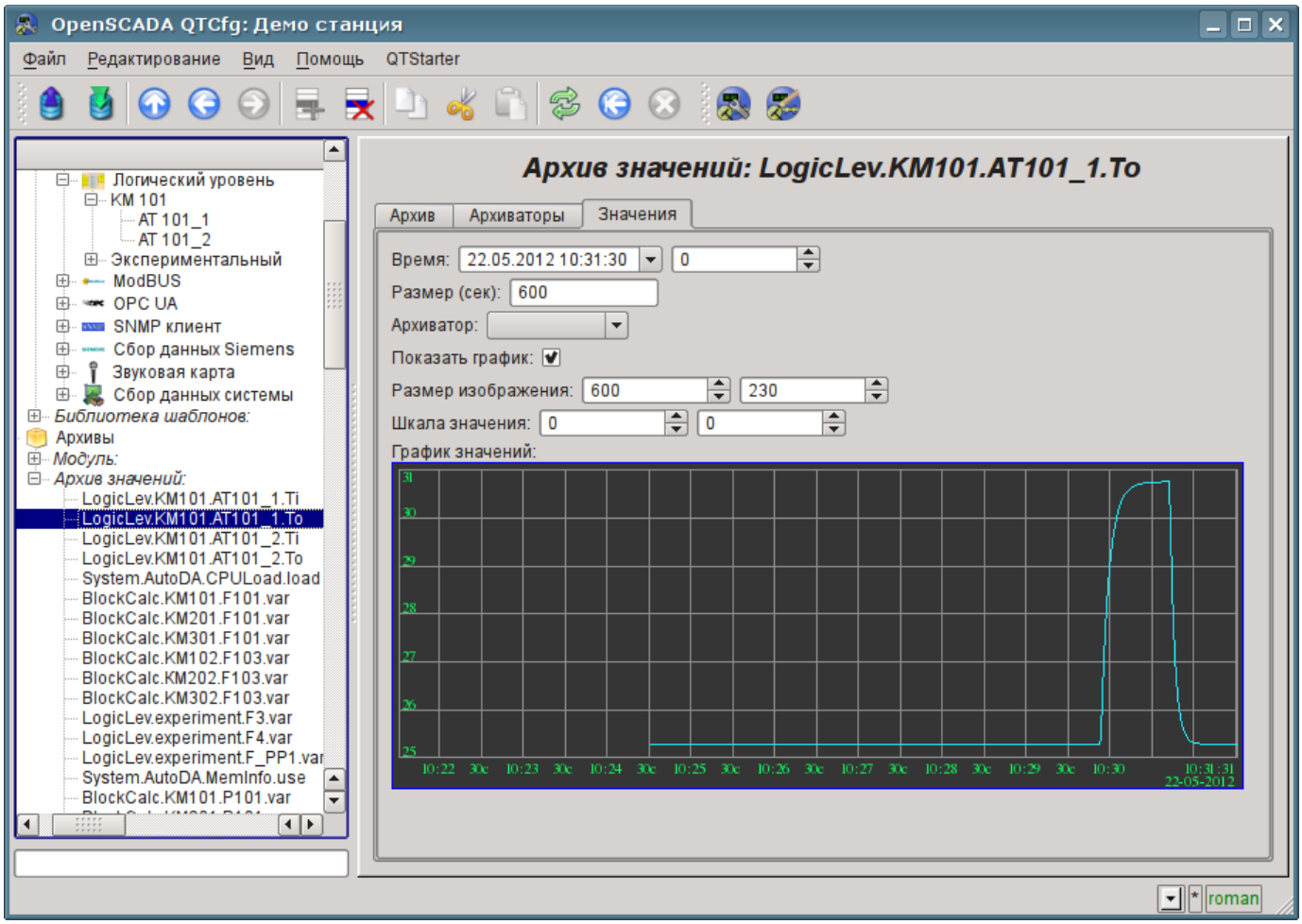

Рис. 4.4.3. Вкладка "Значения" страницы объекта архива атрибута "To" параметра AT101_1.

5. Формирование визуального представления

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

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

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

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

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

Интерфейсы пользователя-оператора в OpenSCADA реализуются проектами визуализации. В библиотеке основных элементов пользовательского интерфейса присутствует шаблон типового проекта визуализации, основанного на концепции объектов сигнализации и видов отображения. Пользователь может начать создавать свою концепцию интерфейса визуализации, в виде нового проекта, а может использовать указанный шаблон. Для реализации нового проекта визуализации потребуются знания третьего уровня и значительные усилия, что находится за рамками этого документа. Поэтому будем рассматривать создание интерфейса визуализации на основе доступного шаблонного проекта.

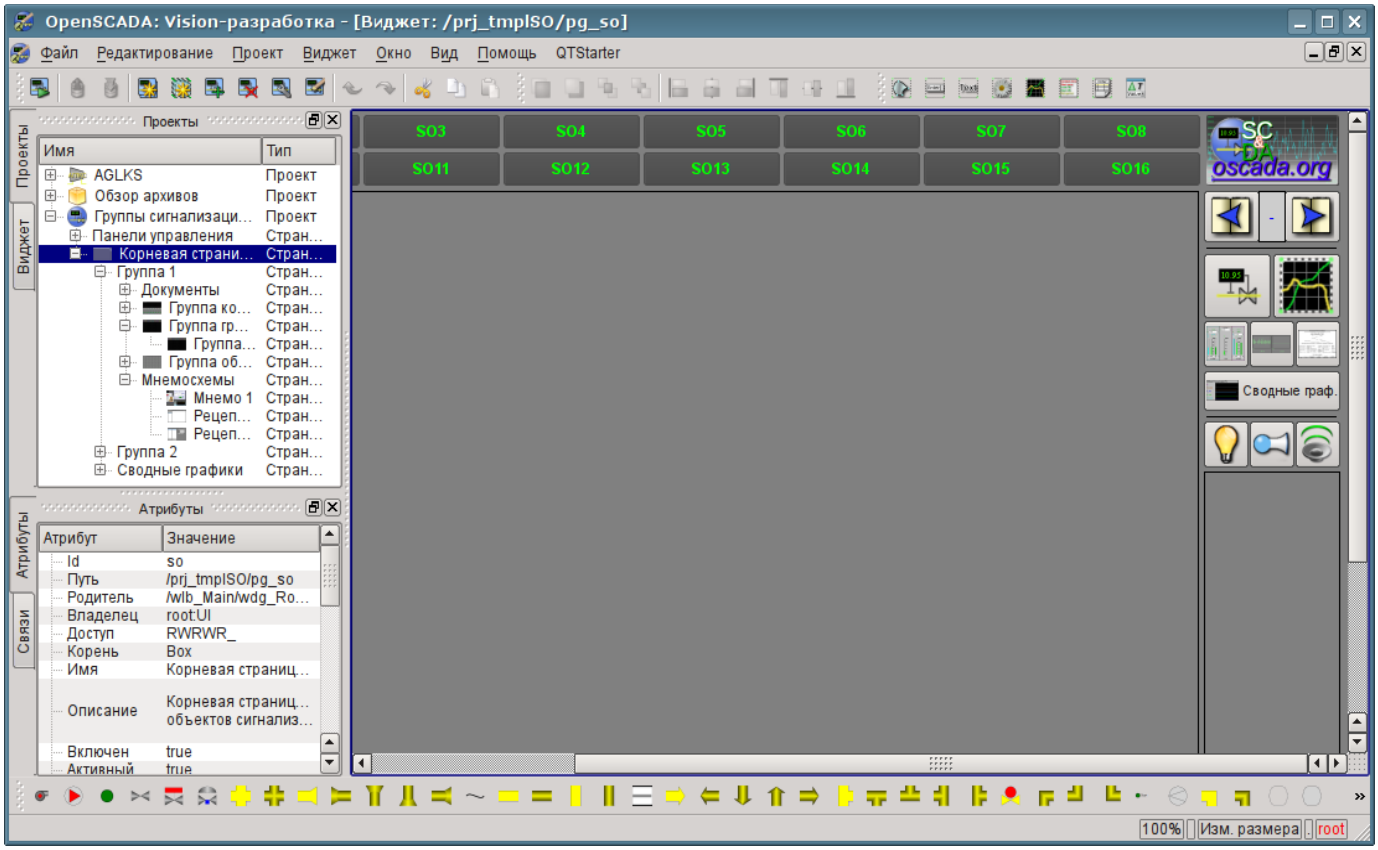

Рис. 5.1. Шаблонный проект по концепции объектов сигнализации.

Данный шаблон содержит две ветви: "Панели управления" и "Корневая страница". Ветвь "Панели управления" содержит набор кадров типовых панелей управления и специализированных кадров. Ветвь "Корневая страница", с корневой страницей в основе, содержит подветви объектов сигнализации "Группа 1", "Группа 2" и отдельную ветвь сводных графиков "Сводные графики". Подветви объектов сигнализации "Группа {n}" имеют цифровой идентификатор и могут расширяться добавлением вплоть до 16. Присутствие подветви "Группа {n}" отражается на активации соответствующей кнопки объекта сигнализации корневой страницы, что позволяет на них переключаться. Каждая подветвь "Группа {n}" содержит контейнеры или шаблоны видов отображения, обычно: "Мнемосхемы", "Группы графиков", "Группы контуров", "Группы обзорных кадров" и "Документы". Наличие страниц в контейнерах отображений включает возможность выбора этого отображения, для соответствующего объекта сигнализации корневой страницы. Детальнее о структуре корневой страницы можно ознакомиться по ссылке http://wiki.oscada.org/Using/GraphicElementsLibraries/MainElements#h1036-45.

Для создания собственного проекта визуализации пользователь может скопировать шаблонный проект и назвать согласно своей тематике. Мы-же продолжим непосредственно с шаблонным проектом, помещая наши страницы в контейнера мнемосхем и групп графиков.

5.1. Добавление шаблонной страницы в проект и подключение динамики

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

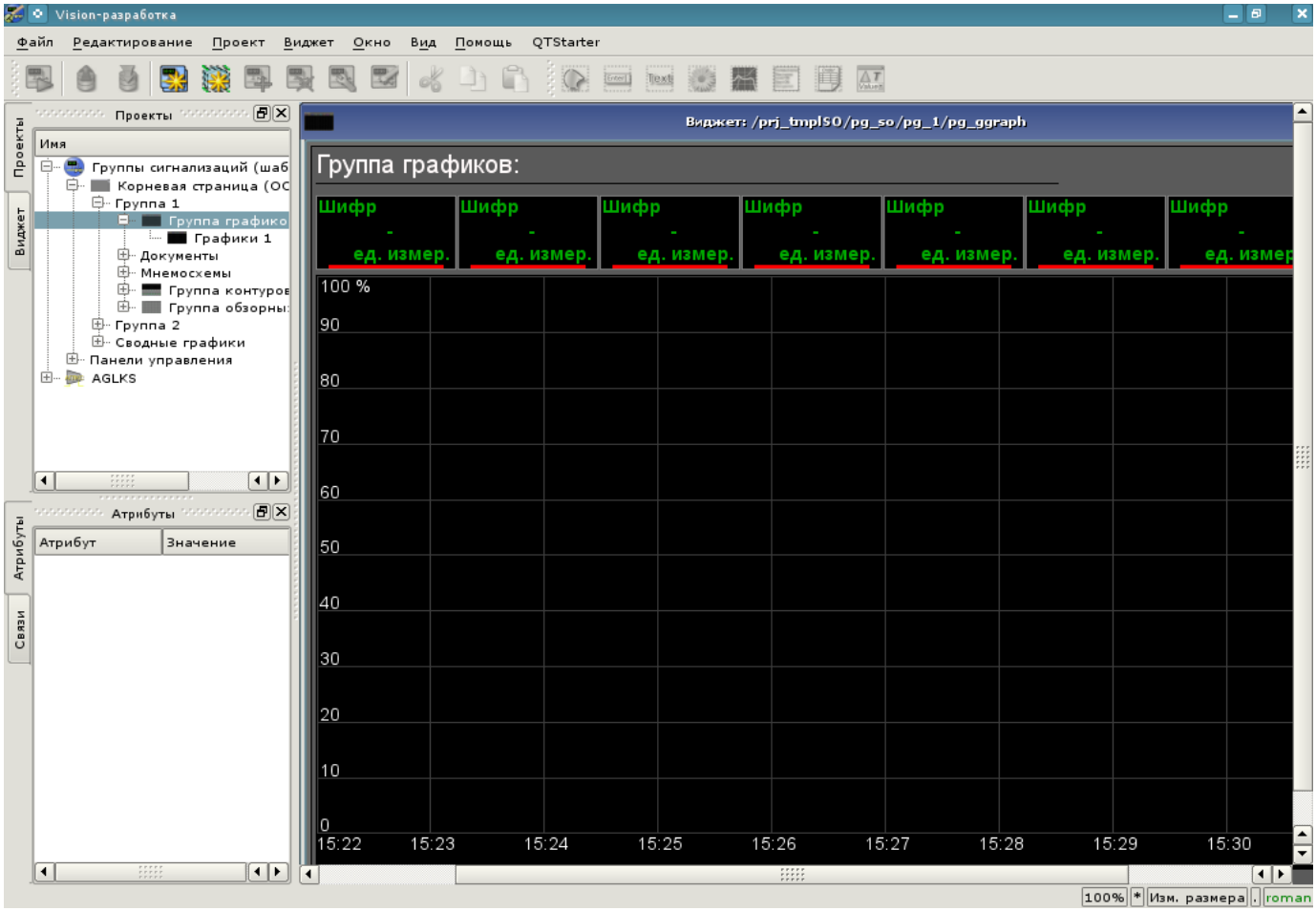

Рис. 5.1.1. Шаблонная страница "Группа графиков".

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

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

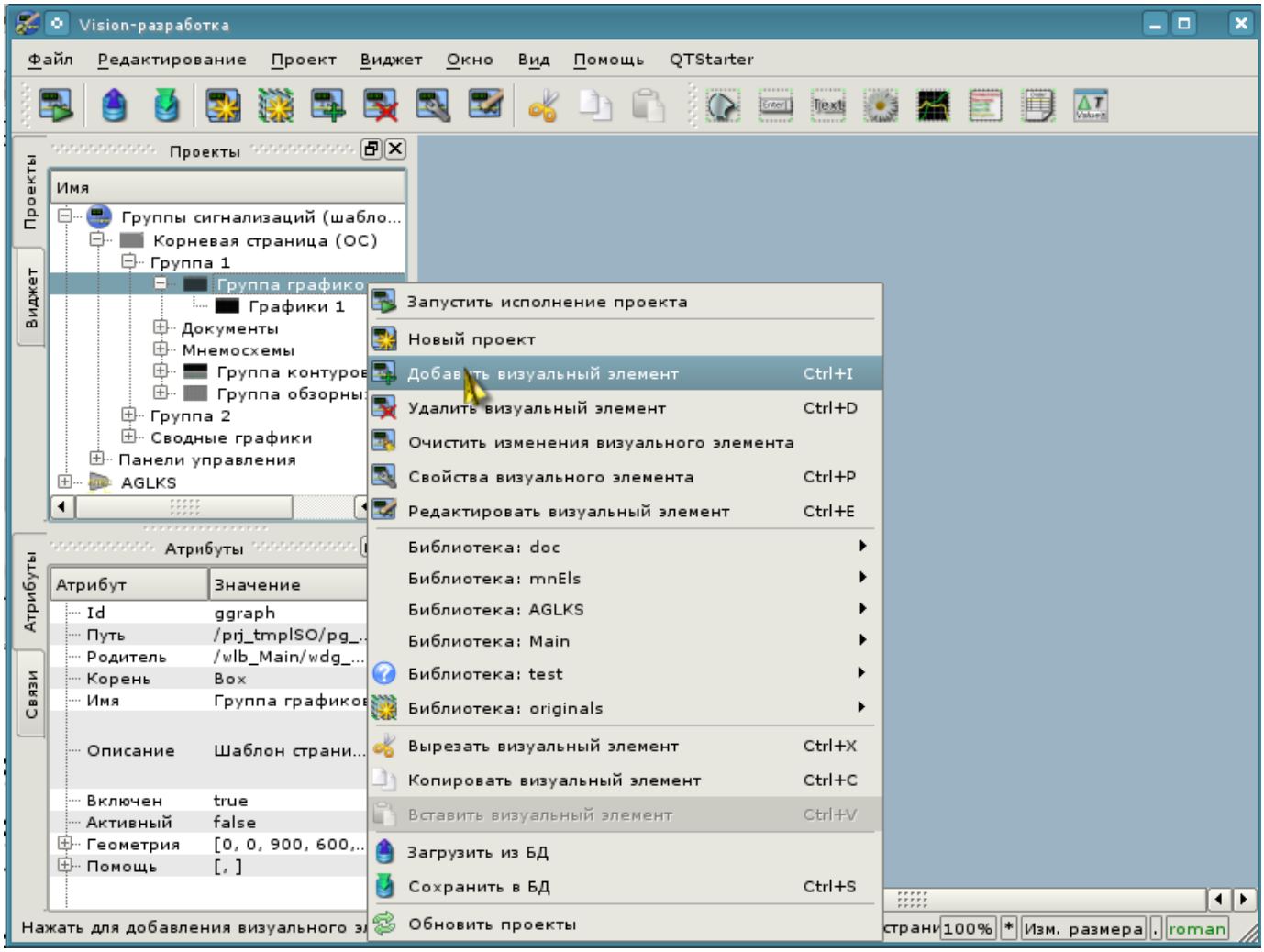

Рис. 5.1.2. Добавление группы графиков "Графики 2".

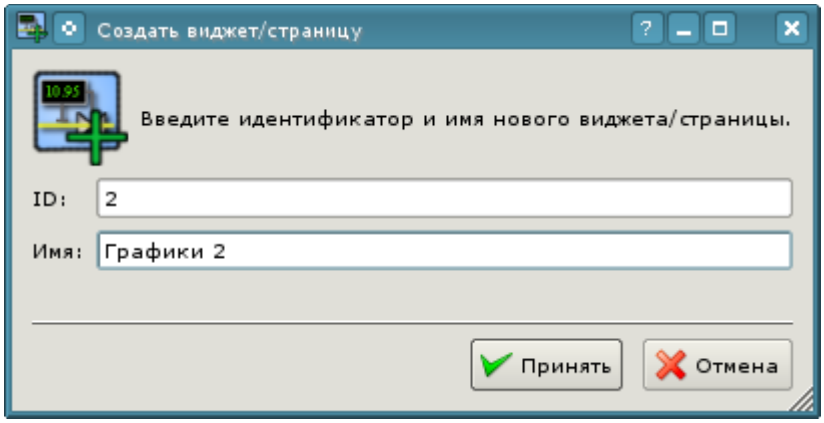

Рис. 5.1.3. Диалог ввода идентификатора и имени.

После подтверждения ввода имени будет создана новая страница. Однако, для её активации нам понадобится её включить. Включить страницу можно в диалоге редактирования свойств страницы (рис.5.1.4). Открыть эту страницу можно посредством выбора пункта меню "Свойства визуального элемента" в контекстном меню вновь созданной страницы. Создать страницу в логическом контейнере, на основе шаблона, можно и простым копированием шаблона внутрь себя.

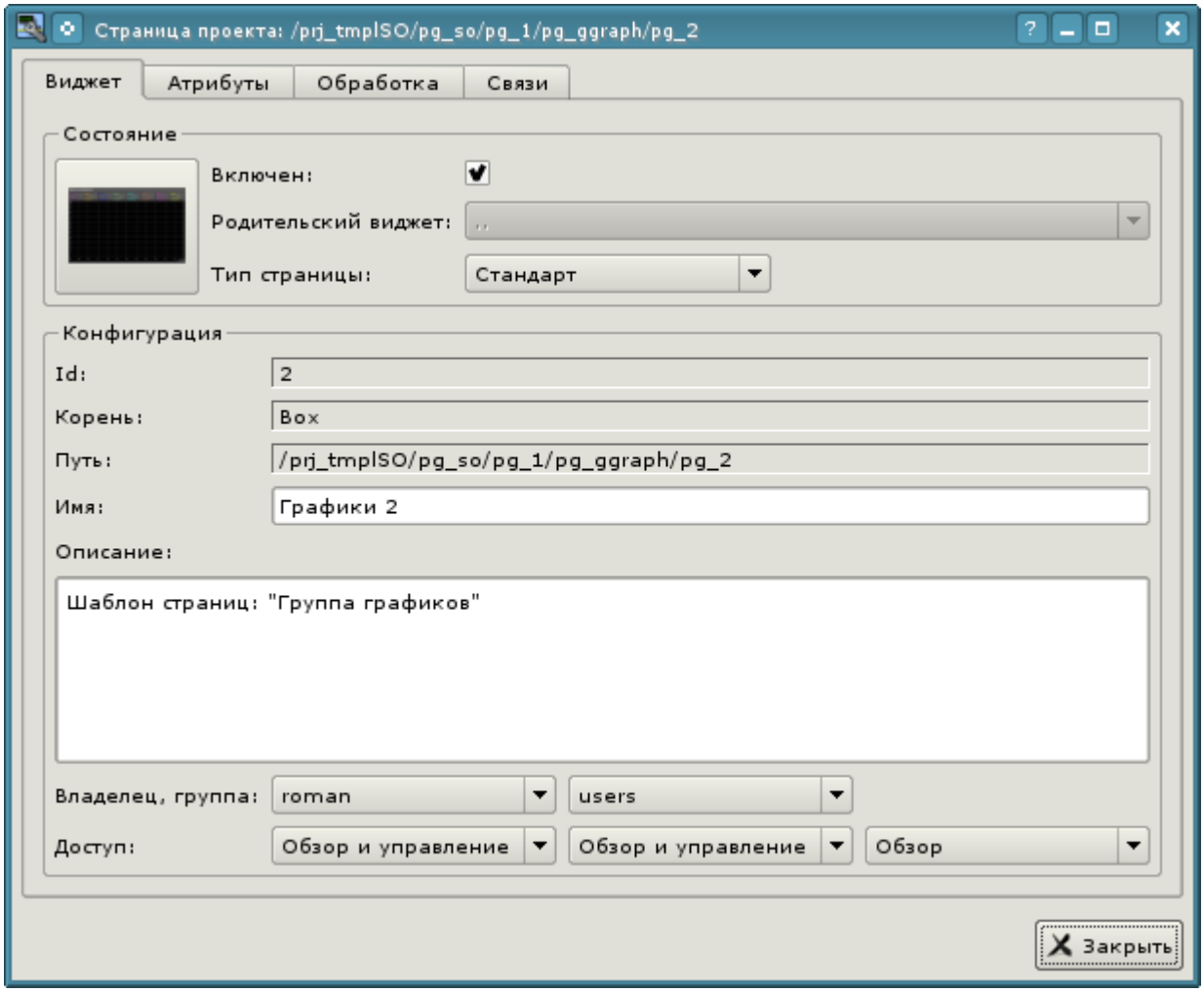

Рис. 5.1.4. Диалог редактирования свойств визуального элемента.

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

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

Если заранее предусмотреть наличие атрибутов, указанных нами постоянными в шаблоне параметра контроллера, то можно будет указывать только параметр, а атрибуты расставятся автоматически, что мы можем увидеть, привязав созданный нами ранее типизированный аналоговый параметр "ModBus.KM101.TE1314_1".

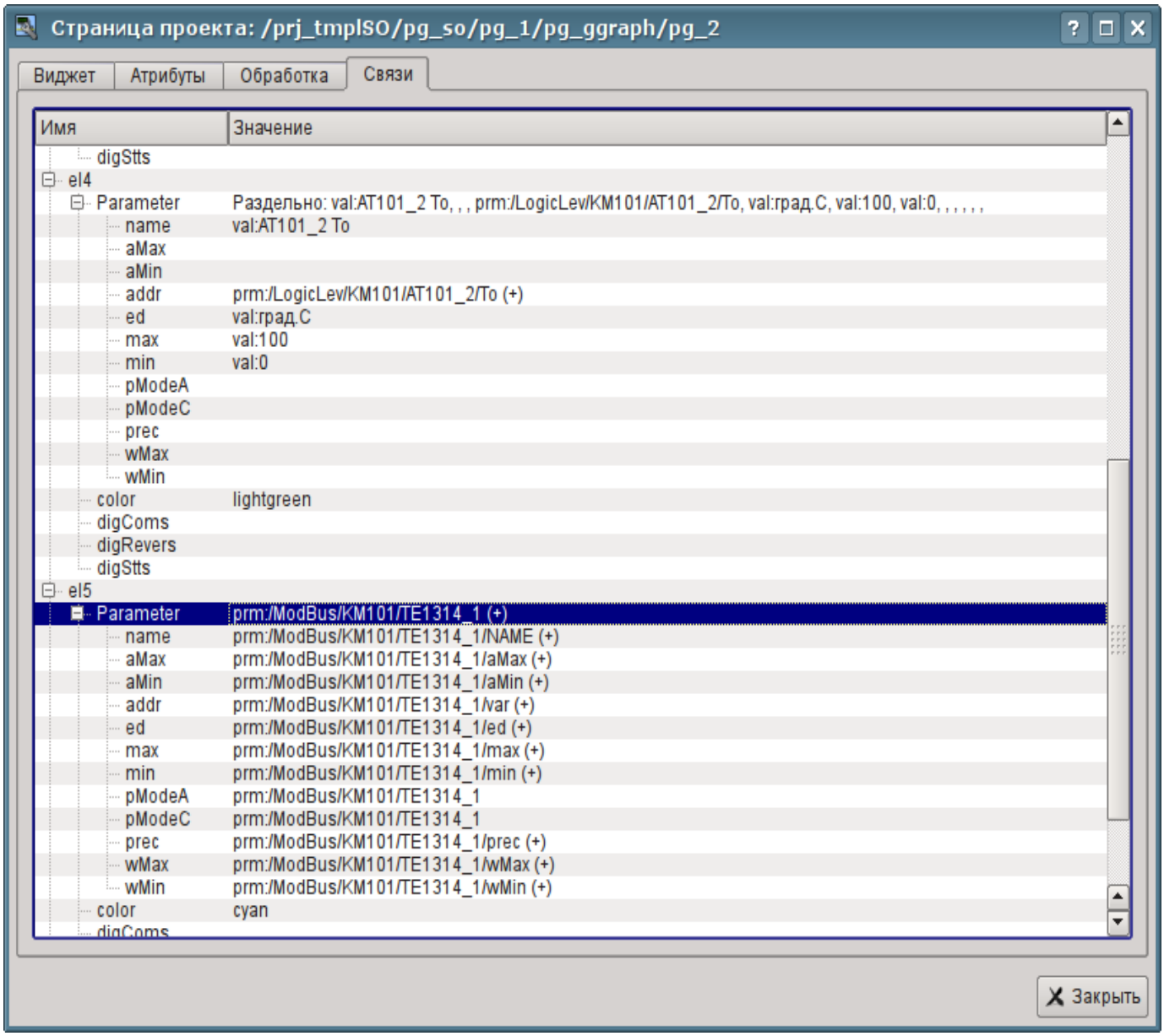

Рис. 5.1.5. Вкладка "Связи" диалога редактирования свойств визуального элемента.

Закончив ввод связей можем проверить что получилось в результате наших усилий. Для этого закроем окно диалога свойств и запустим проект "Группы сигнализаций (шаблон)" на исполнение, про кнопку запуска мы помним из первых глав. Затем выберем графики и переключимся на вторую страницу. При безошибочной конфигурации мы должны увидеть что-то подобное изображённому на рис.5.1.6. Заметьте, что для типизированного параметра, с установленными границами нарушений, выход значения за границы отмечается аварийным цветом. Для того что-бы увидеть выход за границу нарушения Вы можете установить значение производительности вентилятора в 100 (рис.4.2.8).

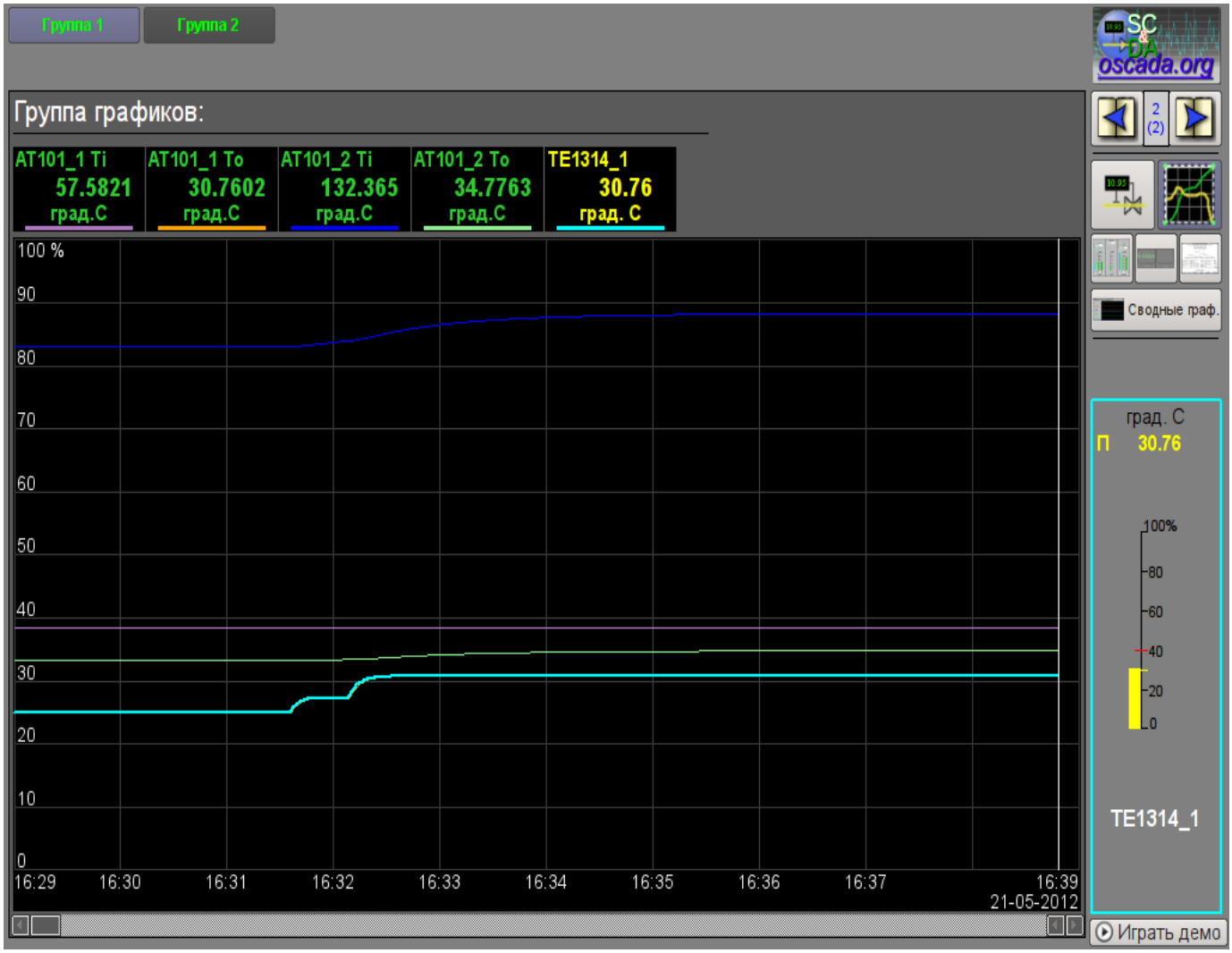

Рис. 5.1.6. Созданная группа графиков с четырьмя подключенными сигналами и одним типизированным параметром.

5.2. Создание нового кадра, мнемосхемы

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

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

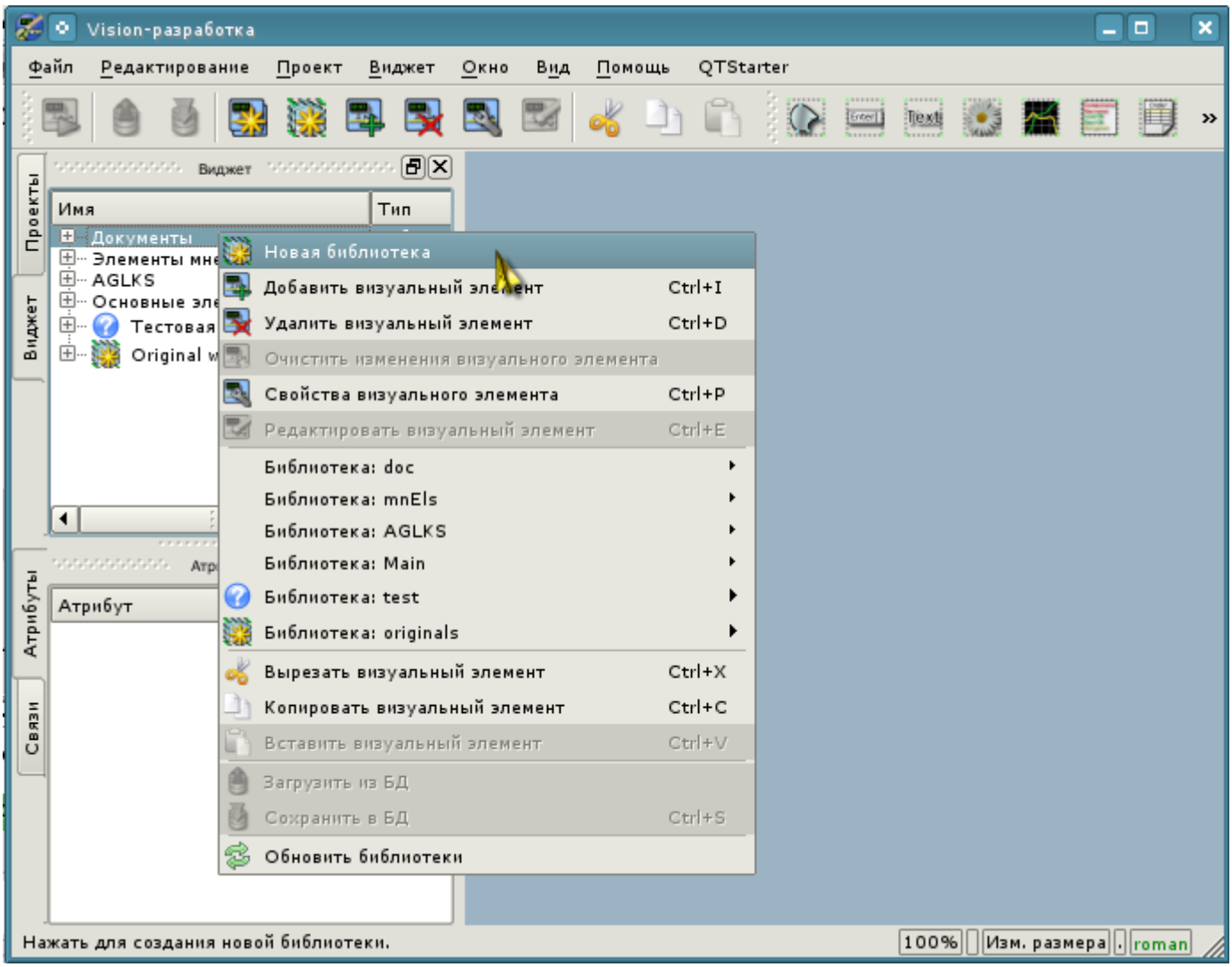

Рис. 5.2.1. Добавление новой библиотеки виджетов.

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

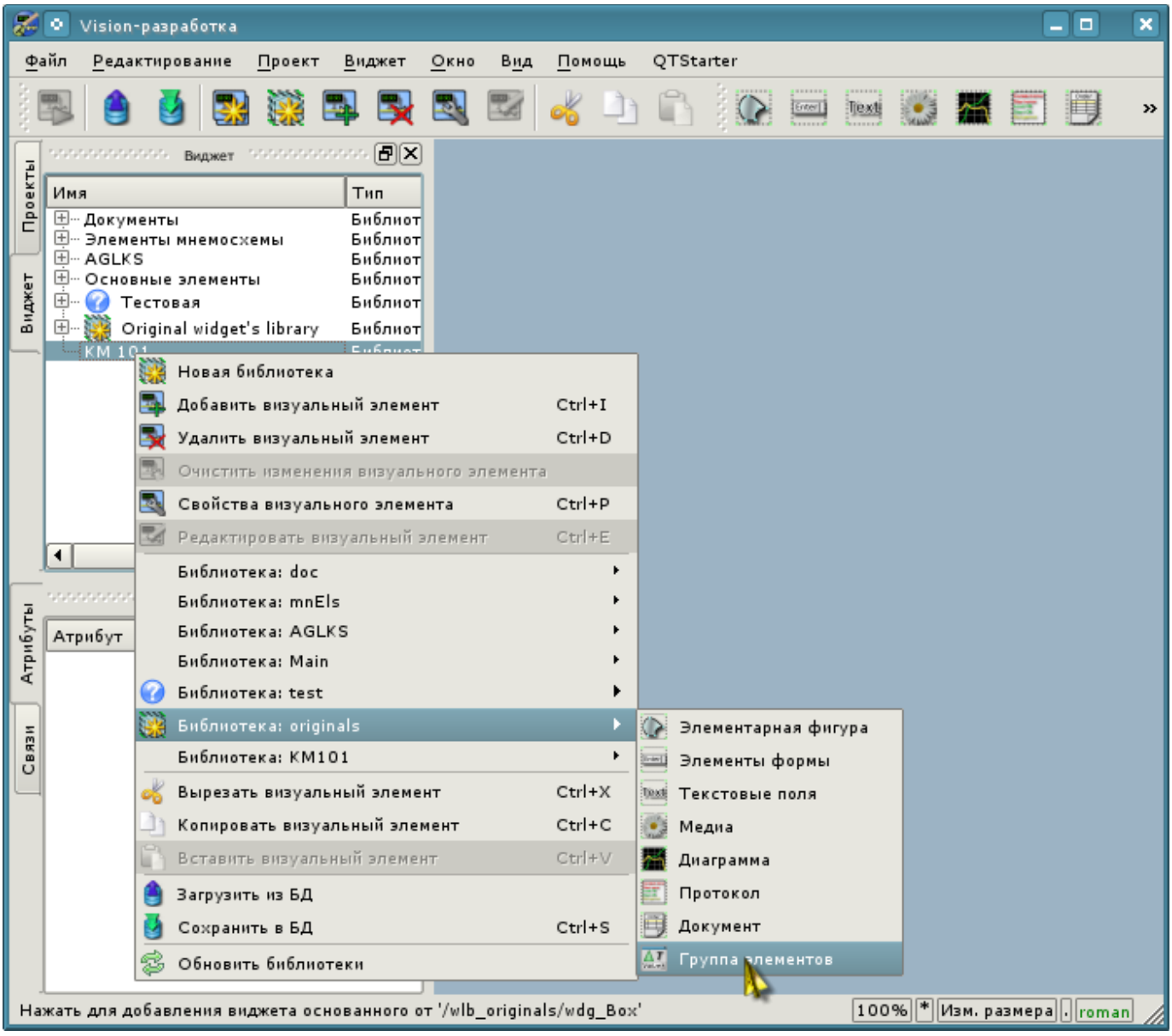

Рис. 5.2.2. Добавление нового кадра.

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

- *Геометрия:ширина* — "900".
- *Геометрия:высота* — "600".
- *Страница:группа* — "so", для включения кадра в контейнер мнемосхем, при исполнении.
- *Фон:цвет* — "#5A5A5A".
- *Граница:ширина* — "1".
- *Граница:цвет* — "black".

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

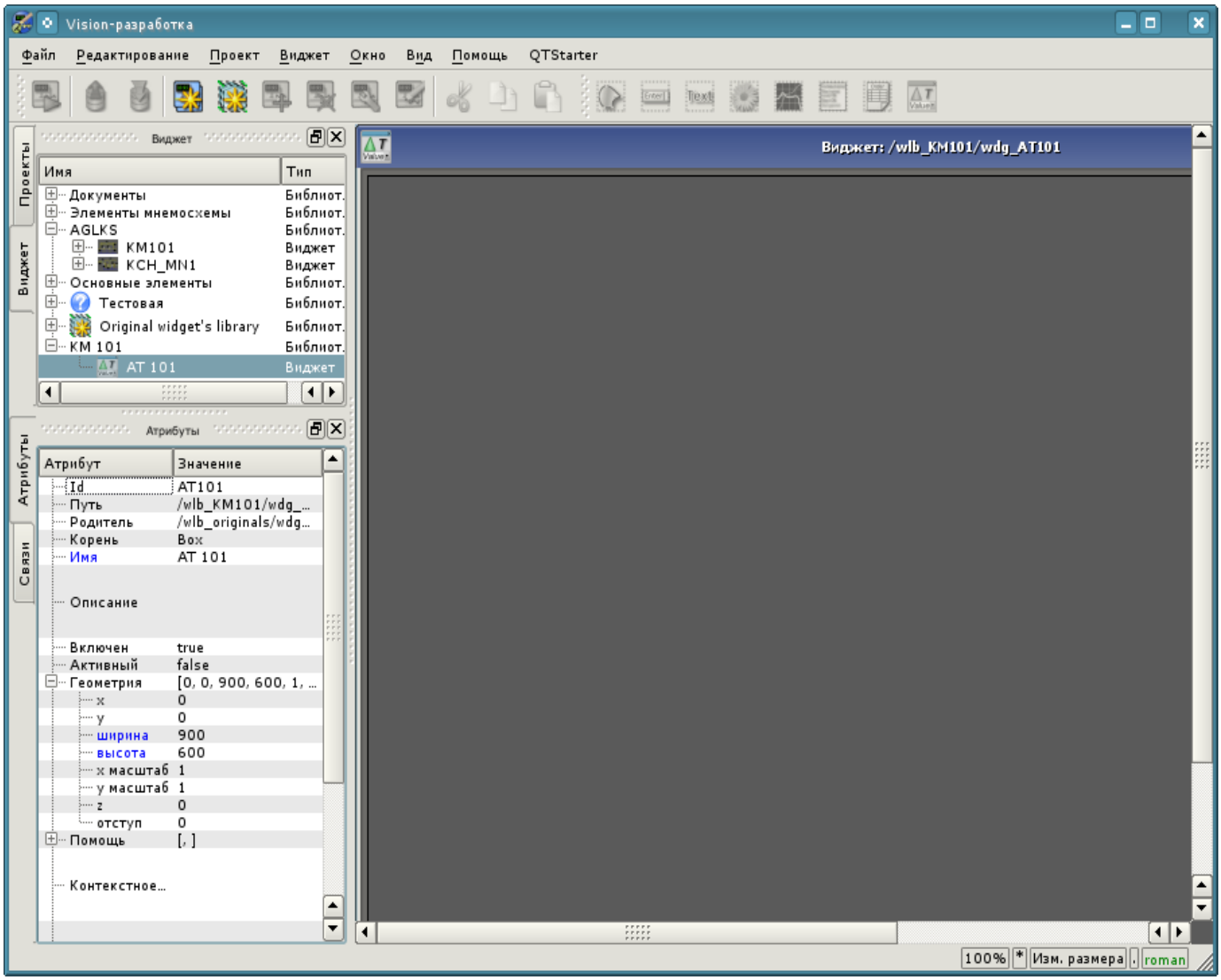

Рис. 5.2.3. Вид нового кадра и установленных атрибутов для мнемосхемы.

Теперь добавим на кадр элементы отображения значения аналогового параметра для наших четырёх сигналов и типизированного параметра "ModBus.KM101.TE1314_1". Для помещения на мнемосхему элемента отображения аналогового сигнала нужно выбрать нашу мнемосхему, а затем в меню окна выбрать пункт меню "Виджет"->"Библиотека: Main"->"Отобр аналог"; после чего появится курсор с образом этого элемента, который нужно подвести в желаемую область мнемосхемы и нажать левую кнопку мыши. В момент добавления появится диалог с запросом имени нового элемента. Добавлять подобным образом будем пять элементов, которые назовём: "A1_Ti", "A1_To", "A2_Ti", "A2_To" и "TE1314_1".

Добавим так-же элемент крана для типизированного дискретного параметра "ModBus.KM101.KSH102", для добавления которого используем библиотечный элемент "Виджет"->"Библиотека: mnEls"->"Кран шаровый" и назовём его "KSH102".

Для отображения списка текущих нарушений поместим на мнемосхему элемент протокола из библиотеки примитивов "Виджет"->"Библиотека: originals"->"Протокол" и назовём его "Protocol". В инспекторе атрибутов установим свойства для протокола:

- *Геометрия:ширина* — "500".
- *Геометрия:высота* — "250".
- *Показать колонки* — "tm;lev;mess".
- *Уровень* — "-1", отображение текущих нарушений любого уровня.

- *Размер, сек* — "0", отображать нарушения на любую глубину.
- *Период слежения (с)* — "1".

Добавленные элементы можно впоследствии расположить как нужно, просто выделяя и перетаскивая мышью. После выполнения всех манипуляций у нас должна получиться мнемосхема с видом, похожим представленной на рис.5.2.4.

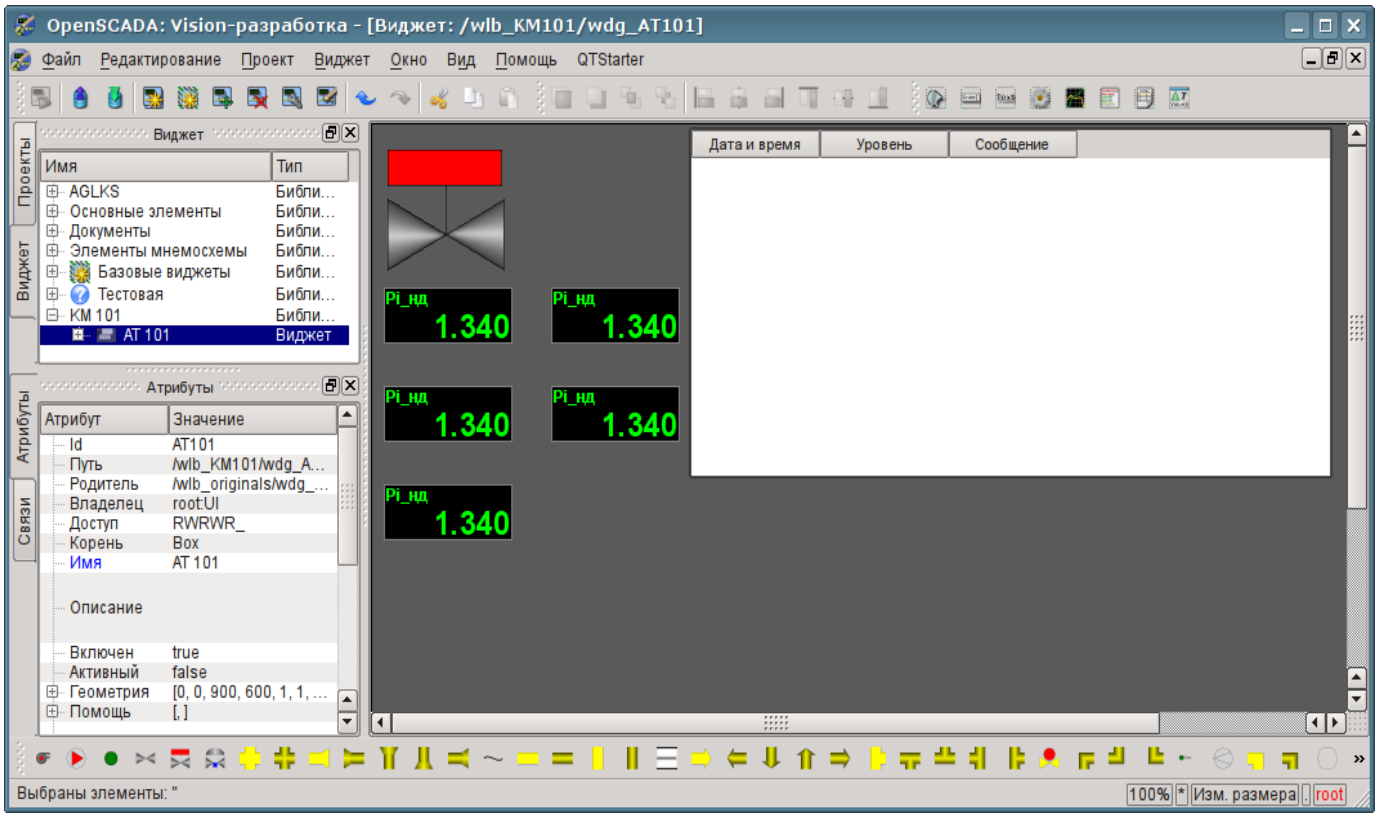

Рис. 5.2.4. Вид нового кадра и установленных атрибутов для мнемосхемы.

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

Поместим нашу мнемосхему в ветвь "Группы сигнализаций (шаблон)"->"Корневая страница"->"Группа 1"->"Мнемосхемы" путём выбора в контекстном меню для пункта "Мнемосхемы" пункта "Библиотека: KM101"->"AT 101". Идентификатор для новой мнемосхемы установим в "2", при этом поле имени оставим пустым.

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

- *pName* — "val:AT101_1 Ti".

Как и в случае с группой графиков, в предыдущем разделе, для типизированных параметров "ModBus.KM101.TE1314_1" и "ModBus.KM101.KSH102" можно указавать только параметр, а атрибуты расставятся автоматически.

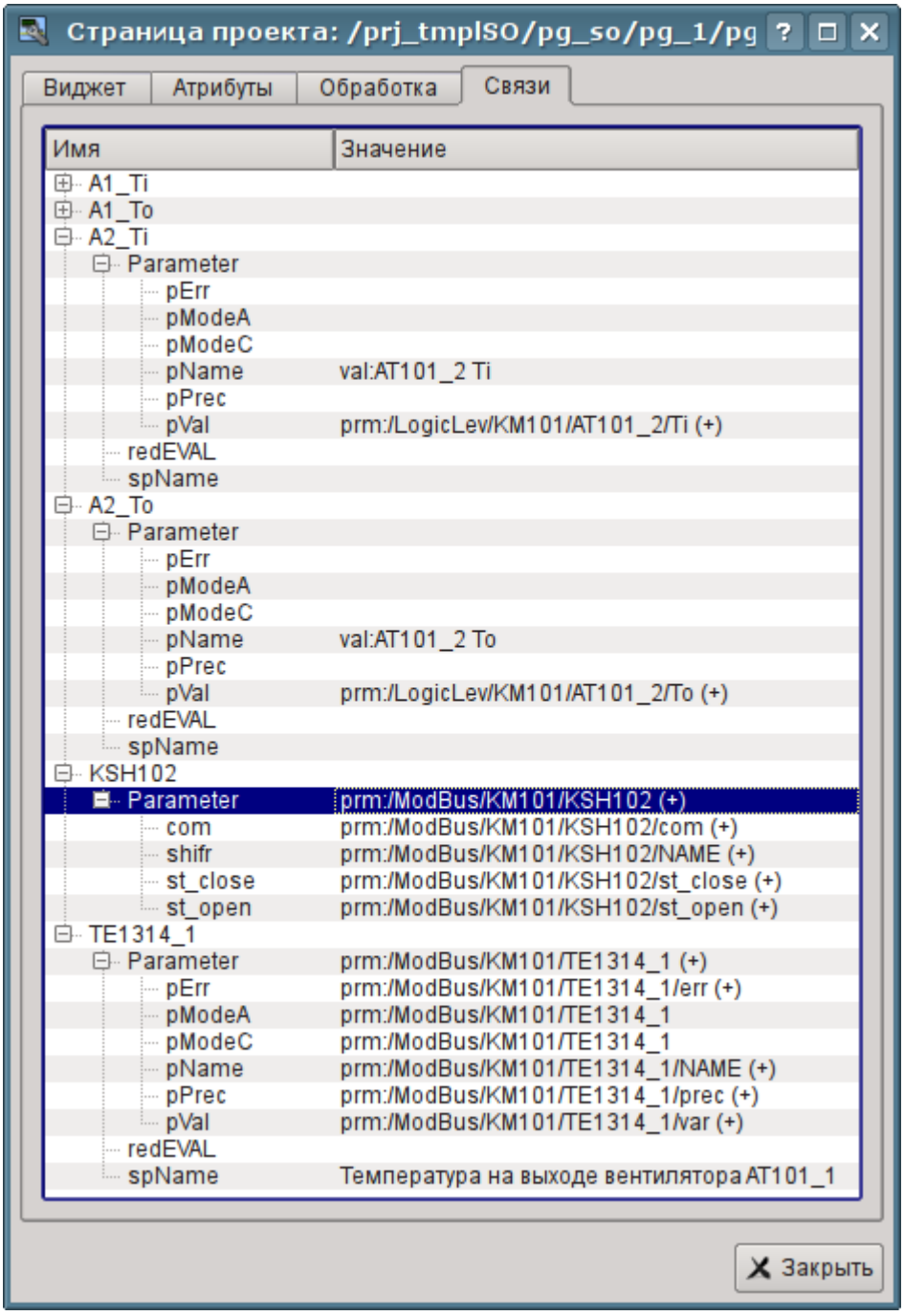

Рис. 5.2.5. Вкладка "Связи" диалога редактирования свойств мнемосхемы.

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

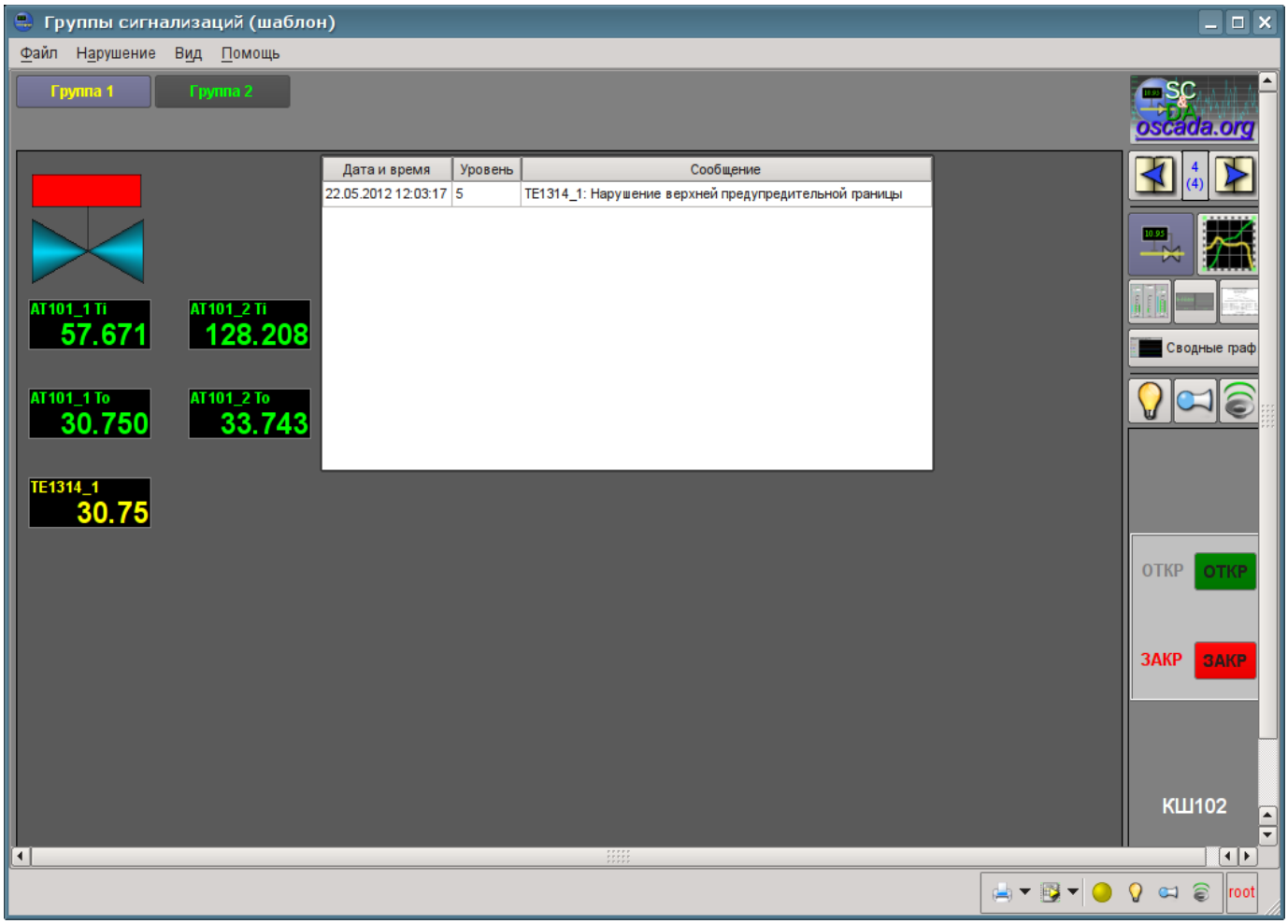

Рис. 5.2.6. Созданная мнемосхема с четырьмя подключенными сигналами, типизированными параметрами и протоколом.

Заметьте, что для типизированного параметра, с установленными границами нарушений, выход значения за границы отмечается миганием через аварийный цвет: параметра, объекта сигнализации и жёлтого кружочка снизу. При нарушении, кроме мигания, производится писк на бузер (если он доступен) и синтез речи с проговором позиции параметра (если установлена программа "ru_tts") указанной в поле связи "spName" (рис. 5.2.5). При активации нарушения справа и справа-внизу активируются кнопки с индикацией типа уведомления, а по нажатию на них осуществляется квитация соответствующего типа уведомления. По нажатию на мигающем жёлтым кружочке справа-внизу квитируются всё. Факт присутствия нарушения так-же отображается записью в протоколе, который мы добавили. Для того что-бы увидеть выход за границу нарушения Вы можете установить значение производительности вентилятора в 100 (рис.4.2.8). Детальнее про концепцию работы с нарушениями можно почитать в разделе "Рецепты".

Историю нарушений можем посмотреть в документе "Протокол нарушений", который доступен при выборе вида "Документ" (рис.5.2.7).

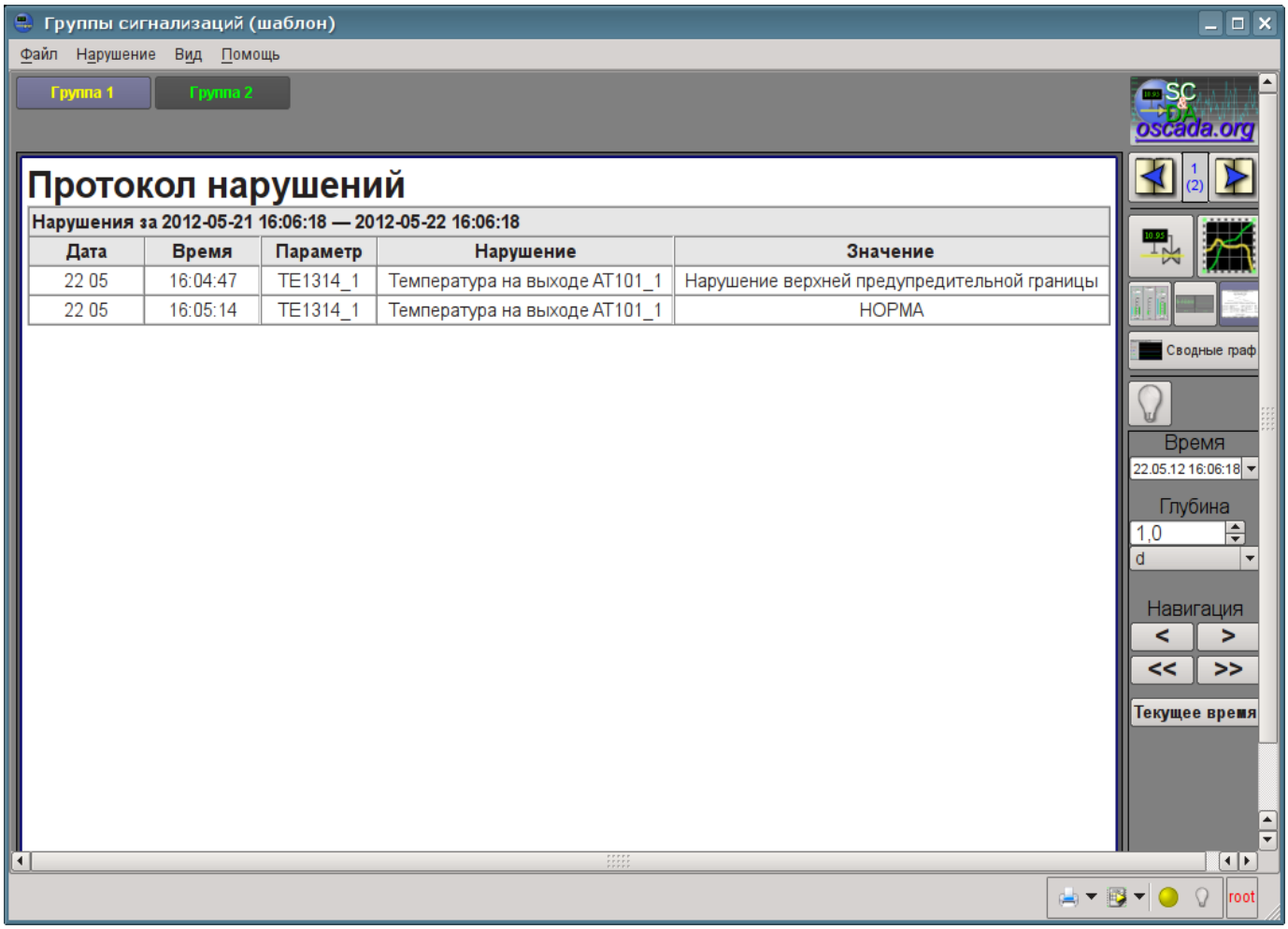

Рис. 5.2.7. Протокол нарушений.

Дискретный типизированный параметр "ModBus.KM101.KSH102", представленный нами в виде шарового крана, является активным, т.е. его можно выбрать, получив панель управления справа, как на рис.5.2.6, а так-же передавать команды (открыть или закрыть). Команды можно передавать с панели управления или через контекстное меню. Все действия оператора по управляющим воздействиям протоколируются, а документ протокола можно посмотреть при выборе вида "Документ" (рис.5.2.8).

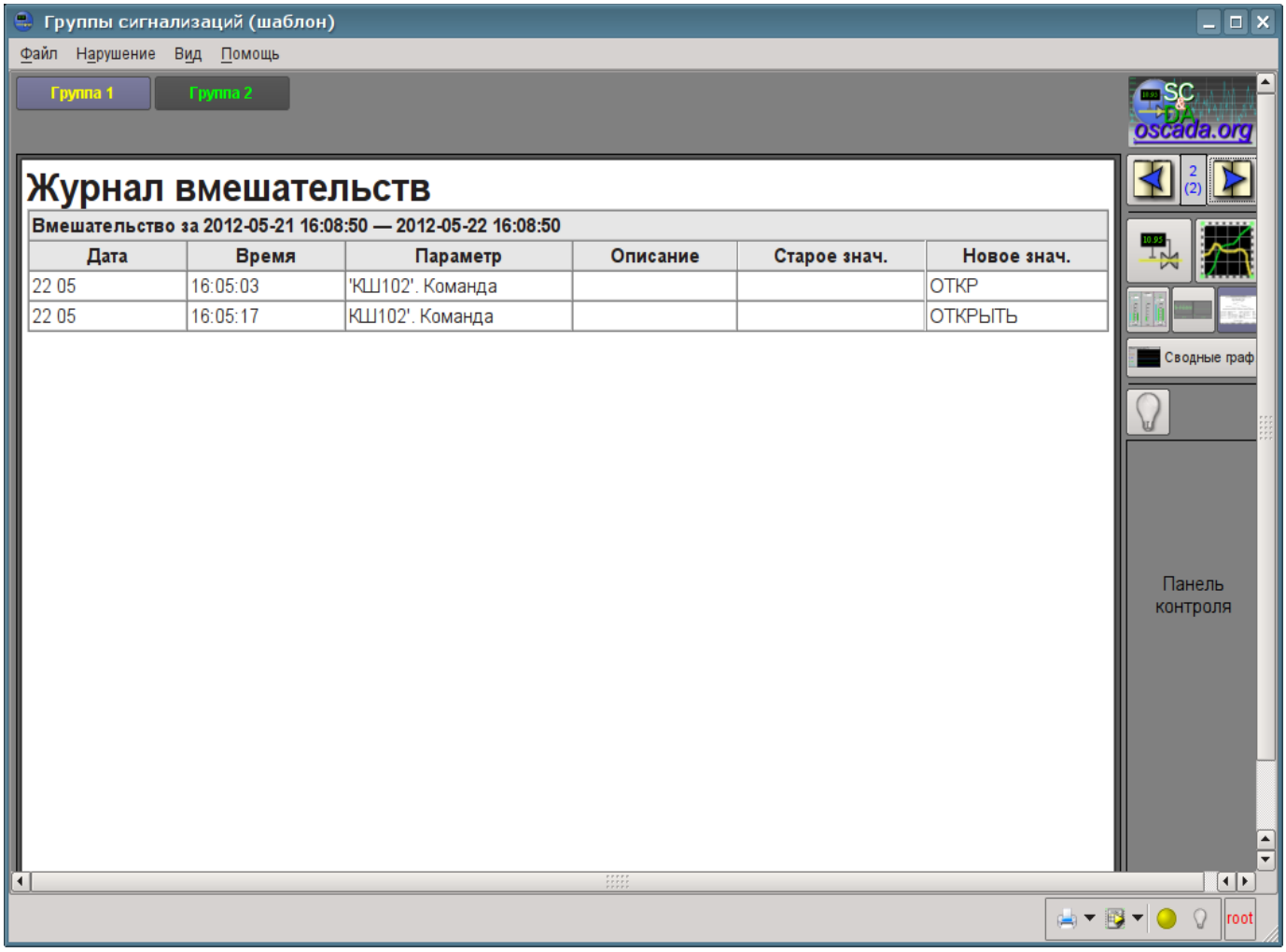

Рис. 5.2.7. Протокол действий оператора.

5.3. Создание нового комплексного элемента

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

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

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

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

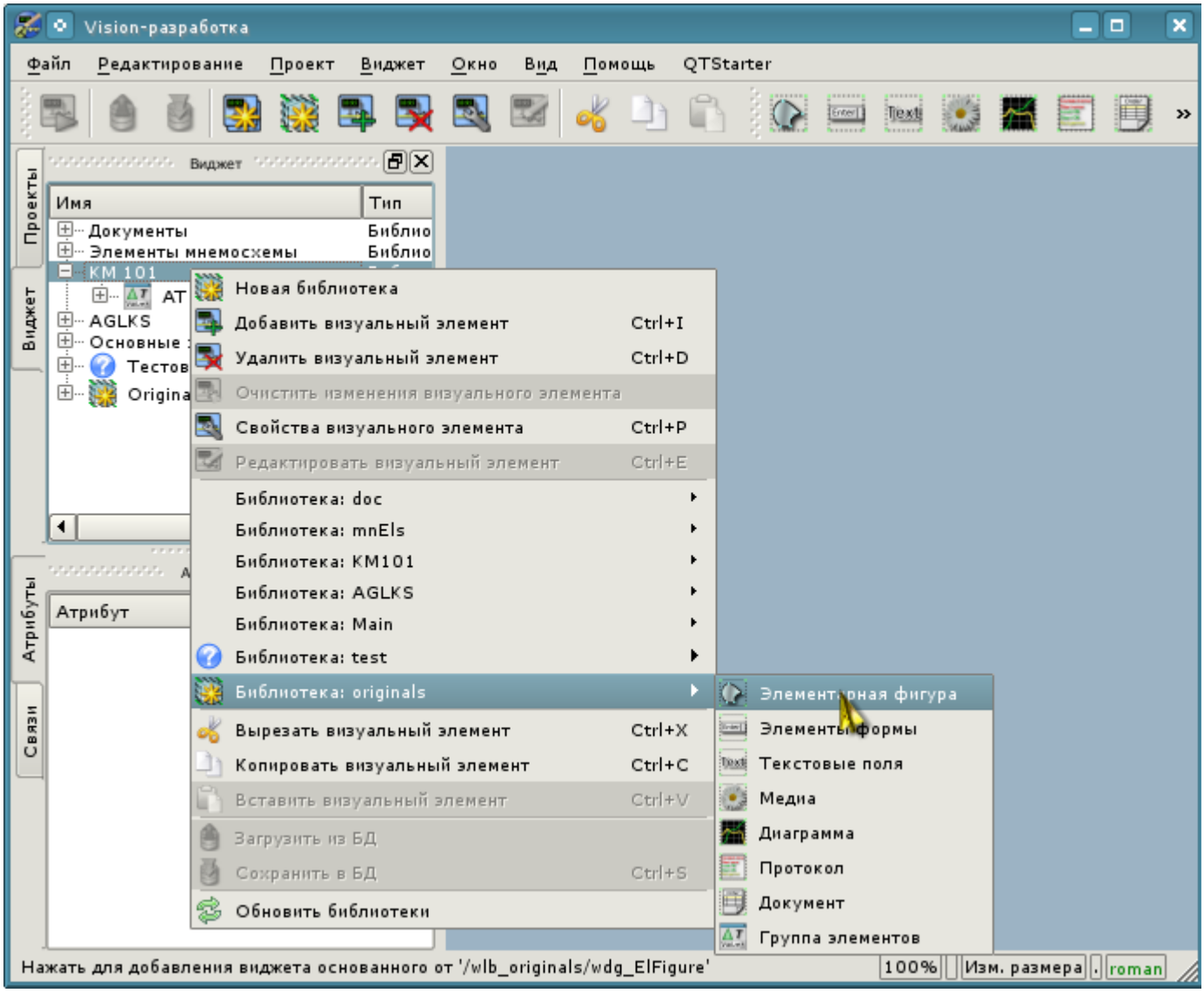

Рис. 5.3.1.1. Добавление виджета на основе примитива "Элементарная фигура" в библиотеку "KM101".

После подтверждения у нас появится объект нового виджета с именем "Воздушный холодильник". Выберем его в списке виджетов библиотеки "KM101" и откроем для редактирования посредством контекстного меню нового элемента (рис. 5.3.1.2). В инспекторе атрибутов установим свойства:

- *Геометрия:ширина* — "200".
- *Геометрия:высота* — "200".
- *Заполнение:цвет* — "lightgrey". Цвет можно задавать, как с помощью названий цветов, так и в формате #RRGGBB (#RRGGBB-AAA).

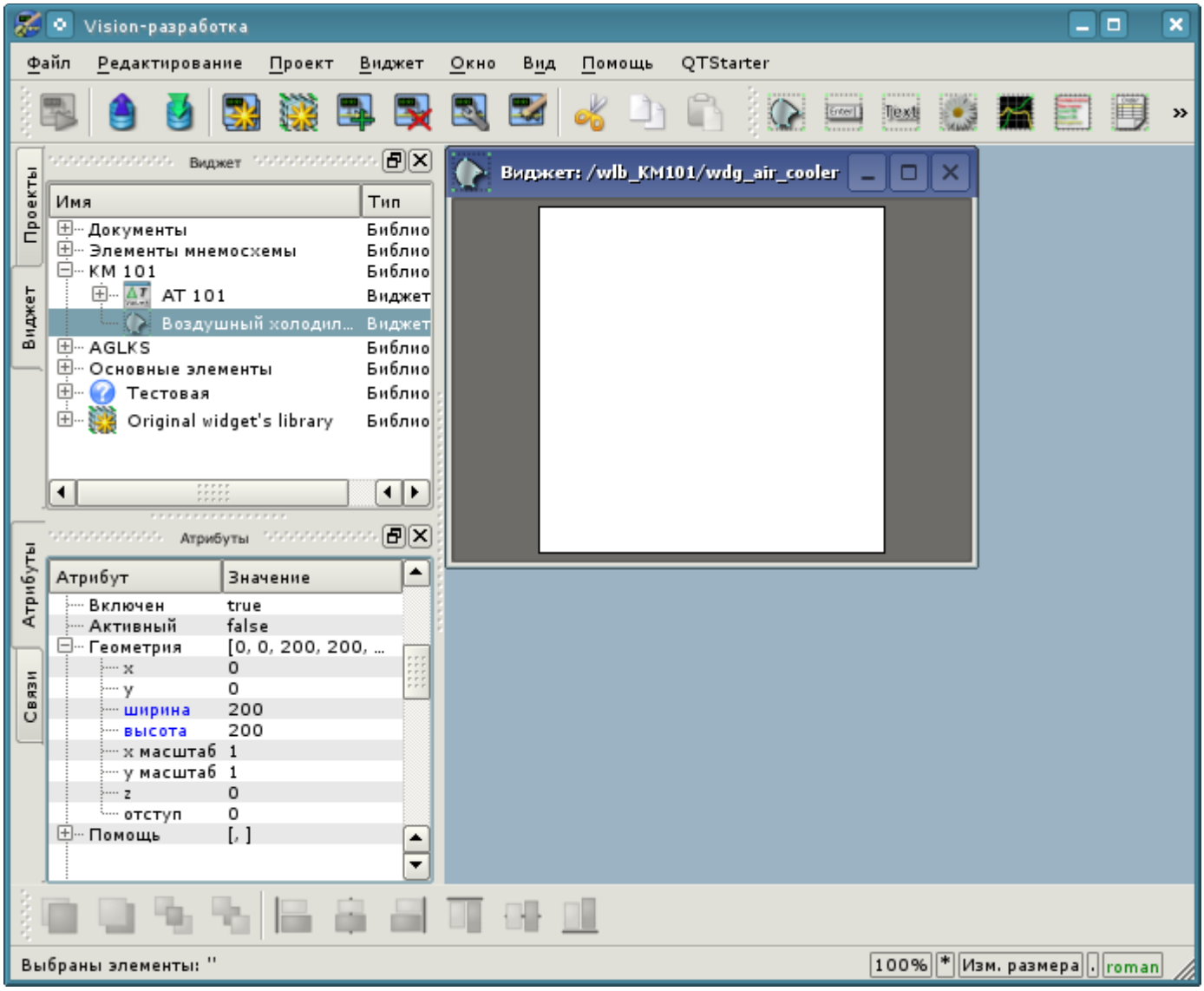

Рис. 5.3.1.2. Первичная конфигурация виджета.

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

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

Дополнительную информацию о редакторе можно прочитать здесь: http://wiki.oscada.org/Doc/Vision/ElFigure

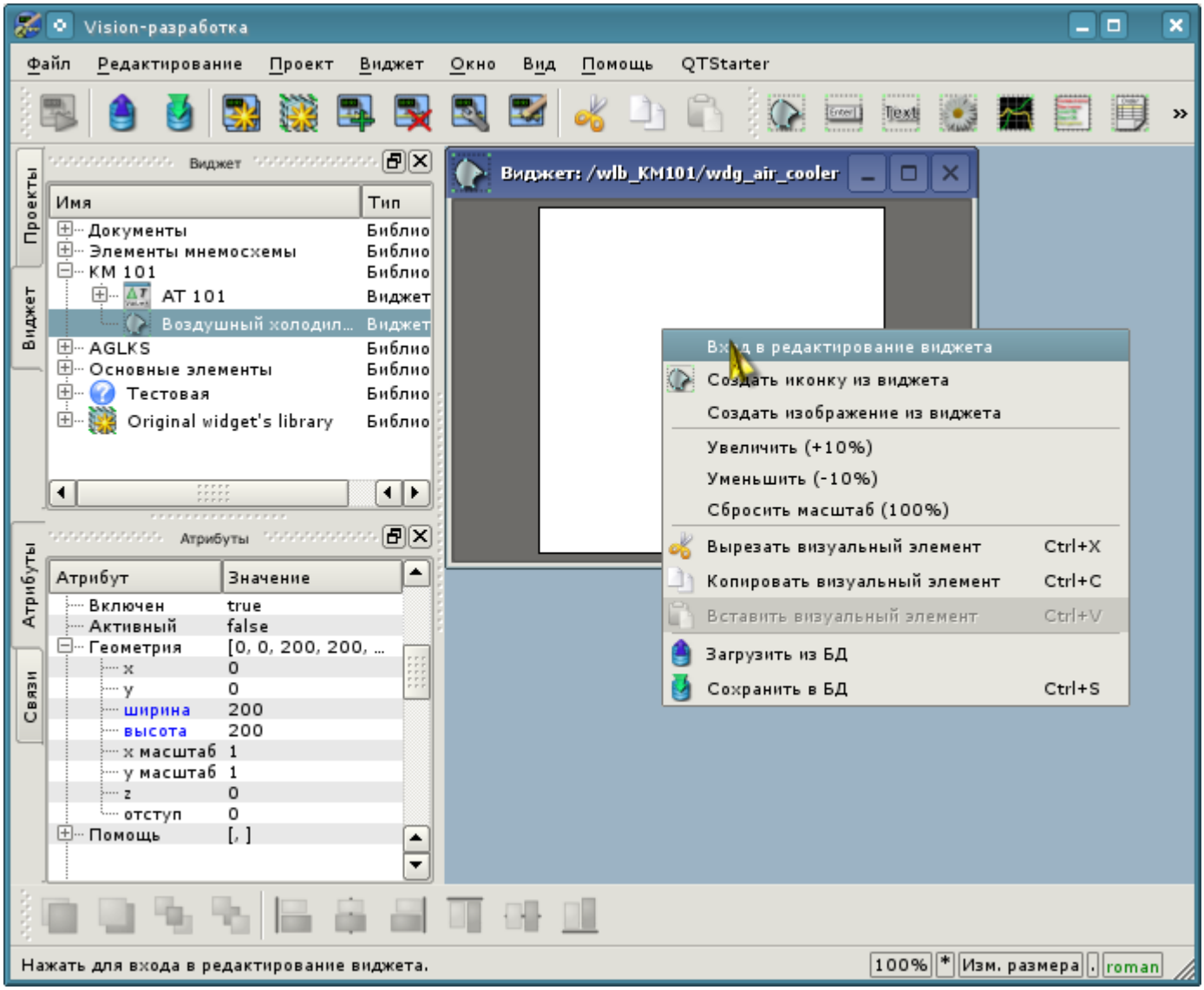

Рис. 5.3.1.3. Вход в режим рисования виджета, основанного на примитиве "Элементарная фигура".

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

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

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

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

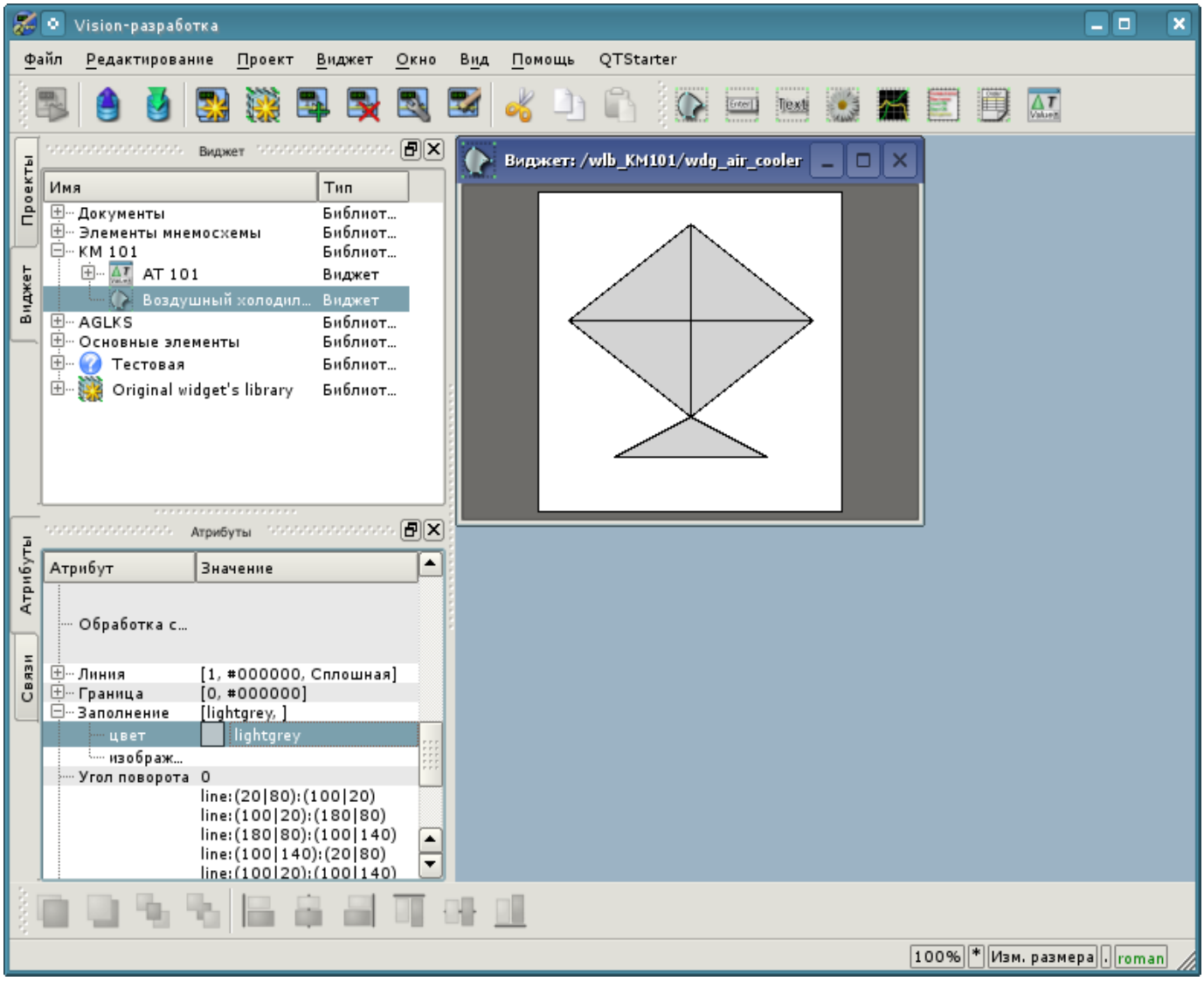

Рис. 5.3.1.4. Изображение, соответствующее "Списку элементов" виджета.

Создадим иконку для нашего виджета, которая будет видна в дереве виджетов библиотеки "KM101" (рис. 5.3.1.5).

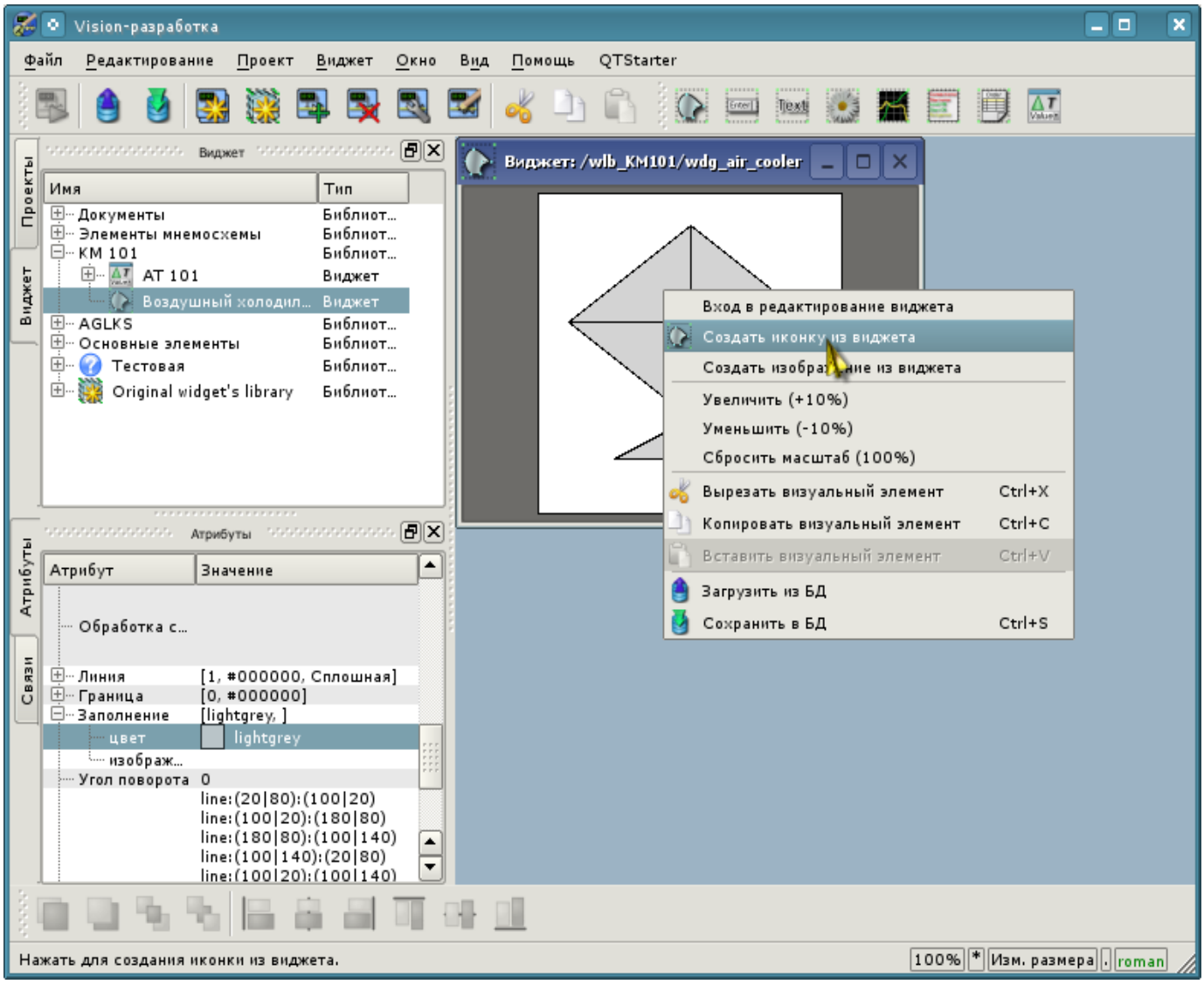

Рис. 5.3.1.5. Создание иконки для виджета.

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

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

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

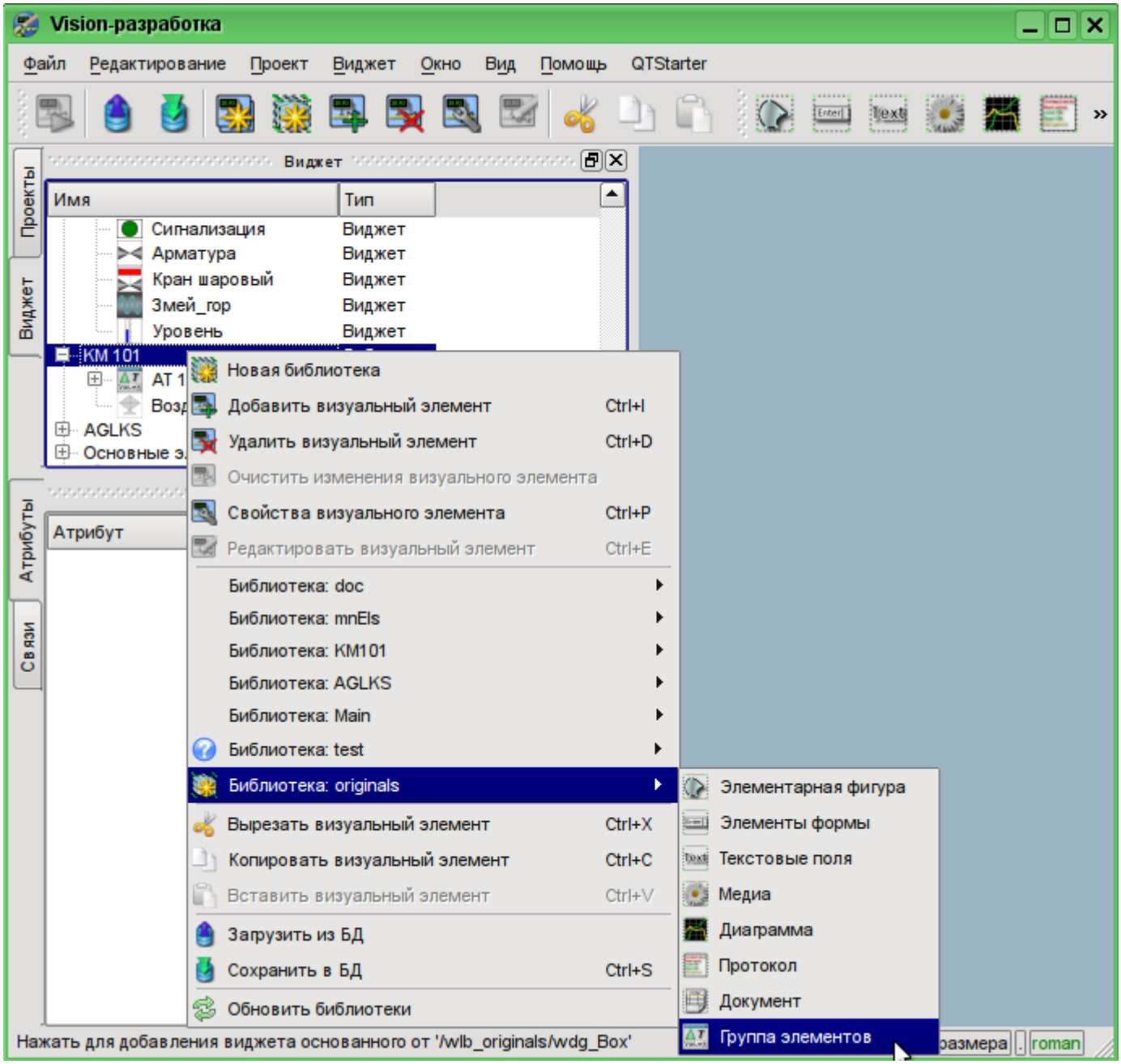

Рис. 5.3.2.1. Добавление виджета на основе примитива "Группа элементов" в библиотеку "KM 101".

После подтверждения у нас появится объект нового виджета с именем "Холодильник". Выберем его в списке виджетов библиотеки "KM 101" и откроем для редактирования. В инспекторе атрибутов установим свойства:

- *Геометрия:ширина* — "250".
- *Геометрия:высота* — "200".

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

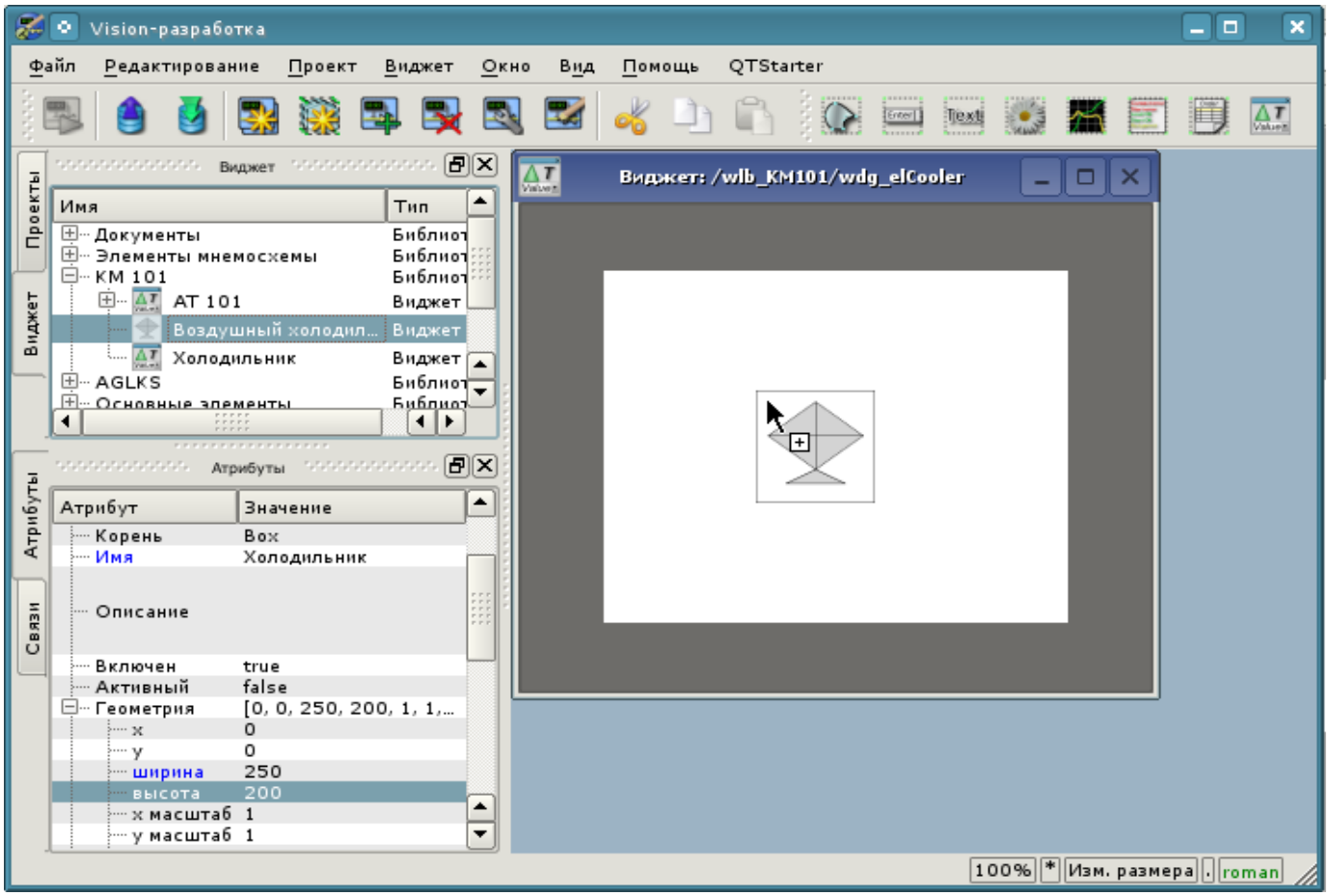

Рис. 5.3.2.2. Перетаскивание (Drag&Drop) виджета "air_cooler" в виджет-контейнер "elCooler".

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

- *Геометрия:x* — "25".
- *Геометрия:y* — "0".

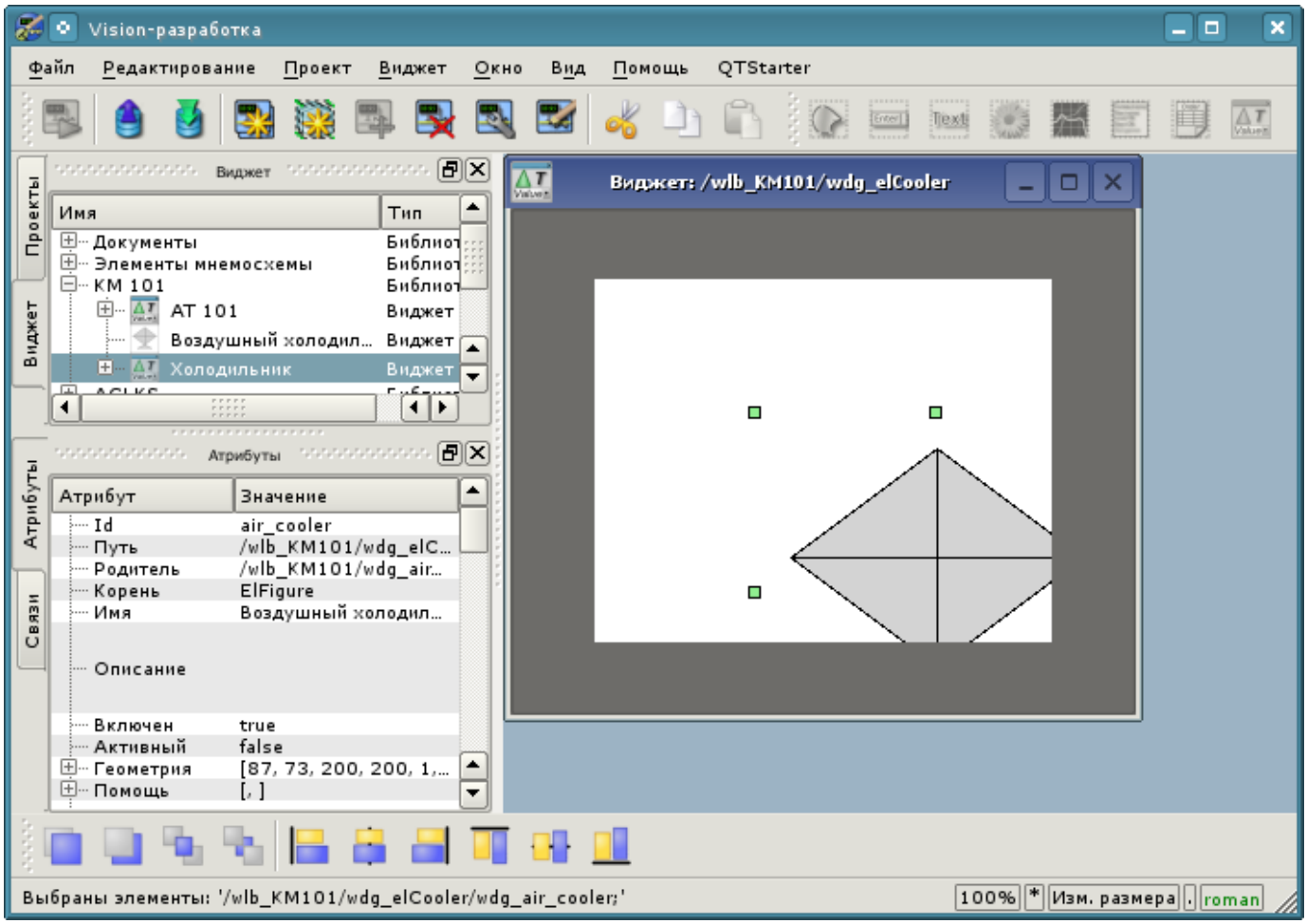

Рис. 5.3.2.3. Добавление унаследованного виджета "air_cooler".

Далее развернем библиотеку "Элементы мнемосхемы", найдем там элемент "Вентилятор" (cooler2) и перетащим его на виджет-контейнер. Этот элемент будет динамически отображать интенсивность работы воздушного холодильника. В результате появится окно диалога для ввода идентификатора и имени нового виджета. Введём идентификатор "cooler2", а имя снова оставим пустым. Таким образом, вновь созданный виджет внутри контейнера "elCooler" унаследует элемент библиотеки "Элементы мнемосхемы" - "Вентилятор" (cooler2). После подтверждения ввода идентификатора и имени виджет "Вентилятор" (cooler2) добавится в наш виджет-контейнер "elCooler". В инспекторе атрибутов установим свойства:

- *Геометрия:x* — "75".
- *Геометрия:y* — "30".
- *Геометрия:z* — "10". Поднять элемент над всеми, можно из панели "Функции видимости виджетов".
- *Цвет1* — "#FFFF00-200", добавили альфа канал, прозрачность 200 ("0" - полностью прозрачный, а "255" - полностью непрозрачный), рис. 5.3.2.4.
- *Цвет2* — "#FF0000-200", добавили альфа канал, прозрачность 200.

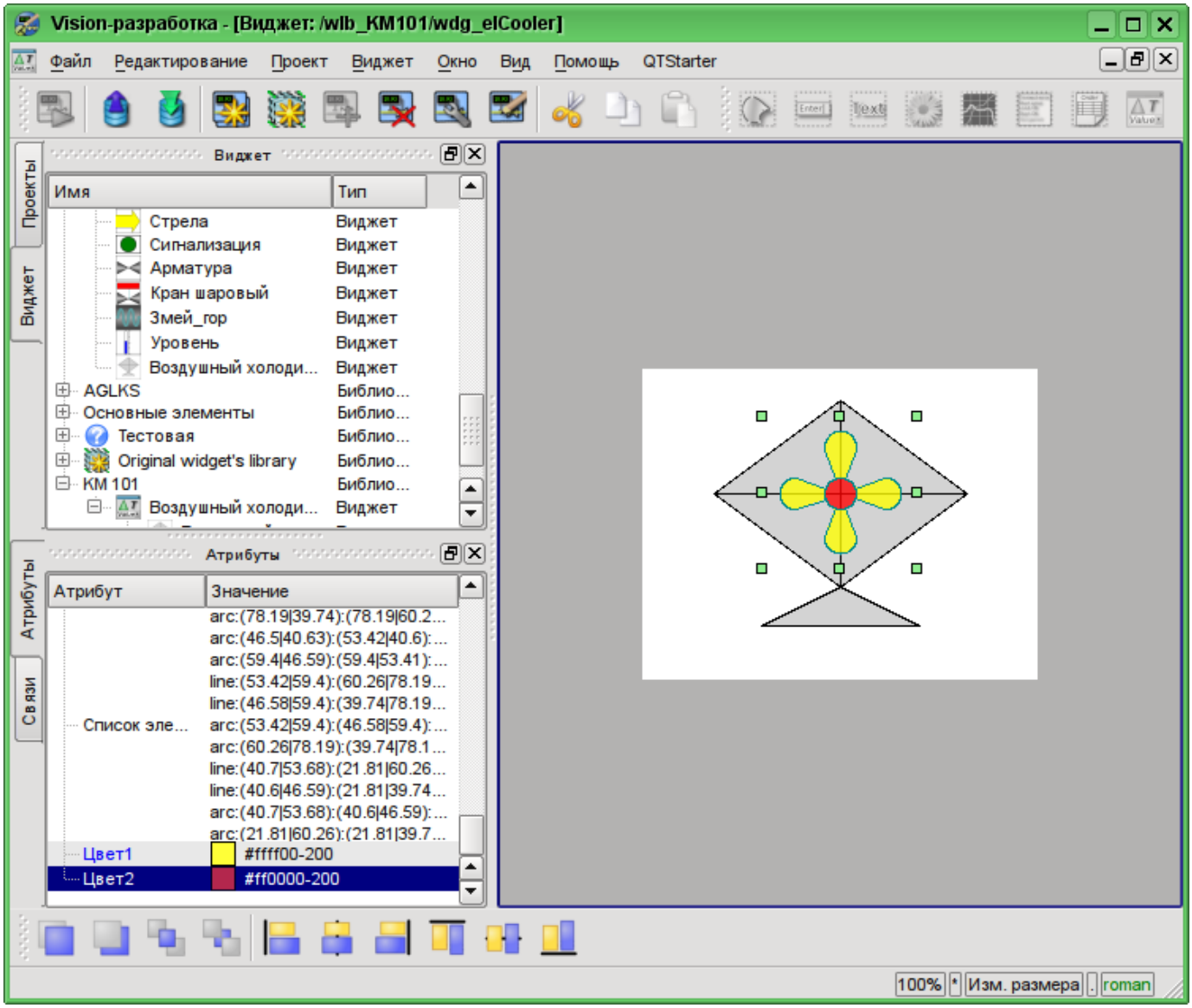

Рис. 5.3.2.4. Изменение прозрачности цветов заполнений у унаследованного виджета "cooler2".

Теперь добавим в виджет-контейнер "elCooler" два текстовых поля, основанных на примитиве "Текст", с целью отображения входной и выходной температур потока. Для этого в библиотеке "KM 101" выделим виджет "холодильник" и нажмем на панели визуальных элементов на иконку примитива "Текст", как это показано на рис 5.3.2.5. В результате появится диалог ввода идентификатора и имени вновь создаваемого элемента. Введем идентификатор "Ti" для первого текстового поля, а имя оставим пустым. В инспекторе атрибутов установим свойства:

- *Геометрия:x* — "5".
- *Геометрия:y* — "20".
- *Геометрия:ширина* — "70".
- *Геометрия:высота* — "35".
- *Выравнивание* — "В центре".
- *Шрифт* — "Arial 14 1". Изменение шрифта можно осуществлять в диалоге, который открывается по нажатию на ключик в поле редактирования (рис. 5.3.2.7).
- *Текст* — "%1{Enter}град.С" (рис. 5.3.2.8). {Enter} — переход на другую строку.
- *Количество аргументов* — "1" (рис. 5.3.2.9):
    - *Аргумент 0:тип* — "Вещественный".
    - *Аргумент 0:значение* — "300.25", задано лишь только с целью наглядности, в режиме исполнения оно будет подменяться реальным значением входной температуры.
    - *Аргумент 0:конфигурация* — "3;f;2".

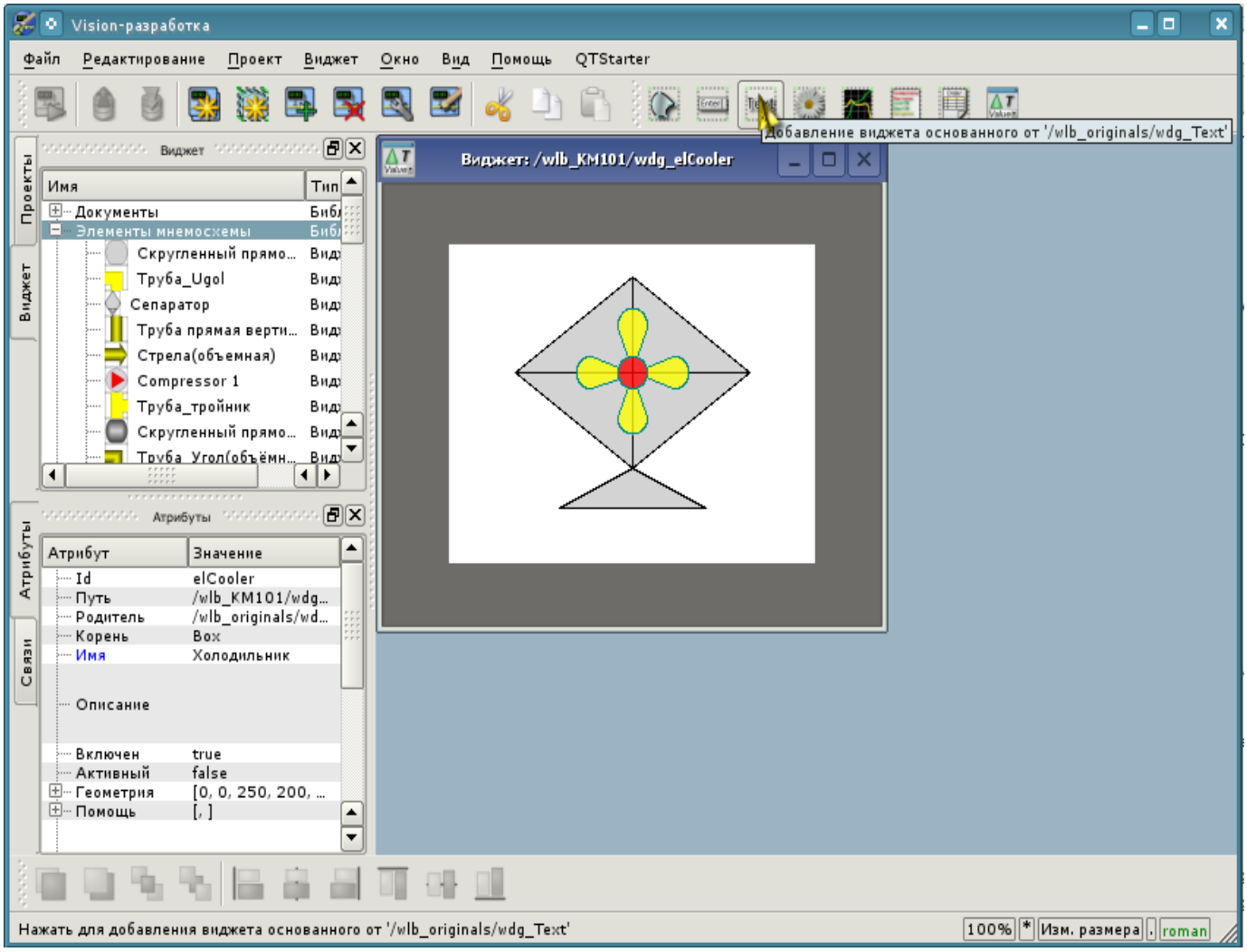

Рис. 5.3.2.5. Добавление в контейнер нового элемента, основанного на примитиве "Текст".

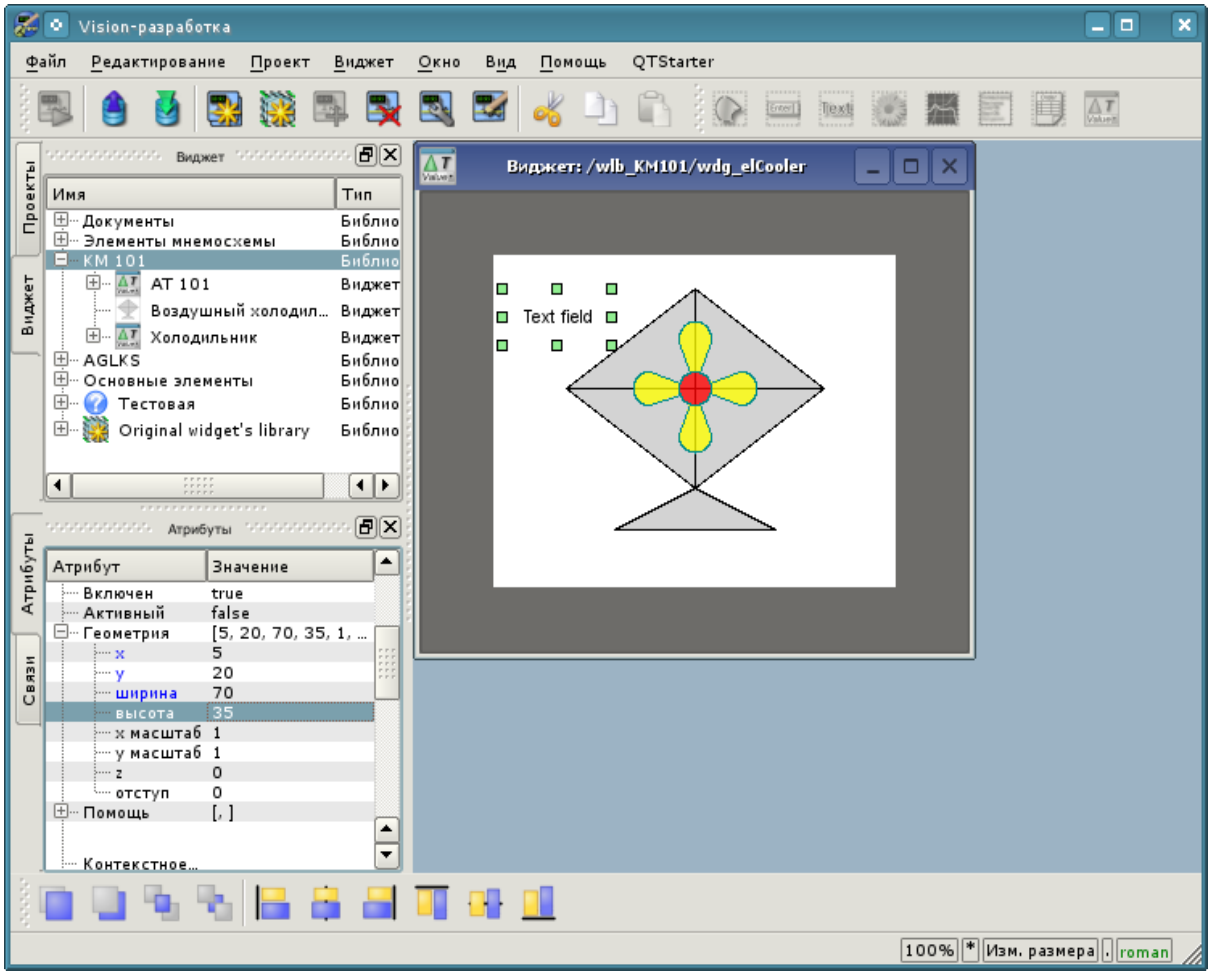

Рис. 5.3.2.6. Задание геометрии виджета "Ti".

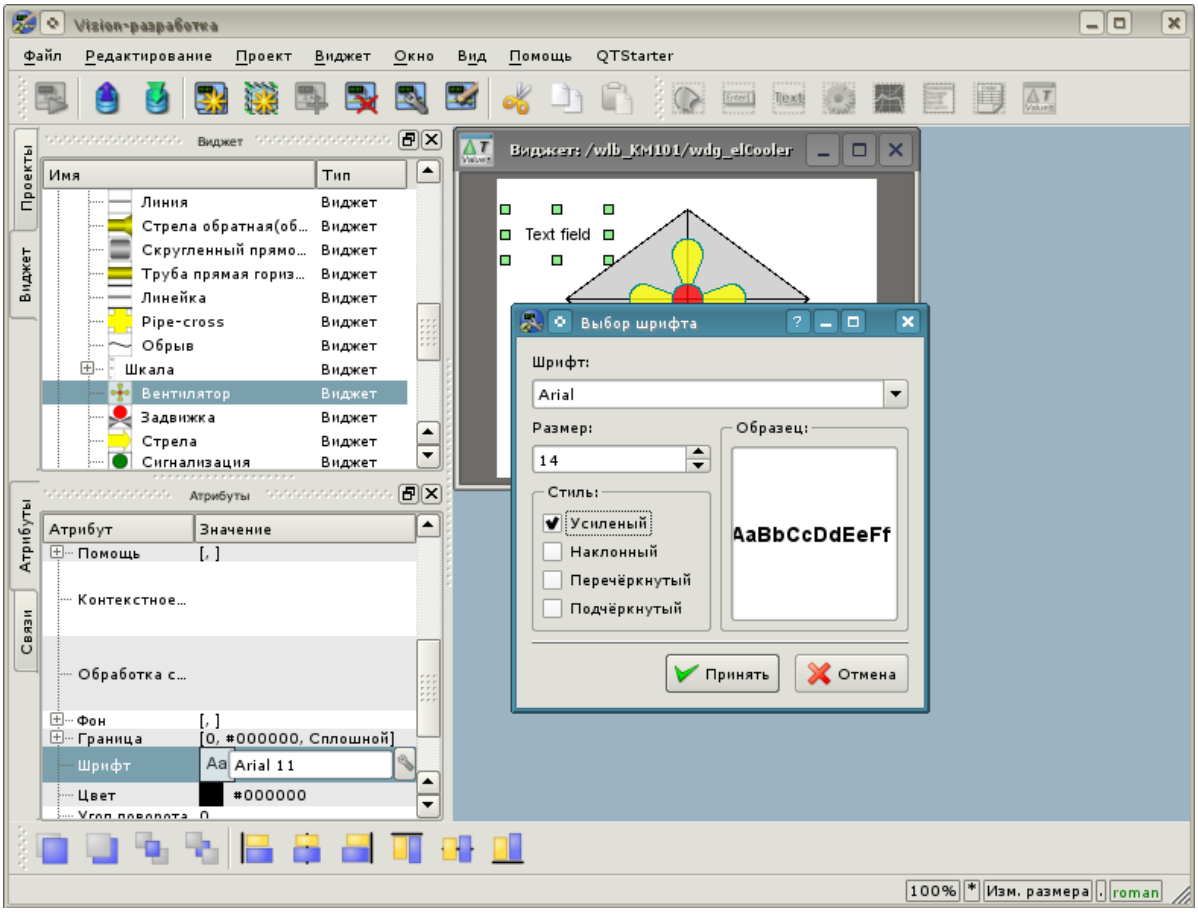

Рис. 5.3.2.7. Изменение размера шрифта для виджета "Ti".

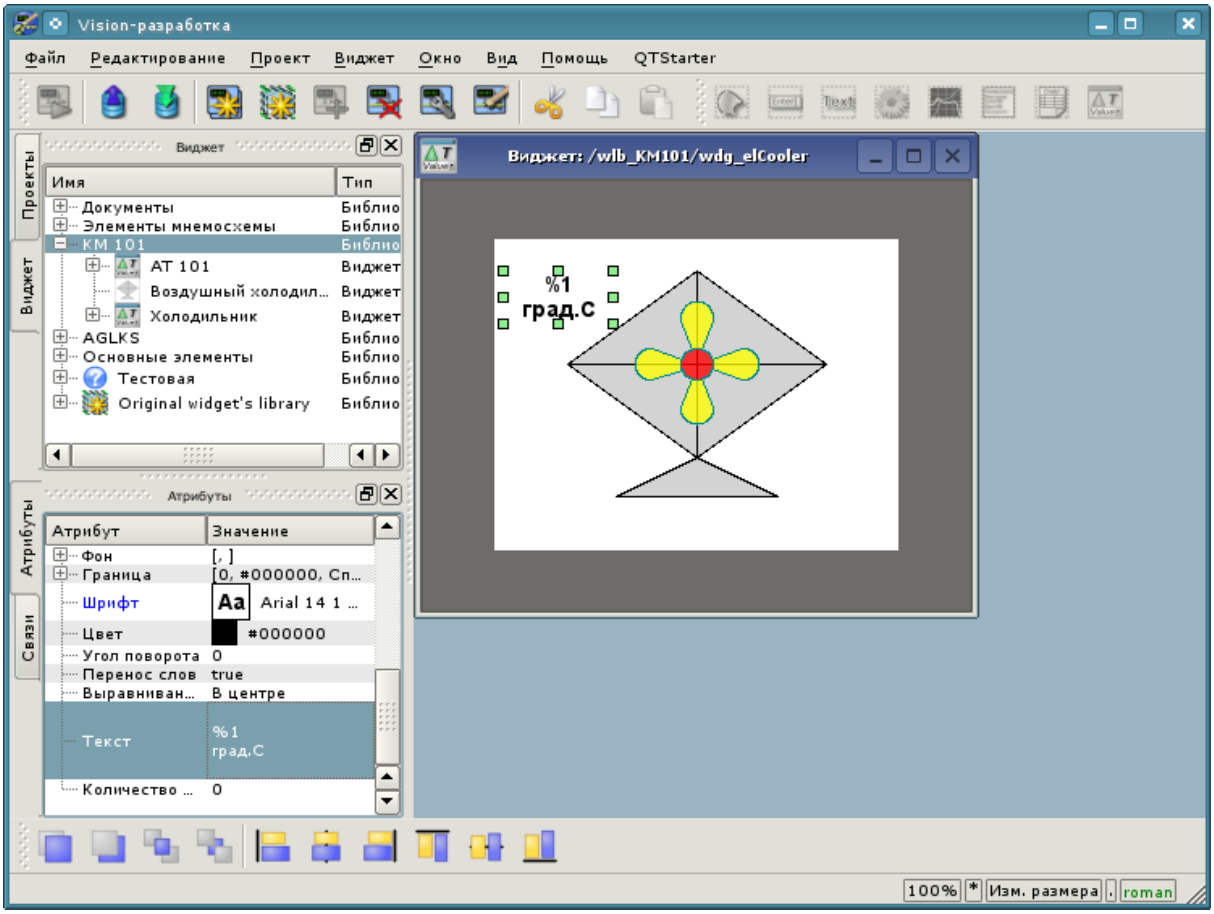

Рис. 5.3.2.8. Изменение поля "Текст" и указание в нем наличия атрибута для виджета "Ti".

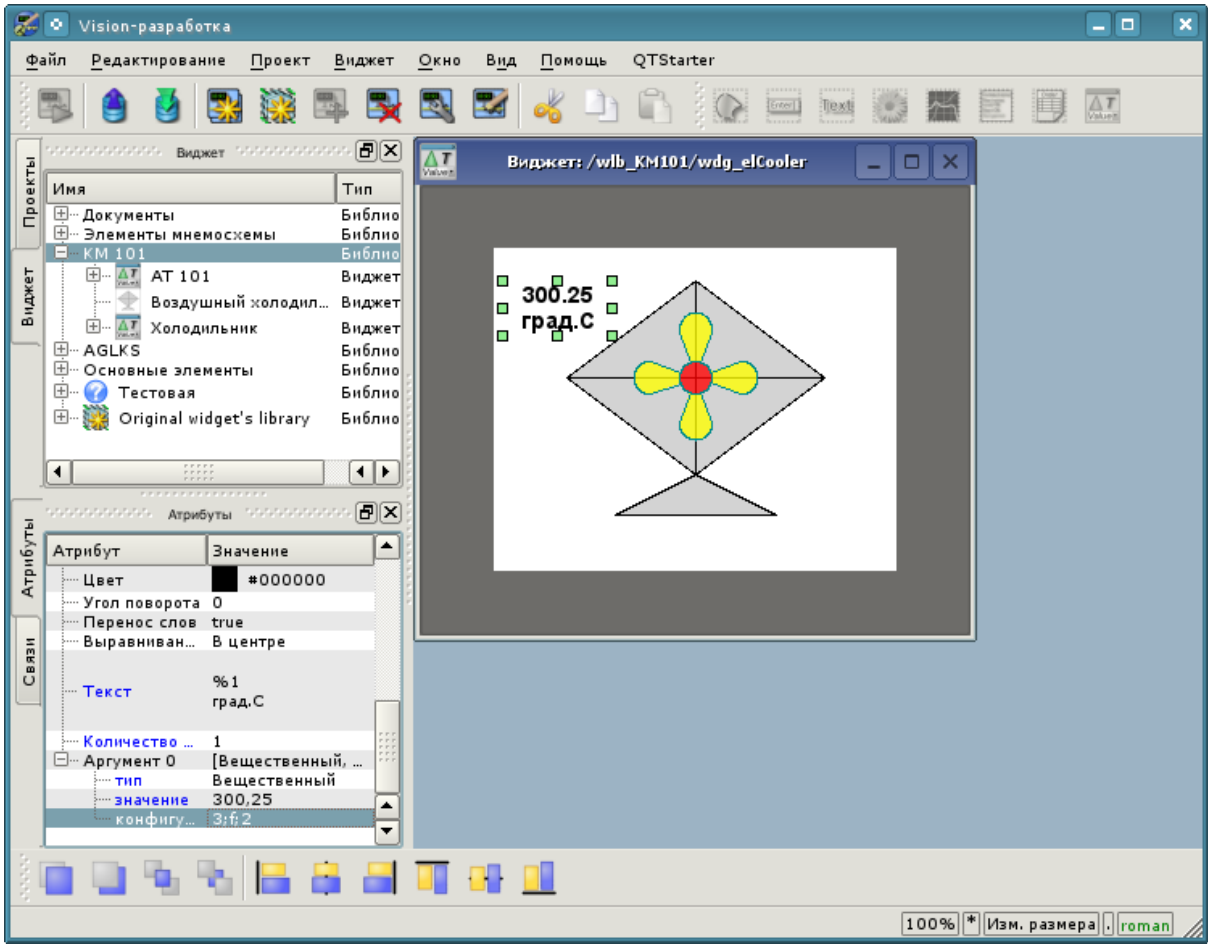

Рис. 5.3.2.9. Конфигурация аргумента для виджета "Ti".

Теперь скопируем виджет "Ti" с целью создания аналогичного ему виджета "To" (выходная температура). Вставим скопированный виджет, в диалоге ввода идентификатора и имени для вновь созданного виджета в поле "ID" укажем "To", а имя оставим пустым (рис. 5.3.2.10). В инспекторе атрибутов установим свойства:

- *Геометрия:x* — "175".
- *Геометрия:y* — "20".

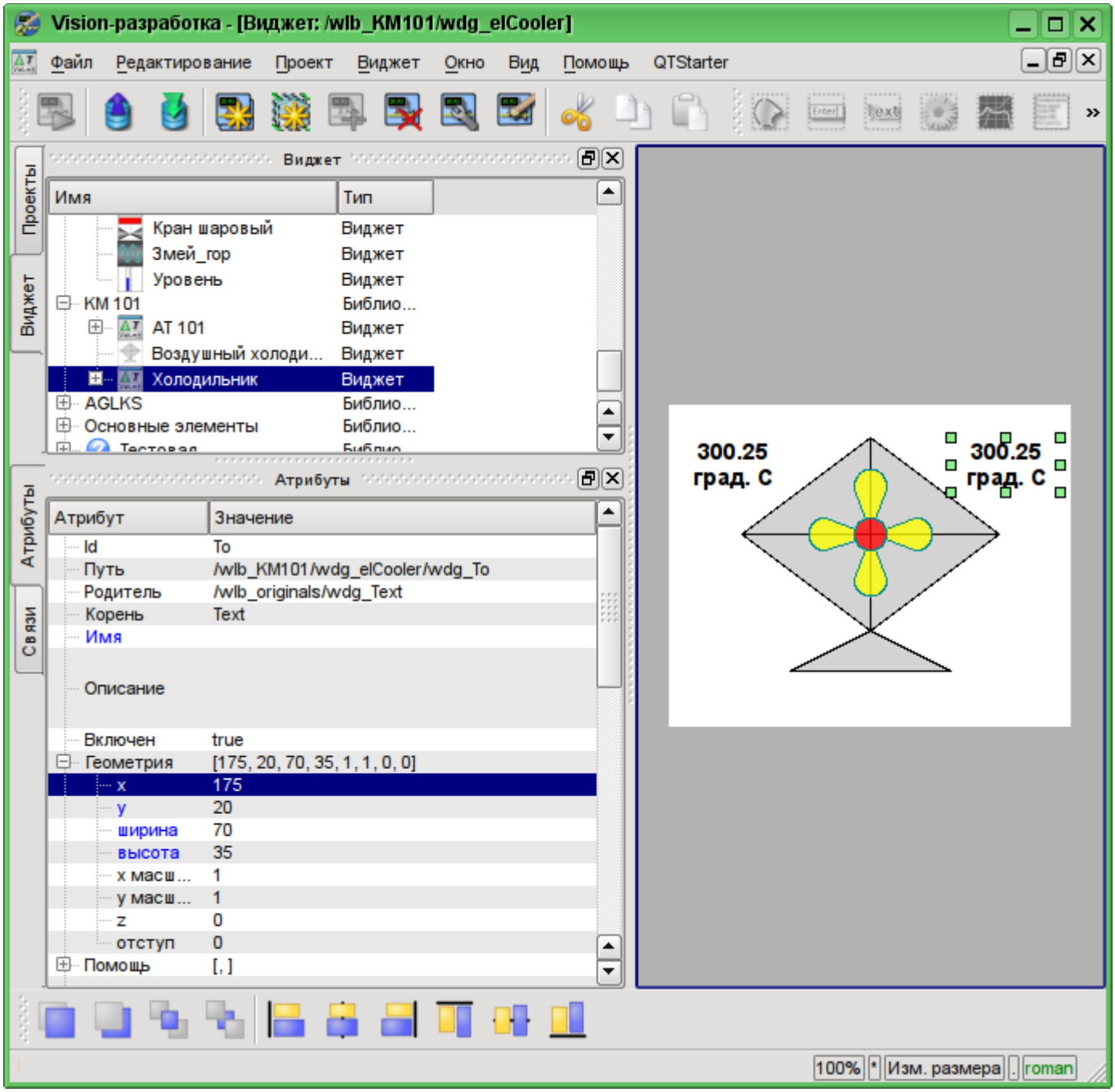

Рис. 5.3.2.10. Виджет "To".

Теперь добавим виджет, основанный на примитиве "Элементы формы" (рис. 5.3.2.11), который будем использовать в качестве ComboBox для выбора заданий производительности холодильника. В качестве идентификатора укажем "cw", и поле "Имя" оставим пустым (рис. 5.3.2.12). В инспекторе атрибутов установим свойства:

- *Активный* — "true".
- *Геометрия:x* — "60".
- *Геометрия:y* — "158".
- *Геометрия:z* — "10". Поднять элемент над всеми, можно из панели "Функции видимости виджетов".
- *Геометрия:ширина* — "60".
- *Геометрия:высота* — "40".
- *Тип элемента* — "Combo Box".
- *Шрифт* — "Arial 14 1".
- *Значение* — "200".
- *Конфигурация* — "0{Enter}50{Enter}100{Enter}150{Enter}200". {Enter} — переход на другую строку.

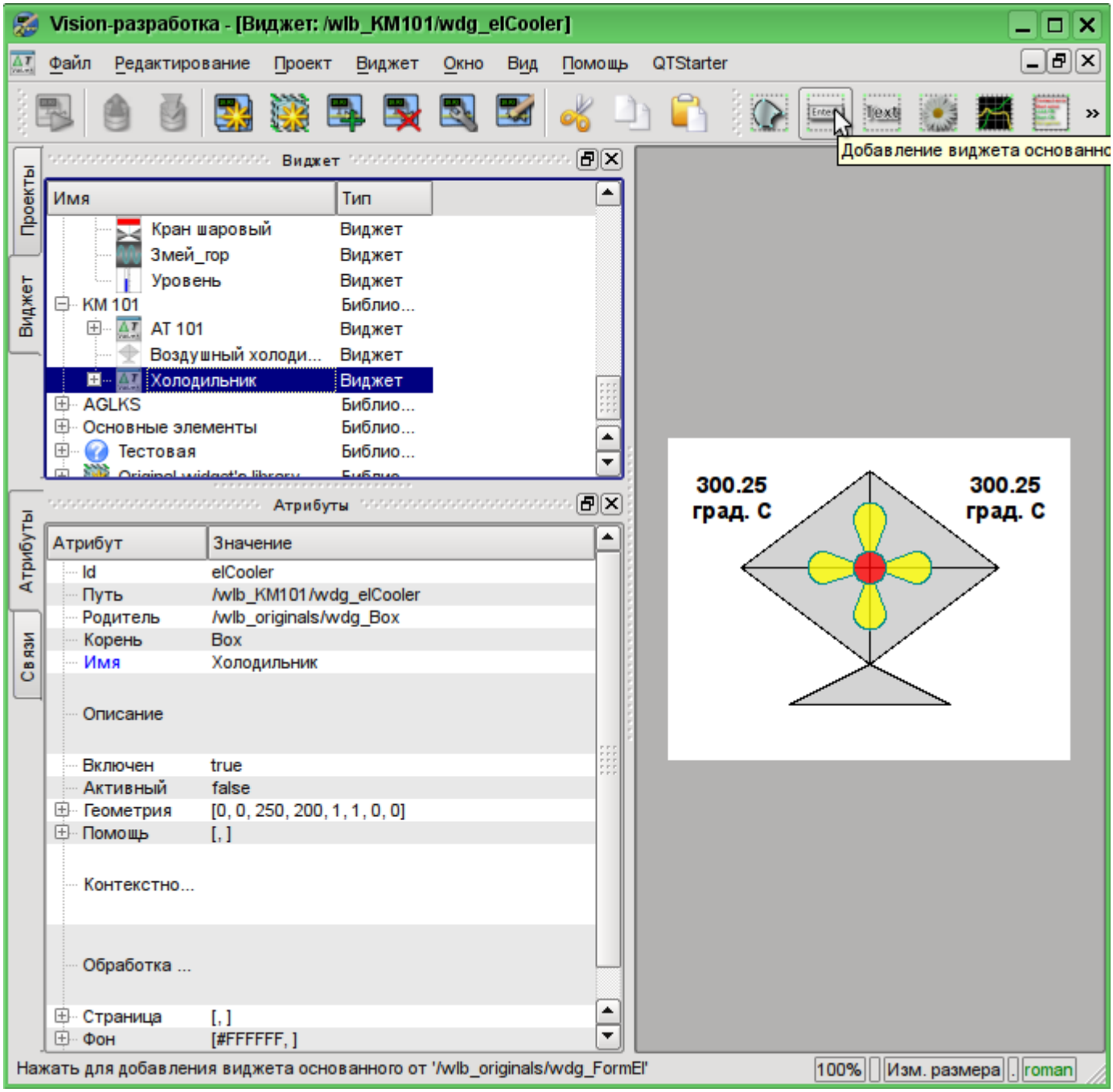

Рис. 5.3.2.11. Добавление виджета, основанного на примитиве "Элементы формы".

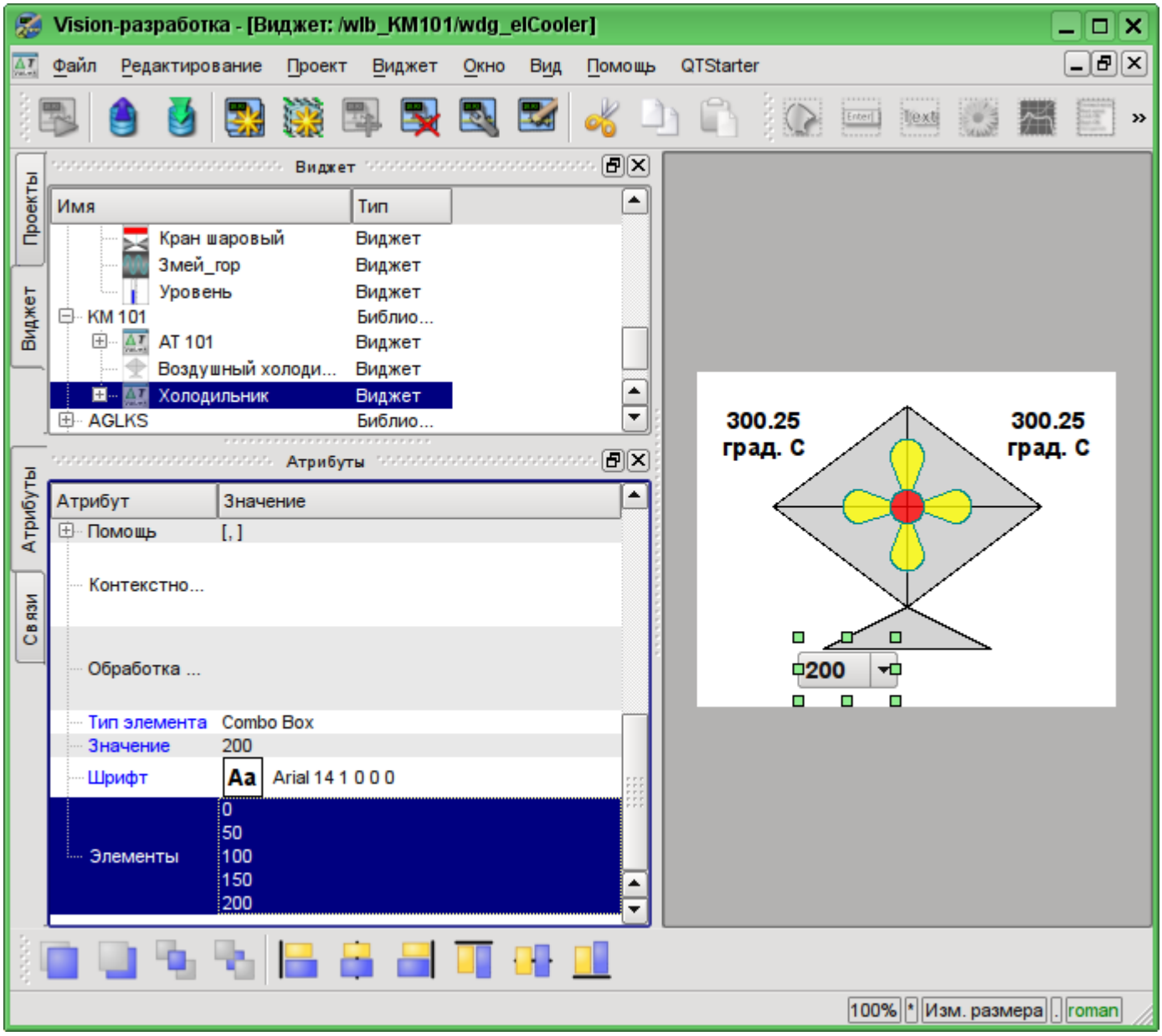

Рис. 5.3.2.12. Заполнение параметров ComboBox "cw".

Для отображения единицы измерения производительности холодильника добавим виджет на основе примитива "Текст". Делаем ту же процедуру, что и для виджета "Ti". Идентификатором вновь созданного виджета сделаем "dimension" (рис. 5.3.2.13). В инспекторе атрибутов установим свойства:

- *Геометрия:x* — "125".
- *Геометрия:y* — "168".
- *Геометрия:ширина* — "60".
- *Геометрия:высота* — "20".
- *Выравнивание* — "В центре".
- *Шрифт* — "Arial 14 1".
- *Текст* — "об./мин.".

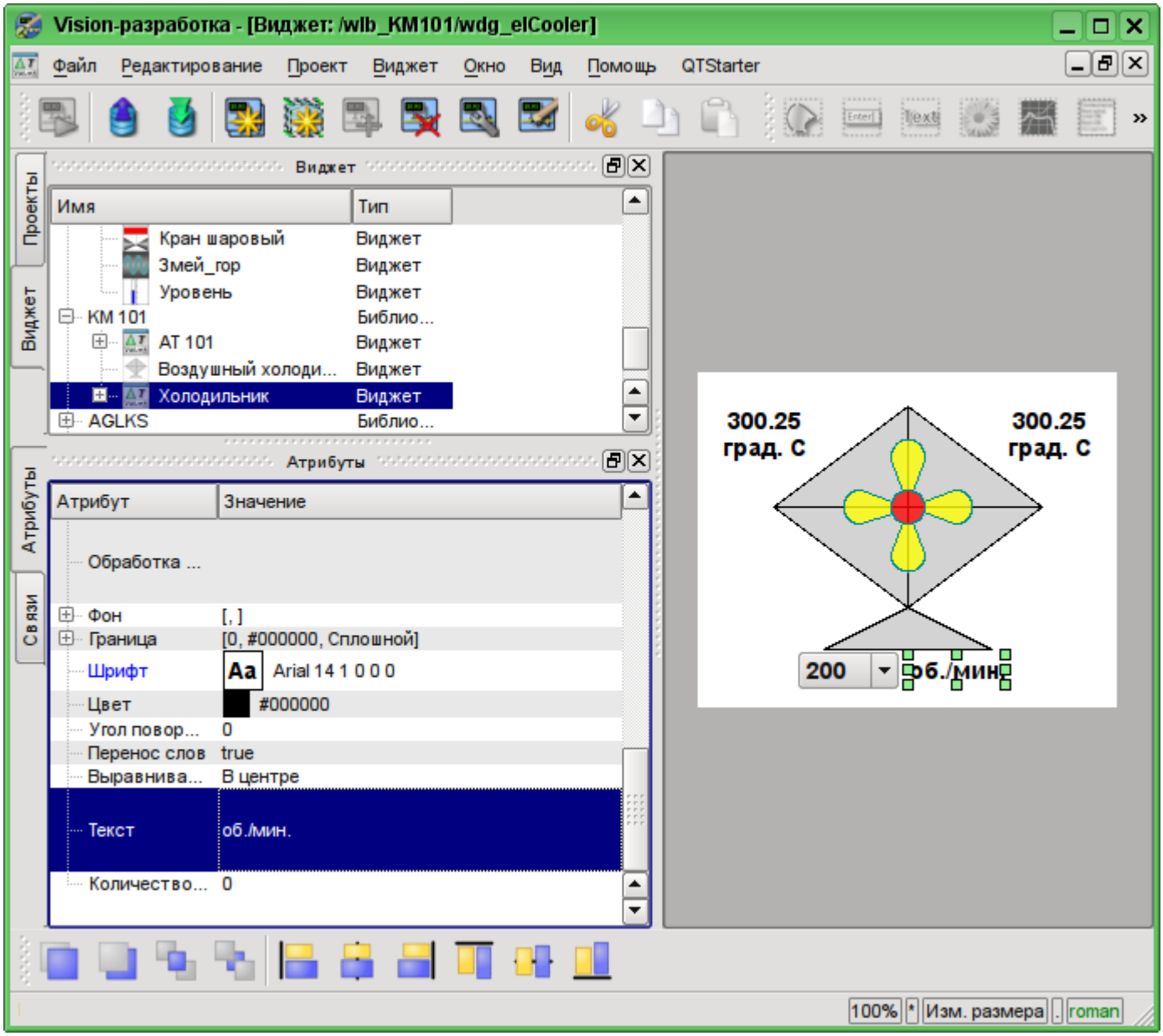

Рис. 5.3.2.13. Добавление виджета "dimension", основанного на примитиве "Текст" и изменение его параметров.

Для добавления логики обработки виджета "Холодильник"(elCooler) откроем диалог редактирования свойств этого визуального элемента и перейдём на вкладку "Обработка". На этой вкладке мы увидим дерево атрибутов виджета и поле текста программы, для обработки атрибутов. Для решения нашей задачи нужно добавить три атрибута: Ti, To, Cw (рис. 5.3.2.14). Для добавления атрибута нужно развернуть корневой элемент ".", выбрать любой элемент внутри корневого и нажать кнопку "Добавить атрибут" снизу.

Далее включим в обработку атрибут "value" комбобокса "cw", как это показано на рис. 5.3.2.15. Аналогично включим в обработку атрибут "arg0val" для Ti и To, а также атрибут "speed" у элемента "cooler2".

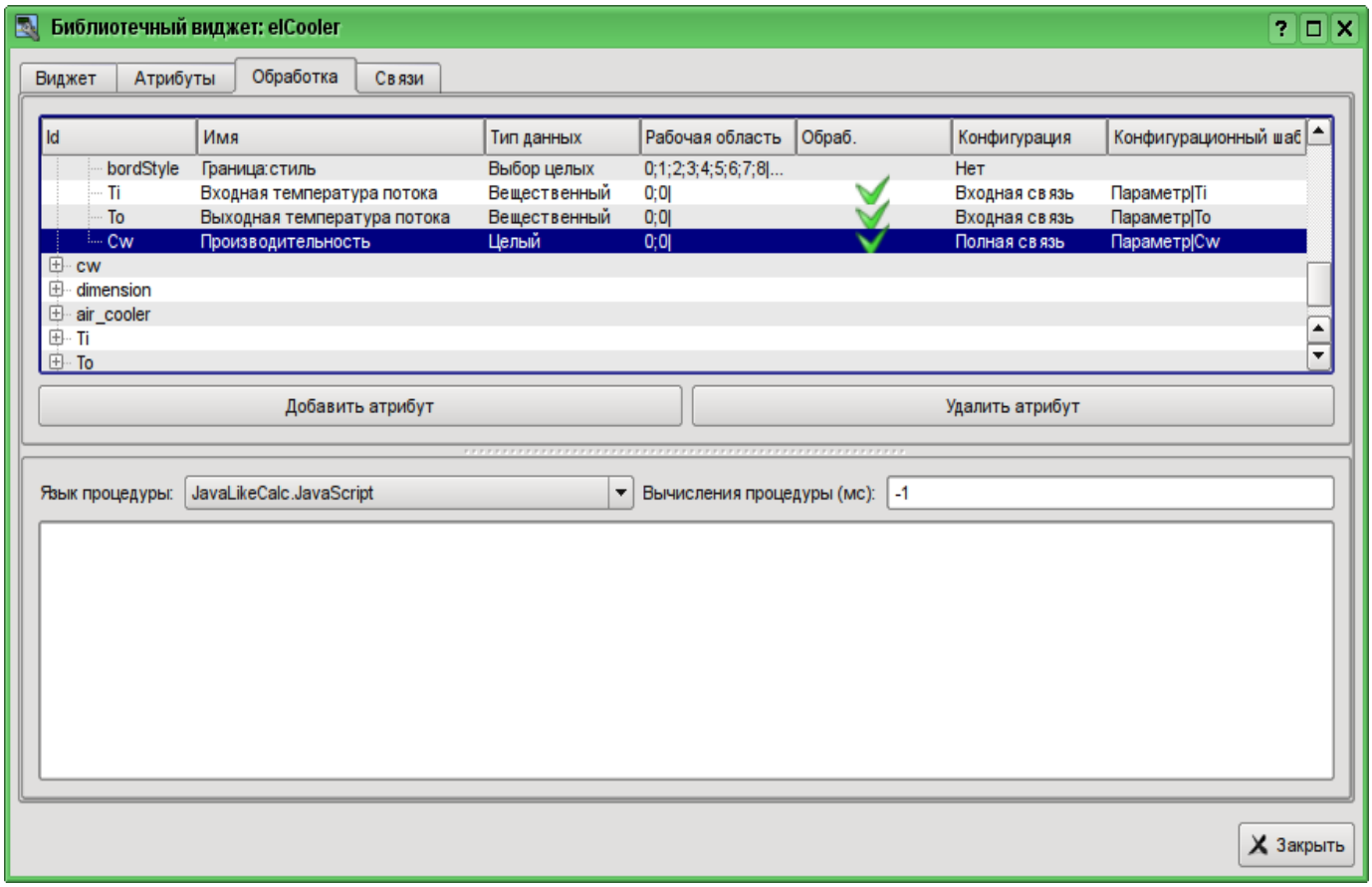

Рис. 5.3.2.14. Добавление трех атрибутов для элемента "elCooler" библиотеки "КМ 101".

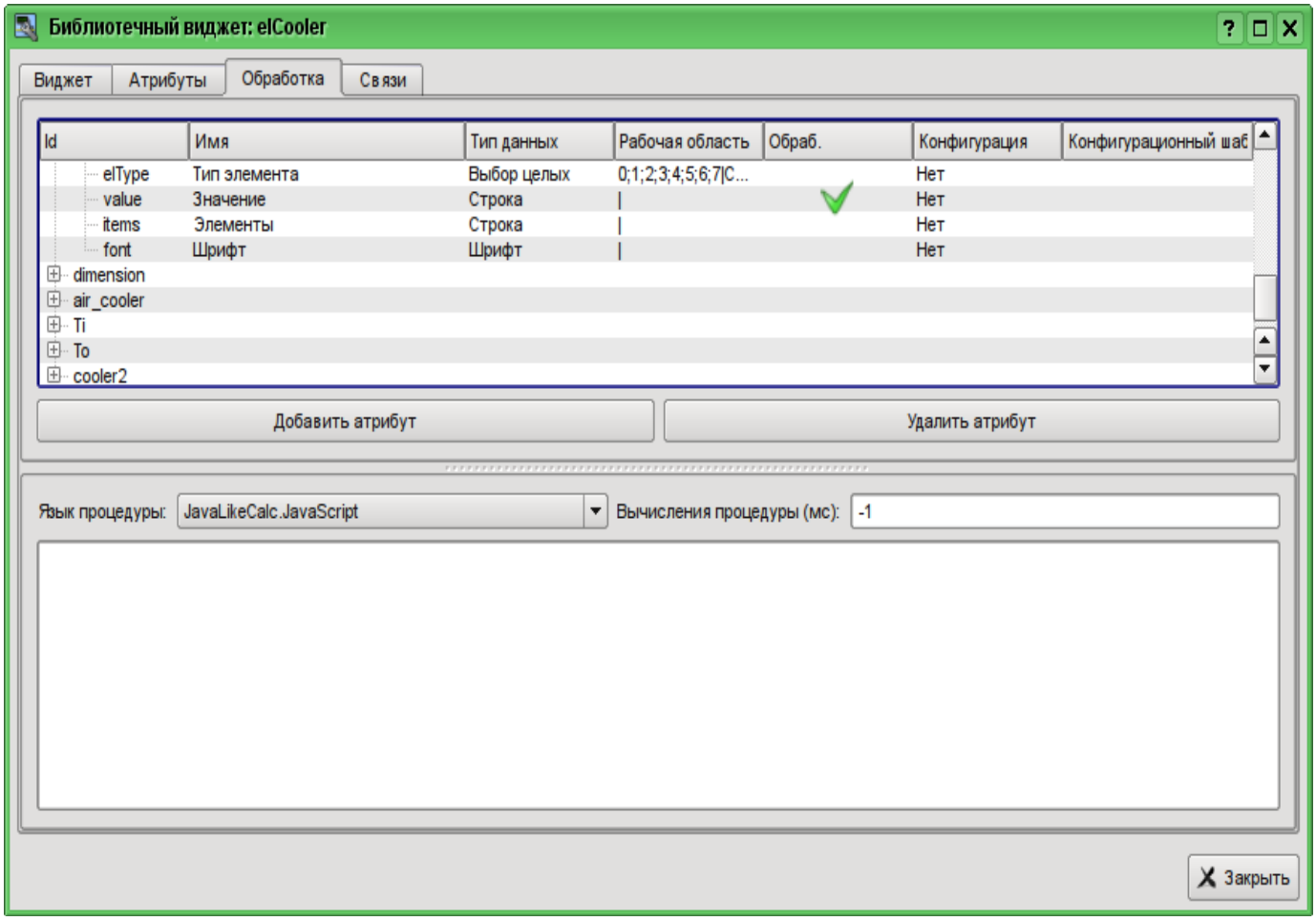

Рис. 5.3.2.15. Включение в обработку атрибута "value" комбобокса "cw".

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

```
Ti_arg0val = Ti;
To_arg0val = To;

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

⚠ Помещение или редактирование программы виджета не приводит к непосредственной её компиляции, а значит не будет сообщений об ошибках в программе, если они имеют место быть. Это связано с тем, что непосредственное исполнение программы, а значит и её компиляция, осуществляется в окружении и в момент запуска на исполнение проекта визуализации. При этом все ошибки, возникшие при компиляции, выводятся в виде сообщений OpenSCADA, а виджеты с ошибками не исполняются. Посмотреть архив сообщений OpenSCADA можно в главной вкладке подсистемы "Архивы" или в терминале запуска OpenSCADA, если запуск был из терминала или его эмулятора.

Результирующий вид вкладки обработка виджета "elCooler" библиотеки "KM 101" будет иметь вид, показанный на рис. 5.3.2.16.

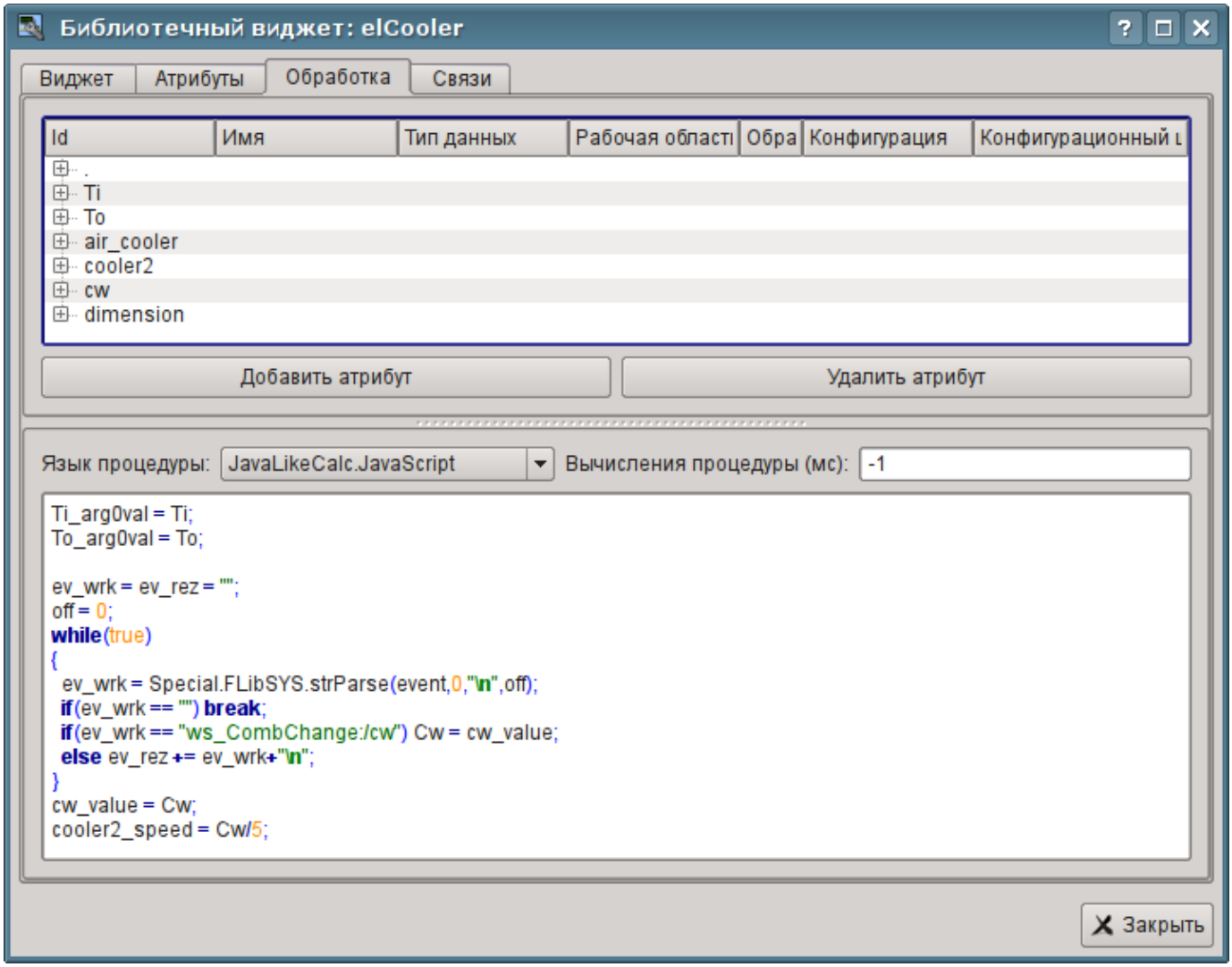

Рис. 5.3.2.16. Результирующий вид вкладки обработка виджета "elCooler" библиотеки "KM 101".

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

На этом разработку комплексного элемента можно считать законченной.

5.3.3. Добавление комплексного элемента на мнемосхему

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

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

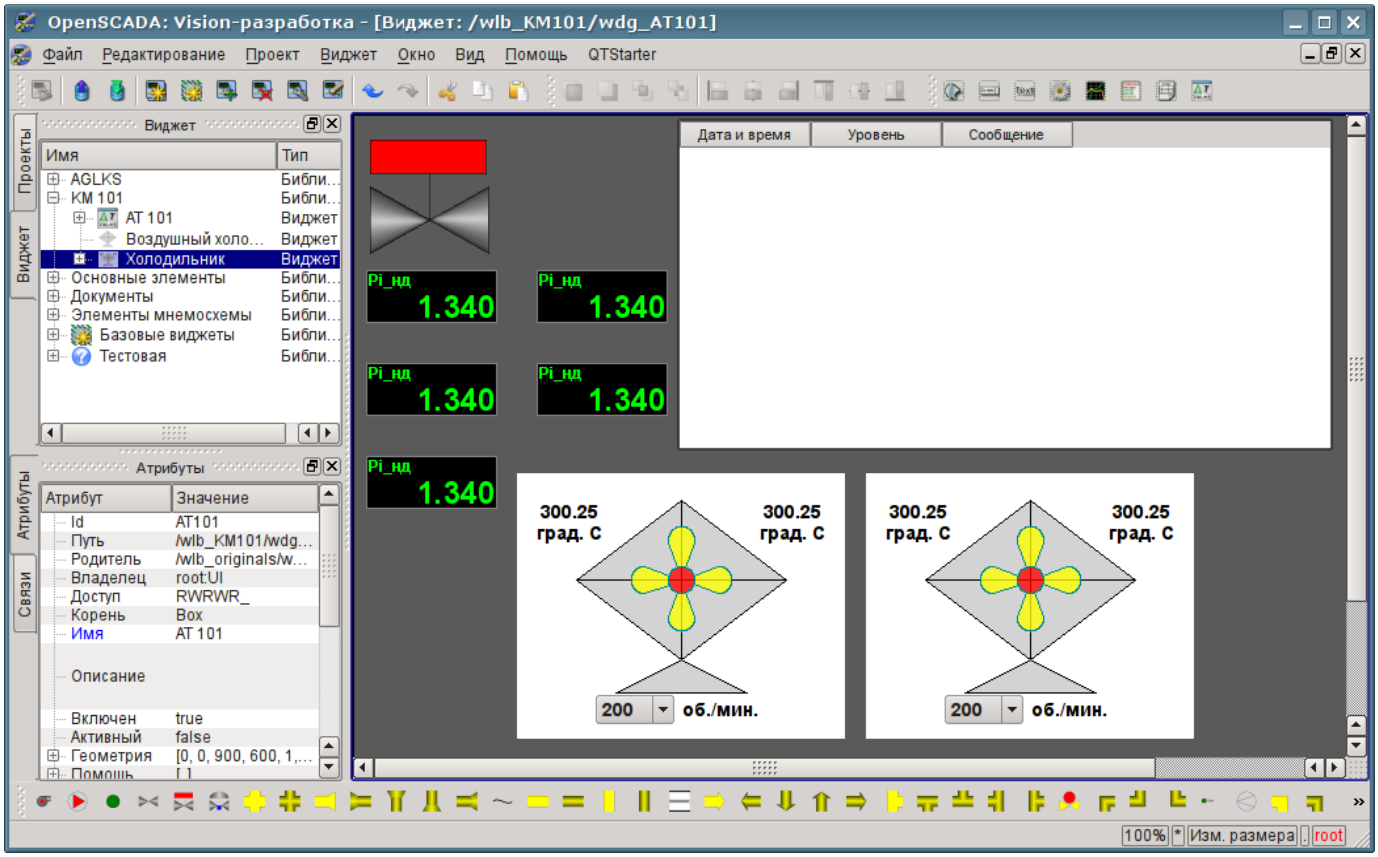

Рис. 5.3.3.1. Вид мнемосхемы с комплексными элементами.

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

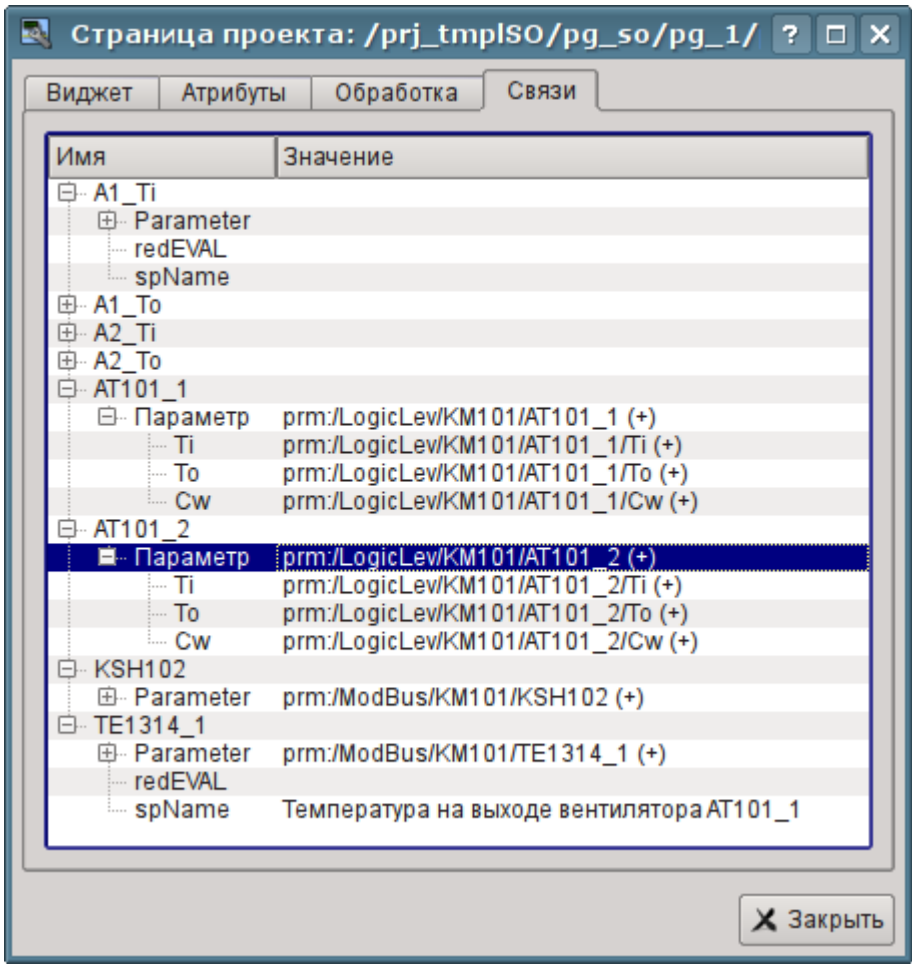

Рис. 5.3.3.2. Вкладка "Связи" диалога редактирования свойств мнемосхемы.

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

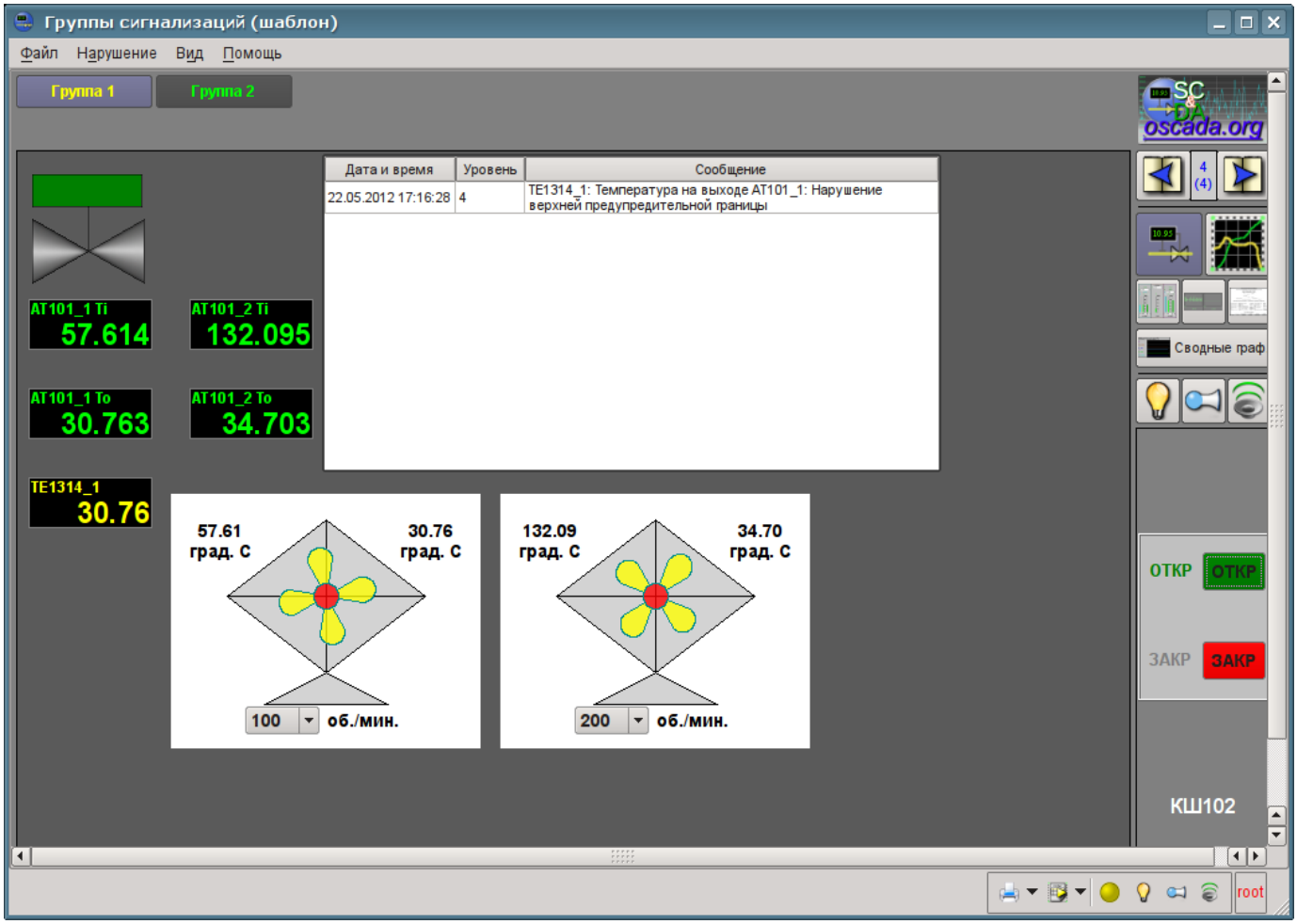

Рис. 5.3.3.3. Результирующая мнемосхема.

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

6. Рецепты

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

6.1. Перенос конфигураций OpenSCADA из одного проекта в другой

Часто востребованной бывает задача переноса конфигурации из одного проекта OpenSCADA в другой. Причём, чаще всего нужно осуществить частичный перенос, например, перенос отдельных наработок, которые могут пригодиться в новом проекте.

Вообще, нужно отметить, что любые наработки с малейшим намёком и перспективой вторичного использования нужно стараться унифицировать и сохранять в отдельные, собственные, библиотеки и БД. Крайне не рекомендуется непосредственно менять стандартные конфигурации и элементы стандартных библиотек, а также сохранять собственные, новые, библиотеки и элементы в базах данных стандартных библиотек. Это позволит Вам впоследствии безболезненно обновлять стандартные библиотеки, а также просто использовать наработки ваших предыдущих проектов.

Простой перенос БД с библиотеками и конфигурацией

Если Вы учли вышеуказанные рекомендации и все Ваши унифицированные наработки содержаться в отдельной БД, то весь процесс переноса будет заключаться в копировании БД и подключении её в новом проекте.

Процедура копирование БД отличается для различных типов БД и с ней нужно будет ознакомиться из документации к БД. Для сопутствующего распространения с дистрибутивами OpenSCADA обычно используется БД SQLite, в виде отдельных файлов *.db. Копирование БД SQLite соответственно заключается в простом копировании нужного файла БД из директории баз данных старого проекта в директорию баз данных нового.

Подключение осуществляется путём создания нового объекта БД в модуле нужного типа БД подсистемы БД и последующей его конфигурации (детальнее). После создания, конфигурации и включения БД можно сразу загрузить конфигурацию из неё, нажав кнопку "Загрузить систему из этой БД" на форме объекта БД.

Выделение нужной конфигурации

В случае, если нужная конфигурация содержится в общей БД или БД стандартных библиотек, то предварительно нужно осуществить выделение её в отдельную БД. Выделить конфигурацию можно или в отдельную БД с Вашими библиотеками или в экспортную БД. Экспортная БД, в отличие от библиотечной, служит только для переноса конфигурации и будет впоследствии удалена. В любом случае нужно создать новую БД для нужного типа БД, подобно процедуре подключения выше. Для переноса нужно использовать тип БД, который планируется использовать в новом проекте. Обычно для переноса лучше использовать тип БД SQLite, ввиду простой процедуры копирования. Однако, если использовать сетевую СУБД эта процедура может превратиться в простое подключение библиотечной или экспортной БД в новом проекте.

Далее необходимо выделить конфигурацию в унифицирующие или экспортные библиотеки, если она не может быть прямо сохранена в БД. Например, отдельные шаблоны параметров или параметры контроллеров сбора данных, визуальные элементы библиотек виджетов и т.д. выделить можно путём создания библиотеки экспорта или унификации соответствующего элемента, например, библиотека шаблонов или контроллер параметров сбора данных, библиотека виджетов и т.д. Для вновь созданной библиотеки в качестве БД нужно указать ранее созданную унифицирующую или экспортную БД. Далее осуществляется копирование нужных элементов из исходной библиотеки в унифицирующую/экспортную посредством стандартной функции копирования. После копирования унифицирующую/экспортную библиотеку нужно сохранить.

В случае необходимости переноса объекта с отдельным свойством БД, или целых библиотек, операцию создания промежуточной библиотеки и последующего копирования можно опустить. Достаточно в поле БД указать ранее созданную унифицирующую или экспортную БД и сохранить элемент.

Дальнейшие действия, а именно простой перенос БД, осуществляются в соответствии с предыдущим разделом.

При переносе конфигурации путём экспортирования необходимо осуществить обратный процесс копирования из экспортных библиотек в локальные библиотеки нового проекта и удаление экспортной БД.

Низкоуровневое копирование содержимого БД

Для переноса можно осуществить избирательное копирование таблиц БД с конфигурацией путём выбора объектов таблиц в объекте БД; команды копирования, выбора объекта новой БД и команды вставки (детальнее). Однако, для этого нужно знать структуру БД, про которую изложено по этой ссылке.

6.2. Особенности циклического программирования в OpenSCADA

У начинающих пользователей часто возникает вопрос обеспечения выдержки временных интервалов при программировании вычислительных процедур в окружении OpenSCADA. Этот вопрос обычно связан с наличием предыдущего опыта программирования линейных вычислений и отсутствием опыта программирования циклических систем реального времени.

В системах реального времени используется, так называемое такт или цикл периодических вычислений, т.е. ритм жизни. В каждом такте выполняется некоторая процедура, которая не должна занимать времени больше такта. Как следствие, если процедура такта останавливается на ожидании то останавливается и жизнь системы реального времени. Следовательно использование в этих процедурах традиционных функций усыпления задачи недопустимо!

Решение задачи выдержки нужного интервала времени в системах реального времени, в рамках ритма жизни, осуществляется в два способа. Первый способ заключается в декременте счётчика, установленного в значение временного интервала, в каждом цикле на значение периодичности такта до значения <=0, например, в OpenSCADA это реализуется таким образом:

```
if((tm_cnt-=1/f_frq) <= 0)   //Декремент
{
    tm_cnt = 10; //Установка счётчика в значение 10 секунд
    //Выполнение других действий с периодичностью 10 секунд
}
```

Второй способ основан на астрономическом времени, т.е. в цикле осуществляется сравнение с текущим временем, например, в OpenSCADA это реализуется таким образом:

```
if(SYS.time() > tm_to)
{
      tm_to = SYS.time()+10;  //Установка порога ожидания на 10
секунд более текущего времени
    //Выполнение других действий с периодичностью 10 секунд
}
```

Второй способ является более надёжным поскольку в нём исключается проблема запаздывания срабатывания по причине возможного выполнения процедуры цикла более времени такта. Хотя в правильно настроенных системах и задачах данный эффект не должен иметь место.

6.3. Живой диск (Live CD/USB)

С целью максимального упрощения развёртывания OpenSCADA можно использовать живые сборки загрузочных дисков CD и USB. Живой диск предусматривает возможность загрузки прямо с него и быстрого получения желаемого рабочего окружения. При загрузке и работе живой диск не использует штатных носителей, а значит Вы можете не беспокоиться за их целостность и сохранность данных на них. В целом живой диск является удобным средством с широким кругом нужных программных инструментов под рукой и независимого от стационарного программного окружения, способного осуществить диагностику программного и аппаратного окружения, а также их восстановление, в некоторых случаях.

Живой диск представляет из себя упакованный образ операционной системы и прикладных программ с размером около 700МБ, записанный на CD/DVD диск или USB-Flash носитель. При своей работе операционная система "налету" распаковывает нужные файлы для запуска программ и открытия документов, а значит не использует оперативной памяти больше чем при её обычной установке.

Живые диски с OpenSCADA собираются в нескольких вариантах на основе дистрибутивов ОС Linux ALTLinux и доступны для загрузки к соответствующей версии OpenSCADA здесь: http://oscada.org/ru/glavnaja/zagruzit. Текущие живые сборки с OpenSCADA обладают значительно большими функциями чем предусматривалось изначально:

- Возможность прозрачного сохранения рабочих изменений, в случае записи на USB-Flash. Функция обеспечивается созданием раздела диска с возможностью записи, на оставшемся месте USB-Flash. Этот раздел отражается на корень файловой системы и все изменения записываются на него. Кроме сохранения рабочих данных в этот раздел можно доустанавливать недостающие пакеты программ из репозитория ALTLinux (последний P6 и T6).
- Возможность совмещения обычного Flash-диска данных с живым Flash-диском. Предусматривает запись образа живого диска прямо на файловую систему USB-Flash, FAT16 или FAT32, что сохраняет функции обычного носителя данных и добавляет функцию живого диска.
- Возможность установки окружения живого диска на стационарный носитель. Позволяет не заниматься глубоким изучением и погружением в операционную систему Linux при её установке, настройке, а так-же развёртывания OpenSCADA. Достаточно загрузиться с живого диска, убедиться что основное оборудование определилось нормально и нужные программы работают, а затем, посредством простой процедуры с иконки на рабочем столе, установить на стационарный носитель. Полученная установка будет точно повторять окружение живого диска.

ISO-образ живого диска

Первый вариант сборки живого диска это ISO-образ (*LiveCD_USB.iso) для записи на CD/DVD диск, который однако является комбинированным и может быть записан прямо на USB-Flash диск.

Для записи ISO-образа на CD/DVD диск можно использовать стандартный инструментарий исходной операционной системы. Запись на USB-Falsh может быть осуществлена только из окружения ОС Linux, например, из окружения этого-же живого диска, записанного и загруженного ранее с CD/DVD диска.

⚠ Пользователь не имеющий опыта работы с ОС Linux должен ограничиться записью на CD/DVD диск или начать знакомиться с Linux, если хочет получить живой USB-Flash диск.

⚠ Запись образа на USB-Flash уничтожит все данные и сделает его непригодным для использования в качестве носителя данных, если не учитывать возможность записи на раздел сохранения изменений окружения ОС живого диска, который будет создан при первой загрузке с живого диска.

Адрес диска для записи ISO-образа имеет вид "**/dev/sd{x}**", а узнать его можно вызовом консольной команды "**$ dmesg**", сразу после подключения целевого диска USB-Flash. Для записи ISO-образа на USB-Flash из окружения Linux можно воспользоваться консольной командой:

```
$ dd if=ALTLinux_6-OpenSCADA_0.8.0-TDE_3.5.13-i586-LiveCD_USB.iso of=/dev/sd{x} bs=4096
# Запись прямо с загруженного CD/DVD диска
$ dd if=/dev/sr0 of=/dev/sd{x} bs=4096
```

FAT-образ живого диска

Второй вариант сборки живого диска это FAT-образ: группа файлов для прямой записи и загрузки с FAT-раздела (*flash.tar). Преимуществом этой сборки, как ранее упоминалось, является совмещение функции USB-Flash диска как носителя данных и как живого диска. Кроме того на основе данной сборки можно создавать компактные, надёжные и функциональные решения встраиваемых систем с OpenSCADA в основе, например, Программируемые Логические Контроллеры (ПЛК), панельные контроллеры (с сенсорным дисплеем), а так-же просто SCADA-сервера и SCADA-станции оператора "быстрого приготовления"; путём записи живого диска на стационарный носитель (HDD, SSD или Flash). Надёжность данного решения достигается путём размещения основного ПО в немодифицируемом упакованном файле, а рабочих данных на журналируемой файловой системе.

Запись данного образа можно осуществить из любой ОС, но установить загрузчик только в Linux, для чего можно воспользоваться LiveCD из прошлого раздела.

Процедура создания живого диска такова:

```
# Подключаем целевой диск, узнаём его адрес и монтируем, всё от суперпользователя:
$ su -
$ dmesg
$ mkdir /mnt/tmp; mount /dev/sd{x}1 /mnt/tmp
# Распаковка содержимого архива прошивки на смонтированный диск:
$ cd /mnt/tmp; tar xvf /var/tmp/ALTLinux_6-OpenSCADA_0.8.0-TDE_3.5.13-i586-flash.tar
# Узнаём UUID для файловой системы целевого диска:
$ blkid | grep /dev/sd{x}1
# Модифицируем файл /mnt/tmp/syslinux/syslinux.cfg в конце
# строки "append initrd=alt0/full.cz live ... disk,uuid:4EB3-0478",
# где указываем ранее полученный UUID # Добавляем или
# модифицируем файл "/mnt/tmp/syslinux/lang" на предмет
# указания локали-языка интерфейса по умолчанию,
# для Русского языка нужно указать "ru_RU", иначе будет Английский.
# Отключение диска:
$ umount /dev/sd{x}1
# Инициализация MBR диска в корректное значение:
$ ms-sys -s /dev/sd{x}
# Инициализация загрузчика:
$ syslinux /dev/sd{x}1
```


Данный способ развёртывания живого диска требует знаний ОС Linux и интерфейса командной строки (консоли), а так-же основ разбиения дисковых носителей поскольку, при некорректном начальном разбиении носителя, загрузка может не пройти. Кроме того, для обеспечения функции прозрачного сохранения рабочих изменений необходимо создать раздел с меткой "alt-live-storage" и файловой системой ext3, что можно сделать программе менеджера разделов, например "**gparted**".

Загрузка

Для загрузки с полученного живого диска нужно перегрузить компьютер и нажать клавишу входа в меню загрузки BIOS, в самом начале загрузки, до загрузчика стационарной ОС, и выбрать там наш носитель (рис.6.3.1). На разных компьютерах клавиша входа в меню загрузки может отличаться и быть одной из "F8", "F9", "F10", "F11" или "F12".

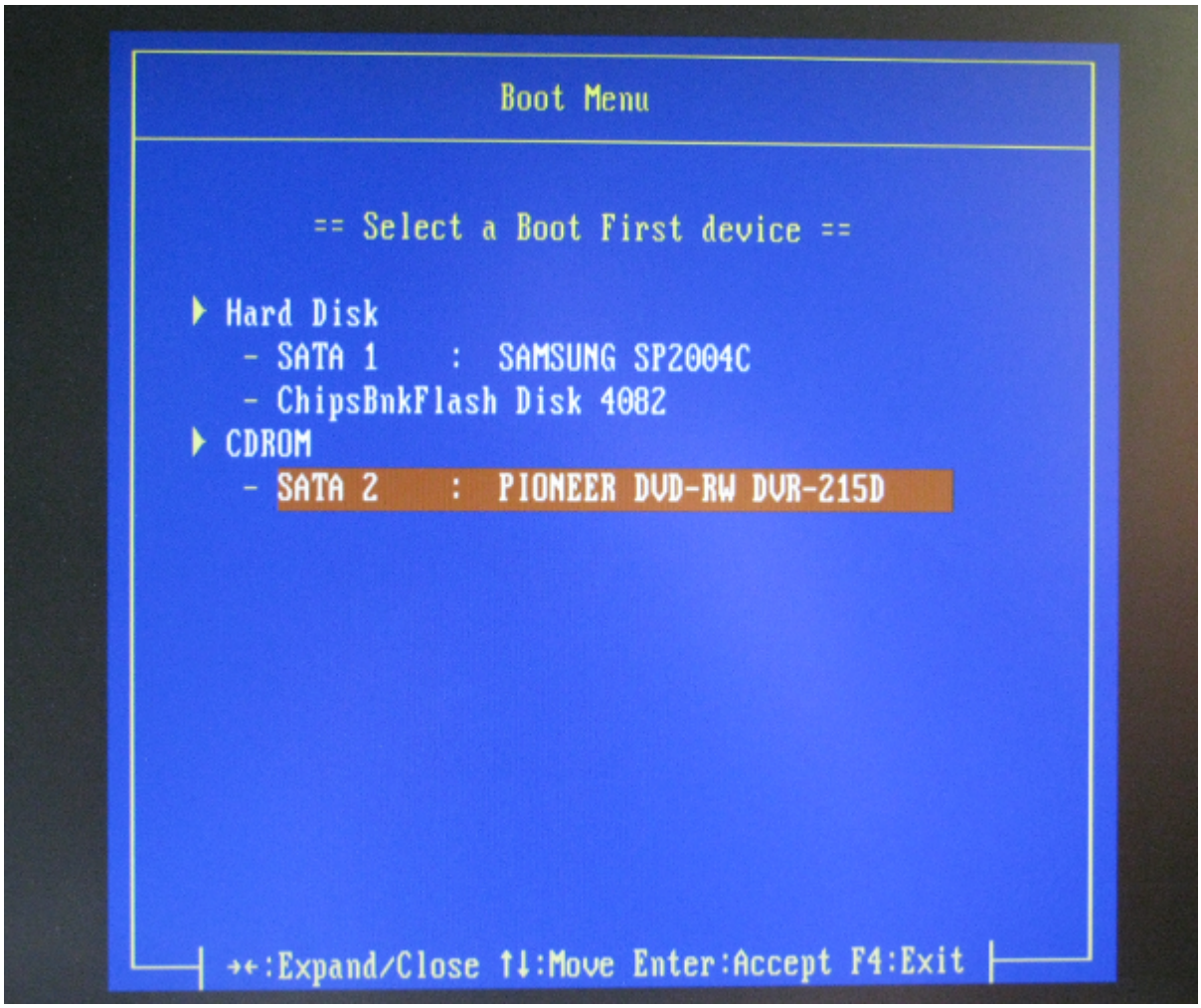

Рис. 6.3.1. Меню выбора устройства загрузки в BIOS.

После выбора устройства должно появиться меню загрузки живого диска, где предварительно важно выбрать Ваш язык, по клавише F2 (рис.6.3.2), если язык по умолчанию не устраивает.

Рис. 6.3.2. Меню выбора языка живого диска.

В результате загрузки живого диска Вы получите рабочий стол TDE 3.5.13 (рис.6.3.3).

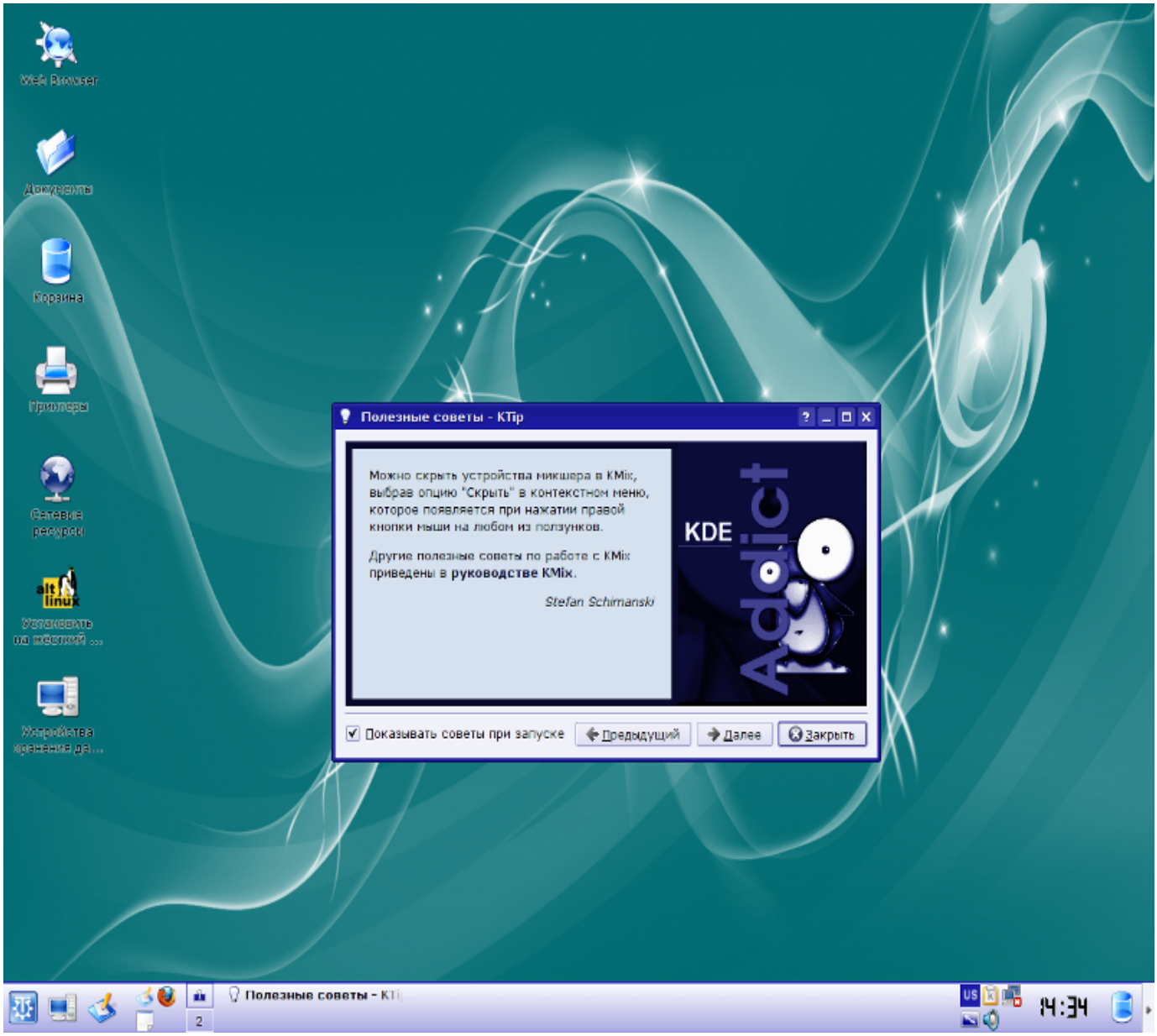

Рис. 6.3.3. Рабочий стол живого диска.

6.4. Общие положения концепции работы с нарушениями, сигнализацией и уведомлениями

Нарушения и работа с ними в OpenSCADA реализуется двояко, что связано со структурой OpenSCADA, способами её использования, а так-же самой природой нарушений.

Первой стороной нарушений, с которыми OpenSCADA работает изначально и которая наиболее востребована, является уведомление различными способами. Поскольку уведомление это часть интерфейса визуализации то и реализованы они в движке визуализации UI.VCAEngine и визуализаторах UI.Vision, UI.WebVision. На данный момент подсистема уведомлений нарушений в OpenSCADA реализует функции:

- Уведомление:
  - *Светом* — мигание объекта, группы сигнализации, общего статуса цветом нарушения.
  - *Звуком* — проигрывание звукового файла или синтез речи из текста, связанного с нарушением;
  - *Гудком* — выдача непрерывного сигнала на системный "бузер", независимо от нарушения.
- Квитация уведомления нарушения:
  - *Полностью* — по нажатию на цветной мигающий кружок статуса нарушений (событие "ws_alarmLev"), снизу справа:
  - *По способу уведомления* — отдельно световую (событие "ws_alarmLight"), звуковую (событие "ws_alarmLight") и гудок (событие "ws_alarmAlarm"), по нажатию кнопки с соответствующим изображением, снизу справа или под кнопками видов отображения;
  - *По объекту нарушения* — к образу визуального представления может добавляться команда квитации уведомления непосредственно по нём;
  - *Поочерёдно, с выслушиванием* — характерно для уведомлений звуком, поскольку каждый объект нарушения может предоставлять своё звуковое уведомление или синтез речи.

При реализации уведомлений в среде визуализации нет установленного правила получения и формирования признака нарушения поскольку в разных ситуациях нет однозначности. На данный момент, на стороне типизированных шаблонов источника данных, практикуется способ формирования атрибута ошибки "err" с кодами и текстом нарушения, а обработка их и формирование уведомления осуществляется в визуальном образе объекта данных. Иногда-же обработка границ уставок осуществляется прямо в визуальном образе объекта данных.

Впоследствии возникла необходимость протоколирования, а также учёт актуальных на текущий момент нарушений. Если для протоколирования достаточно формирования системных сообщений с оговоренной категорией и форматом сообщения, то для контроля за текущими нарушениями необходим некий буфер. Впоследствии такой буфер был добавлен в виде надстройки над подсистемой сообщений, а адресация к нему осуществляется инверсией уровня сообщения. Так запись сообщения с уровнем "-2" и категорией "TEST" поместит сообщение в буфер нарушений и продублирует его в архиве сообщений, с уровнем "2". При запросе сообщений с отрицательным уровнем сообщения будут браться из буфера нарушений. Удаление/снятие нарушения осуществляется записью сообщения с той-же категорией "TEST" и неотрицательным уровнем.

Такая концепция учёта активных нарушений позволяет использовать стандартные механизмы работы с сообщениями для учёта нарушений:

- Регистрация нарушения: *SYS.message("alCategory", -3, "Параметр: нарушение");*
- Снятие нарушения: *SYS.message("alCategory", 1, "Параметр: норма");*
- Формирование списка активных нарушений: посредством примитива "Протокол" или "Документ", с отрицательным уровнем, для всех "-1".

Регистрацию сообщений лучше всего осуществлять на стороне типизированных шаблонов источника данных посредством специальной функции "**SYS.DAQ["Modul"] ["Controller"].alarmSet(string mess, int lev = -5, string prm = "")**", которая унифицирует категорию. Для вызова этой функции из контекста шаблона нужно добавить IO "this" типа "Объект", после чего установка нарушения будет иметь вид "**this.nodePrev().alarmSet("Параметр: нарушение", -5, "prm");**". Указанная функция сейчас используется во многих модулях источников данных для учёта глобальных нарушений объектов контроллеров. Функция формирует нарушение с категорией: **al{ModId}:{CntrId}[.{PrmId}]**, где:

- *ModId* — идентификатор модуля;
- *CntrId* — идентификатор контроллера;
- *PrmId* — идентификатор параметра, из аргумента <*prm*>.

В целом нужно отметить, что уведомления и регистрация нарушений это разные механизмы, которые могут использоваться по отдельности, для простых проектов, или вместе для больших комплексных проектов.

Заключение

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